University of West Bohemia
Faculty of Applied Sciences

TEXT-MINING WITH LINKED DATA

Ing. Martin Dostal

doctoral thesis
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in specialization Computer Science and Engineering

Supervisor: Prof. Ing. Karel Ježek, CSc.

Department: Department of Computer Science and Engineering

Pilsen 2014

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

TEXT-MINING S VYUŽITÍM LINKED DATA

Ing. Martin Dostal

disertační práce
k získání akademického titulu doktor
v oboru Informatika a výpočetní technika

Školitel: Prof. Ing. Karel Ježek, CSc.

Katedra: Katedra informatiky a výpočetní techniky

Plzeň 2014

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě aplikovaných věd Západočeské university v Plzni.

Prohlašuji tímto, že tuto práci jsem vypracoval samostatně, s použitím odborné literatury a dostupných pramenů uvedených v seznamu, jenž je součástí této práce.

V Plzni dne 26. 10. 2014 Martin Dostal

Abstract

This thesis proposes the progress in the area of text-mining realized with methods improved by semantic information from Linked Data. This approach is demonstrated on well-known text-mining tasks like feature extraction, classification and clustering. This approach is evaluated with common available data corpuses and with my own several corpuses in cases when the large enough corpuses were not available or were not suitable for an experiment. The standard explored data sets include:

- 20 News Groups,
- Reuters-21578,
- The Open Directory Project,
- WOS data collection for citation analysis,
- data collections from Stanford University.

These corpuses were created and they are available on demand for other academic researchers:

- Collection of Call for papers announcements - approx. 18 000
- Newspaper collections:
    - BBC,
    - The New York Times,
    - CNN,
    - The Washington Post.

Anyway some of the proposed methods had to be evaluated manually because the convenient corpus was not available and its creation would be quite challenging.

This thesis also covers some experiments from my other areas of interest close to text-mining and that are related to my field of study. These experiments were realized with my coworkers and they include infometrics, citation analysis and enhancement of PageRank-style graph algorithms.

Keywords: text-mining, Linked Data, clustering, classification

Abstrakt

Tato práce představuje můj vývoj v oblasti text-miningu realizovaný s využitím sémantické informace získané z Linked Data. Tento přístup je demonstrován na dobře známých text-miningových úlohách jako je volba vlastností, klasifikace a shlukování. Tento přístup je vyhodnocen s využitím běžných datových kolekcí a s využitím několika vlastních korpusů v případech, kdy dostatečně velké korpusy nebyly k dispozici nebo nebyly vhodné pro daný experiment. Standardní datové kolekce zahrnují:

- 20 News Groups,
- Reuters-21578,
- The Open Directory Project,
- Kolekci článku z WOS pro citační analýzu,
- Datové kolekce ze Stanford University.

Následující korpusy byly vytvořeny v průběhu mého doktorského studia a jsou na základě žádosti k dispozici pro ostatní vědecko-výzkumné pracovníky:

- Kolekce Call for papers oznámení – přibližně 18 000
- Kolekce novinových článků:
    - BBC,
    - The New York Times,
    - CNN,
    - The Washington Post.

Některé navržené metody, prezentované v této práci, však musely být vyhodnoceny manuálně z důvodu neexistence vhodného korpusu, jehož vytvoření by bylo značně náročné.

Tato práce pokrývá i některé další experimenty, které se přímo netýkají text-miningu, ale které jsou této oblasti velmi blízké. Tyto experimenty byly realizovány s mými kolegy a zahrnují infometrii, citační analýzu a vylepšení grafových algoritmů typu PageRank.

Klíčová slova: text-mining, Linked Data, shlukování, klasifikace

Contents

List of Figures

List of Tables

1 Introduction

The Internet presents a huge amount of information that is usually formatted and published primary for a human. Due to the heterogeneity and the lack of structure of web documents, access to this huge collection of information has been limited to browsing and searching. Moreover the volume of electronic documents is rapidly increasing, which leads to a requirement for their ingenious digital processing. Therefore the robust information extraction (IE) systems that are able to transform human readable data to software friendly content are welcome and furthermore essential for many areas of human activity. Although different approaches for data extraction have been developed, there are still limitations for their wider use. These systems are usually focused on specific tasks which require test data instead of generally applicable approaches with simple and fast configuration.

1.1 Text mining background

Progress in digital data processing and storage technology has resulted in the growth of huge databases and data collections. This has occurred in all areas of human activity, from credit card transactions to government statistics. The interest has grown in the possibility of analyzing these data, of extracting from them information that might be of value to the owner of the database or community in the case of public sources. The discipline concerned with this task has become known as data mining [1].

Data mining is the analysis of data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful. Data mining typically deals with data that have already been collected for some purpose then the data mining analysis. This means that the objectives of the data mining play no role in the data collection strategy. This is one way in which data mining differs from much of statistics, in which data are often collected by using efficient strategies to answer specific questions. For this reason, data mining is often referred to as “secondary data analysis” [1].

The process of seeking relationships within a data set involves a number of steps:

- Determining the nature and structure of the representation to be used.
- Deciding how to quantify and compare how well different representations fit the data.
- Choosing an algorithmic process to optimize the score function.
- Deciding what principles of data management are required to implement the algorithms efficiently.

Text documents are important sources of information [1] and data mining methods can help in retrieving useful text from large data collections. However, research in information retrieval has traditionally focused more on facilitating information access rather than analyzing information to discover patterns, which is the primary goal of text mining. The goal of

information access is to connect the right information with the right users at the right time with less emphasis on processing or transformation of text information. Text mining [2], also referred to as text data mining, can be regarded as going beyond information access to further help users analyze and digest information and facilitate decision making. Moreover text mining refers to the process of deriving high-quality information from collected text documents. A typical collection of documents for text mining evaluation is Reuters-21578[1]. Each document in this collection is a short newswire article.

1.2 Motivation and state of the art

One of the popular approaches to organize textual data is the use of clustering algorithms which divide documents into coherent clusters. The goal of clustering methods is to identify distinct groups in a dataset. The basic implementation of clustering puts together documents that share many terms. More generally, it puts together documents with similar features. The cluster hypothesis [3] says that documents in the same cluster behave similarly with respect to the relevance to information needs. The hypothesis states that if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant.

The exploration of Internet resources is even more challenging than local text-document processing because of uncertainty as far as the quality of documents. Instead of making high quality web pages, some authors aim to make their pages rank highly by playing with the Web page features that search engine ranking algorithms are based on. This behavior is usually called search engine spam [4] [5]. Very often the unwanted pages have a higher ranking than the expected information sources. This is caused by the preparation of texts with regard to routine statistical analysis methods that should increase the position of an unwanted page in a search results. Search engine ranking systems were designed to prevent these search engine optimization (SEO) techniques.

The most famous ranking algorithms are PageRank [6] and HITS [7]. Those algorithms are based on the theory that the most important pages on the Internet are those pages with the most links leading to them. Links can be marked as votes. The importance of the page that contains the link and the number of outgoing links is also considered.

This thesis is focused on approaches for text data organization and processing with semantic information automatically gathered from Web. PageRank will be used for graph processing in the form of nodes from Linked Data. Therefore the main areas of interest for this thesis are:

- Feature extraction – these features are used in clustering and classification,
- Clustering – used in searching and spam detection algorithm,
- Classification – especially for tasks where we need to divide documents into existing classification classes,

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

- Semantic analysis of software specifications – text-mining and semantic analysis are used for checking and guidance of development of these documents,
- Experiments with PageRank – PageRank is used for graph processing in different information extraction tasks.

1.2.1 Feature extraction and feature selection

Data objects that are the subject of analysis using machine learning techniques are usually described by a large number of features. Therefore it is often beneficial to reduce the dimension of the data. Dimension reduction is beneficial for its computational efficiency and moreover for accuracy of the analysis. These techniques for dimension reduction can be divided into:

- Techniques that apply supervised or unsupervised learning,
- Feature selection or feature transformation (e.g. extraction) techniques.

Traditional algorithms used in machine learning are often susceptible to the well-known problem of the *curse of dimensionality*, which refers to the degradation in the performance of a given learning algorithms as the number of features increases. To deal with this issue, dimension reduction techniques are often applied as a data preprocessing step. This typically involves the identification of a suitable low-dimensional representation for the original high-dimensional data set. By working with this reduced representation, tasks such as classification or clustering can produce more accurate and readily interpretable results, while computational costs are also significantly reduced. The motivation for dimension reduction can be summarized as follows:

- The identification of a reduced set of features that are predictive of outcomes can be very useful from a knowledge discovery perspective.
- For many learning algorithms, the training and classification time increases directly with the number of features.
- Noisy or irrelevant features can have the same nuance on classification as predictive features so they will impact negatively on accuracy.

As mentioned before, there are two main designs for dimension reduction. The first design decision is whether to select a subset of the existing features or to transform to a new reduced set of features. The other design is focused on question of whether the learning process is supervised or unsupervised. Example of these designs:

- Supervised feature selection – e.g. information gain (IG),
- Unsupervised feature transformation – e.g. principal component analysis (PCA).

1.2.1.1 Feature selection

Some examples of methods for feature selection [8] are:

- Document frequency-based selection,
- Term strength,
- Entropy-based ranking,
- Information gain (IG).

Other methods for feature selection include correlation coefficients [8], mutual information [9], SVM [10] or X^2.

Document frequency [1] based selection is the simplest possible method for feature selection to filter out irrelevant features. This type of selection may not be sufficient to reduce the noise effects of very frequent words, known as stop words, such as “a”, “an”, “the” and so on. Therefore these stop words have to be removed based on vocabulary. Also infrequent words can be removed because they have no meaning for document clustering or classification. TF-IDF is one of the well-known algorithms based on document frequency.

Term strength is used to measure how informative a word is for identifying two related documents. For two related documents is the term strength defined in terms of the following probability:

$$s(t) = P\ (t\ \in x\ |t\ \in y) \tag{1.1}$$

Where:

- *x* and *y* are two related documents
- *s(t)* – term strength of term *t*

The entropy-based ranking approach is based on measure how big will be entropy reduction when the term is removed. The entropy is defined as follows:

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n}\left(S_{ij}.\log\left(S_{ij}\right) + \left(1 - S_{ij}\right).\log(1 - S_{ij})\right) \tag{1.2}$$

Where:

- *E(t)* – entropy of the term *t*,
- *n* – number of documents in a collection.

IG [11] measures the amount of information that each argument contains about the other. For term *t* and category *c*, information gain is defined as:

$$IG(t,c) = \sum_{x\in\{t,\bar{t}\}}\sum_{y\in\{c,\bar{c}\}} P(x,y)\ log\frac{P(x,y)}{P(x)P(y)} \tag{1.3}$$

When *t* and *c* are independent, IG(t,c) = 0.

1.2.1.2 Feature transformation

Feature transformation refers to techniques that transform the original features of a data set to an alternative, more compact set of dimensions. These techniques can be divided into:

- Feature extraction – it involves the production of a new set of features from the original features in the data, through the some kind of mapping. Well-known unsupervised feature extraction methods include Principal Component Analysis (PCA) and spectral clustering.
- Feature generation – it involves the discovery of missing information between features in the original dataset and the enrichment of that set with additional features that emphasize the newly discovered information.

Our approach for feature selection with Linked Data, proposed in this thesis, covers the combination of feature selection and feature transformation techniques.

1.2.2 Clustering

The aim of the clustering algorithm [1] [12] is to divide objects into semantically related groups called clusters. These clusters are coherent internally, but clearly different from each other. In other words, documents within a cluster should be as similar as possible; and documents in one cluster should be as dissimilar as possible from documents in other clusters [3]. The problem of clustering can be very useful in the text domain, where the objects to be clusters can be of different granularities such as documents, paragraphs, sentences or terms.

Clustering is the most common form of unsupervised learning. No supervised learning vision means that there is no human expert who has assigned documents to classes. In clustering, it is the distribution and makeup of the data that will determine cluster membership [3].

Clustering is primarily used in searching and browsing, which may be utilized either for internal processing of documents, or even better for the presentation of results to the user. The main pioneers of semantic search using clustering include search engines Yippy[2] and Dogpile[3]. In the first case, it is already deployed in practice very good multilingual clustering tool for automatic processing of the Google, Yahoo! and Bing search results. This method is very convenient, because the user is not forced to go through hundreds of search results. He just limits search results to a cluster, which intrigued him.

Clustering finds applicability for a number of tasks:

- Document organization and browsing – for example the hierarchical organization of documents into coherent categories can be very useful for systematic browsing of the document collection.

[2] Yippy search engine. http://yippy.com.
[3] Dogpile search engine. http://dogpile.com.

- Corpus summarization – clustering techniques provide a coherent summary of the collection in the form of cluster-digests [1] or word-clusters, which can be used in order to provide summary insights into the overall content of the underlying corpus.

Typical cluster models include:

- Connectivity models – e.g. hierarchical clustering based on distance connectivity.
- Centroid models – e.g. k-means algorithm represents each cluster by a single vector.
- Distribution models - clusters are modeled using statistical distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.
- Density models - for example DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- Subspace models - in Biclustering (co-clustering), clusters are modeled with both cluster members and relevant attributes.

There are two basic types of clustering algorithm:

- Hard clustering - each document is assigned to exactly one cluster.
- Soft clustering - each document can be assigned to multiple clusters.

According to the number of levels, we can distinguish:

- Flat clustering - all clusters are on the same level.
- Hierarchical clustering - clusters are combined to form a hierarchical.

Now we will focus on connectivity models also known as distance-based clustering algorithms. These models are usually used for evaluation in this thesis and they will be discussed in next sections.

1.2.2.1 Distance-based clustering algorithms

Distance-based clustering algorithms are designed by using a similarity function to measure the closeness between the text objects. The most well-known similarity function which is used commonly in the text domain is the cosine similarity function. Computation of text similarity is a fundamental problem in information retrieval. Although most of the work in information retrieval has focused on how to assess the similarity of a keyword query and a text document, rather than the similarity between two documents, many weighting heuristics and similarity functions can also be applied to optimize the similarity function for clustering [1].

Common approaches for distance-based clustering algorithms are:

- Hierarchical clustering algorithms – single linkage clustering, group-average linkage clustering and complete linkage clustering.
- Distance-based partitioning algorithms – k-medoid clustering algorithms and k-means clustering algorithms.

- The Scatter-Gather method.

The Scatter-Gather method was mentioned in the introduction of this section and this method plays a big role in a motivation of this work. The goal of scatter-gather is a better user interface for searching and browsing.

This algorithm clusters the whole collection to get groups of documents that the user can select or gather. The selected groups are merged and the resulting set is again clustered. This process is repeated until a cluster of interest is found. An example is shown in Figure 1.1.

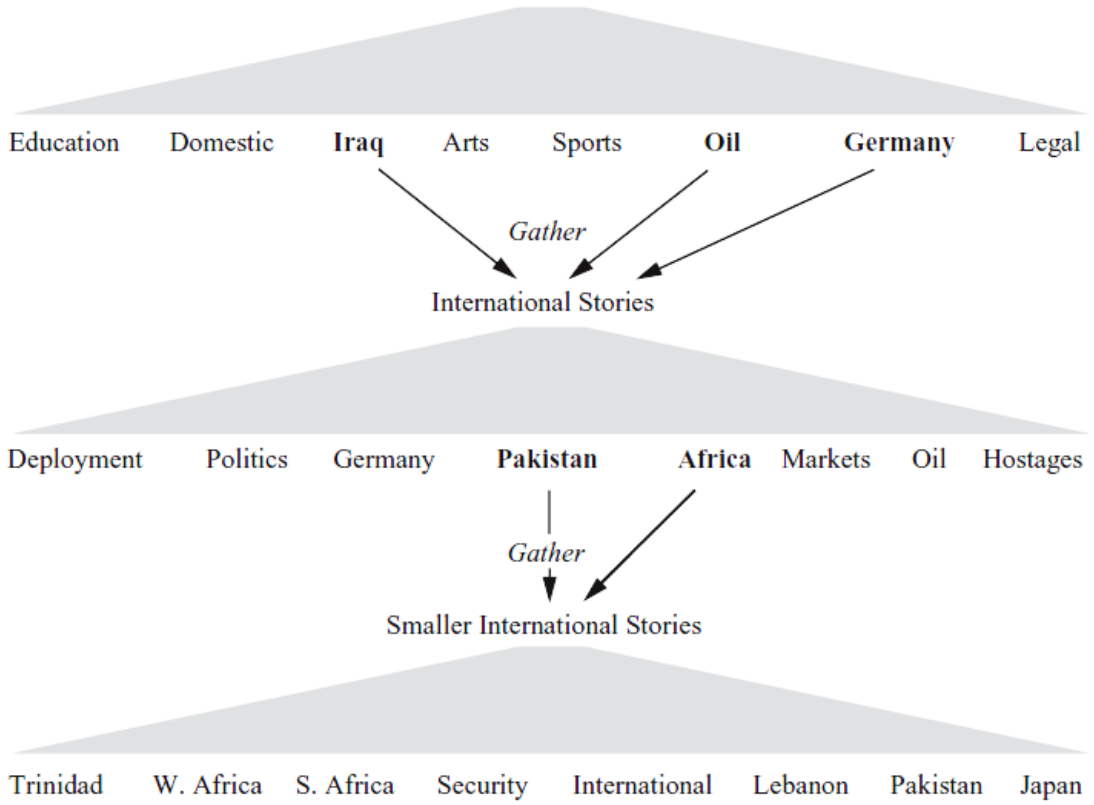

Figure 1.1: The Scatter-Gather [3]

Automatically generated clusters are not as neatly organized as a manually constructed hierarchical tree like the Open Directory Project[4] (ODP). Also, finding descriptive labels for clusters automatically is a difficult problem which will be discussed in this thesis.

The cluster-based navigation is an interesting alternative to keyword searching, the standard IR paradigm. This is especially true in scenarios where users prefer browsing over searching because they are unsure about which search terms to use [3].

[4] http://dmoz.org

1.2.3 Classification

The problem of classification is defined as follows. We have a set of training records $D = \{X_1, \ldots, X_N\}$, such that each record is labeled with a class value drawn from a set of k different discrete values indexed by $\{1 \ldots k\}$. The training data is used in order to construct a classification model, which relates the features in the underlying record to one of the class labels [1].

For a given test instance for which the class is unknown, the training model is used to predict a class label for this instance. There are two versions of classification:

- Hard,
- Soft.

In the *hard* version of the classification problem, a particular label is explicitly assigned to the instance, whereas in the *soft* version of the classification problem, a probability value is assigned to the test instance. Soft version allows to assign one document to more classes.

Other variations of the classification problem allow ranking of different class choices for a test instance, or allow the assignment of multiple labels to a test instance.

There have been many supervised learning techniques for document classification. Some of these techniques include:

- Naive Bayes [3],
- k-nearest neighbors [13],
- vector approaches - e.g. Rochio,
- support vector machines,
- boosting [14],
- rule learning algorithms [15],
- Maximum Entropy [16],
- and Latent semantic analysis [17].

Naïve Bayes is defined as:

$$P(c \mid d) = P(c) \prod_{1 \le k \le n_d} P(t_k \mid c) \tag{1.4}$$

Where:

- t_k – term from class c,
- c – class,
- d – document.

Best class is chosen using formula:

$$c_{map} = \arg\max_{c \in C} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k \mid c) \quad (1.5)$$

Where:

- $\hat{P}(c)$ and $\hat{P}(t_k \mid c)$ is based on approximation data from the training corpus

Another approach to document classification [18] proposed term graph model as improved version of the vector space model. The aim of this model is to represent the content of a document with relationship between keywords. This model enables to define similarity functions and PageRank-style algorithm. The vectors of PageRank score values were created for each document. The rank correlation and the term distance were used as a similarity measures to assign document to classification class.

An alternative approach to document classification uses hypernyms and other directly related concepts [19] [20]. Next step in document classification can be marked as feature expansion with additional semantic information from ontology [21]. This approach [21] is exploiting the external knowledge for mapping terms to regions of concepts. For exploration of related concepts, the traversal graph algorithm is used.

Some examples of domains in which text classification is commonly used are as follows [1]:

- News filtering and organization - most of the news services today are electronic in nature in which a large volume of news articles are created every day by the organizations. In such cases, it is difficult to organize the news articles manually. Therefore automated methods can be very useful for news categorization in a variety of web portals and information services.
- Document organization and retrieval - the above application is generally useful for many applications beyond news filtering and organization. A variety of supervised methods may be used for document organization in many domains. These include large digital libraries of documents, web collections, scientific literature, or even social feeds. Hierarchically organized document collections can be particularly useful for browsing and retrieval.
- Opinion mining and sentiment analysis - customer reviews or opinions are often short text documents which can be mined to determine useful information from the review. Sentiment analysis is focused on decision about positive or negative meaning of the review.
- Email classification and spam filtering - it is often desirable to classify email in order to determine either the subject or to determine junk email in an automated way. This is also referred to as spam filtering or email filtering.

1.2.4 Semantic analysis of software specifications

A software requirements specification (SRS) is a description of the behavior and appearance of a system to be developed. It includes different requirements and use cases that describe interactions between users and the software. The requirements are most often elicited from the user by the analyst through the use of some type of an interview [22]. The root of the requirements problems lies in the common ground between the user and the analyst, which can only be discovered through communication activities that facilitate a sharing of information [23]. The most expensive failures in software projects have their roots in requirements specifications. Misunderstanding between analyst, domain experts and users is very common. They may have different reasons but now we are focused on the fact that these SRS are widely adopted using just natural language without any additional semantic markup for automatic evaluation and inspection.

Essentially software is determined by its functional and non-functional characteristics [24]. Functional characteristics describe software behavior that can be directly implemented and evaluated by a common programmer. Furthermore, software specification documents are usually focused on functional characteristics, because it is easy to understand and describe application behavior to the customer. Unfortunately, there has been a lop-sided emphasis in the functionality of the software, even though the functionality is not usable without the necessary non-functional requirements.

These requirements are usually hidden and it occurs at the time when it is least suitable. It includes usability, interoperability, flexibility, performance and software security. Real-world problems are more non-functionally oriented than they are functionally oriented, so even great application could be unusable if it disregards the typical use. These problems include high cost and slow processing, poor productivity, lower profit and it leads to unhappy customer.

1.3 Problem statement and goals

The main goals for this thesis are:

G1 *Propose a feature selection method with additional semantic information.*

There are a number of different statistically oriented methods for feature selection. It includes correlation coefficients [8], mutual information [9] or SVM [10]. These methods require relatively big training collection that is often missing. This goal focuses on situations with small training set with regard to semantically correct and human-readable features.

G2 *Application and evaluation of feature selection from goal G1 to classification and clustering.*

The aim of this goal is to evaluate an effect of feature selection with Linked Data to standard text-mining tasks like classification and clustering especially on problems with small training sets.

G3 *Cluster labeling with Linked Data.*

Currently there are two main approaches to cluster labeling: differential cluster labeling and cluster internal labeling. Differential cluster labeling [25] compares clusters with each other and chooses terms that maximally distinguish the individual clusters. Cluster internal labeling [26] computes a label that depends on the cluster itself, not on other clusters. The common method is to label a cluster with the title of the document closest to the centroid. This goal focuses on human-readable cluster labeling that maximally distinguish clusters and semantically covers the topics of the articles within a cluster.

G4 *Software specification analysis with Linked Data.*

The aim of this goal is to process software specifications and help users, analytics and developers to maintain and check requirements during software lifecycle. This approach can be primary used in agile software development during the whole sw lifecycle for consistency evaluation between individual iterations.

G5 *Citation network analysis with PageRank.*

This goal is focused on approach for research evaluation in the form of citation analysis with PageRank. New modifications of PageRank have been developed in previous goals thus we looked for other opportunities of application. Nowadays automatic research evaluation techniques are a hot topic for grant agencies. Therefore new modifications of PageRank applied on citation networks for evaluation of research quality, contribution and significance is very beneficial.

1.4 Cooperation with other members of Text-Mining research group

We realized that nowadays it is not possible to work alone. So we have combined our skills with my college Michal Nykl together to handle significantly larger problems that would not be manageable alone. It led to a number of joint publications, where I was focused on semantic web stuff and Michal was responsible for graph processing with PageRank.

Chapter 7 “PageRank variants in the evaluation of citation networks” was realized together with Michal Nykl and previously published in [27]. I was responsible for WoS data preprocessing and evaluation with Turing and Codd awards. PageRank variations formulas were constructed primary by Michal.

Now we have to declare skills of each partner that will illustrate the responsibilities in each article. My skills are related to: Semantic Web, Linked Data, web technologies, web crawlers, storage methods (relational and no-sql databases), data processing and semantic data analysis. Michal Nykl is focused on: scientometrics, PageRank, iterative computation and graph processing.

1.5 Structure of the thesis

Chapter 2 is focused on introduction to Semantic Web and Linked Data. This chapter covers the current state of the art in named entity recognition with Linked Data and related methods.

Chapter 3 introduces related work to keywords extraction. These methods are expanded with my own approaches for keywords and keyphrases extraction. The first idea about implementation of Linked Data as source of semantic information to tag-suggestion system was proposed. This was the breaking point developed in the first year of PhD study. It leads to the next Linked Data experiments.

Chapter 4 is devoted to clustering from basic definition to my own methods. These methods solve very hot problems related to this text-mining task. First of them is feature selection with Linked Data and the second of them is cluster labeling with Linked Data. Feature selection is constructed with semantic information therefore it’s able to group related documents without statistical methods and with no exactly matching terms contained.

Chapter 5 covers classification algorithm from definition to my own experimental implementation of feature selection. Completely new method for feature selection with Linked Data and PageRank will be proposed and evaluated. Basic features are expanded with Linked Data and PageRank is used for selection of the most important features for each document.

Chapter 6 introduces a new approach for semantic analysis of software specifications. Nowadays there is no completely working possibility how to verify software requirements

specifications against check list or against previous versions in iteration lifecycle of a document. This approach is not a silver bullet but it is the first step for document verification with semantic information gathered from public sources. Moreover this approach is able to combine different public and private data sources together.

Chapter 7 covers experiments realized with other members of Text-mining research group focused on PageRank and citation networks.

2 Semantic Web and Linked Data

This chapter was previously published as part of these articles: [28] [29] [30]. In this chapter the most important terms will be defined:

- Semantic Web – the extension of the existing WWW,
- Linked Data – the method for publishing of structured data,
- DBpedia Spotlight – the important webservice used in the next chapters.

The Semantic Web is a collaborative movement led by W3C. It is an extension of the existing World Wide Web and it provides a standardized way of expressing the relationships between web pages to allow machines to understand the meaning of published information. The Semantic Web aims at converting the current unstructured web documents into a "web of data". It is regarded as an integrator across different content published on the Web. Linked Data describes a method of publishing structured data to accomplish the aim of the Semantic Web.

2.1 Linked Data

The concept of Linked Data [31] was first introduced by Tim Berners-Lee. He set up four rules for machine readable content on the Web:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information using the standards (RDF*, SPARQL[5]).
- Include links to other URIs so that they can discover more things.

More specific is the idea of Linked Open Data[6] (LOD), which is based on the presumption of freely published data without restrictions in usage or additional fees.

The Linked Data initiative has given rise to an increasing number of RDF[7] documents as well as other machine-readable sources, many of which are freely accessible online. These resources are often created as a result of database exports. That is the reason why we have to deal with duplicate information sources. There are two basic problems with duplicates resources: disambiguation and co-reference resolution.

[5] http://www.w3.org/TR/rdf-sparql-query/
[6] http://lod-cloud.net
[7] http://www.w3.org/RDF/

These problems were discussed in [32]. DBLP[8] and DBpedia[9] are two of those common Linked Data resources often used for academic research.

Linked Data contains information about a resource and moreover links to other related resources. There are two basic types of links that we can directly use:

- Parent-child relation,
- links to synonyms.

These connections are bidirectional so a child can find his parent and a parent can find his children. Relations are described by ontology predicates. For example: “dbpedia-owl:genre”, “skos:broader”, “dcterms:subject”. The meaning of these predicates differs, but we can use it in the same way. Synonyms are designated by the ontology relation: “*owl:sameAs*“, which indicates true synonyms, and the relation “*skos:related*“, which indicates related concepts.

2.2 DBpedia Spotlight

DBpedia Spotlight [33] is an open source software designed for automatic processing, analyzing and semantic enrichment of a text. It automatically annotates mentions of DBpedia resources in text, and goes through the whole analysis life cycle. It consists of entity detection (spotting), candidate selection and disambiguation based on related concepts.

UseCase: **Buyer** searches for an **offer**
Scope: Marketplace
SuD: Marketplace **Information System**
Primary actor: Buyer
Main **success scenario specification**:
1. **Buyer** enters basic **search criteria**
2. **System** responds with the list of **matches**
3. **Buyer** requests the complete listing of a selected **offer**
4. **System** responds with the requested information

Extensions:
2a. No **matches** found
2a1. Use case aborted
Sub-**variations**:
2b. The amount of **matches** i stoo high
2b1. **Buyer** narrows the **search results** with additional criteria

Figure 2.1: DBpedia Spotlight annotation

[8] http://dblp.uni-trier.de
[9] http://dbpedia.org

DBpedia Spotlight is not just a tool but moreover it can be used as a web service. Nowadays there are these REST endpoints available:

- Spotting - this service takes text input and recognizes the potential surface forms e.g. names of entities (Figure 2.1). Several spotting techniques are available, such as dictionary lookup and Named Entity Recognition (NER). The output is the list of annotations in structured form like XML or JSON.
- Annotate – runs spotting and disambiguation. It retrieves the candidate DBpedia resources, disambiguates them if needed, and links the mentions to the best one. These output formats are available: XML, JSON, HTML, RDFa and NIF.
- Candidates – similar as annotate, but does not disambiguate the candidates for each mention. Rather it returns a ranked list of candidates. This list contains attributes in the form of scores expressing the significance of the word. There are different scores based on links from other resources and the significance in current context.
- Disambiguate – does not do spotting, it just selects the candidates for the given mentions and does disambiguation. This web service will be deprecated soon.

JSON (JavaScript Object Notation) is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML. It was originally derived from the JavaScript scripting language. Nowadays JSON is a language-independent data format and JSON data is readily available in a large variety of programming languages. Example of JSON data:

```
" entities ": [{
        " entity ":" Tim Berners - Lee",
        " type ":" Person ",
        " uri ":" http :// dbpedia . org / resource /Tim_berners_lee ",
        " nerdType ":" http :// nerd . eurecom .fr/ ontology #Person ",
        " startChar ":30 ,
        " endChar ":45 ,
        " confidence ":1 ,
        " relevance ":0.5
}]
```

DBpedia Spotlight works in four-stages:

- Spotting stage – it recognizes in a sentence the phrases that may indicate a mention of a DBpedia resource.
- Candidate selection – it is subsequently employed to map the spotted phrase to resources that are candidate disambiguations for that phrase.
- Disambiguation - uses the context around the spotted phrase to decide for the best choice amongst the candidates.

- Configuration - the annotation can be customized by users to their specific needs through configuration parameters.

We will discuss only first three of them because there are the most important for understanding of our approach.

2.2.1 Spotting step

This step uses a lexicon that was generated from the extended set of labels from the lexicalization dataset. The implementation uses the LingPipe[10] Exact Dictionary-Based Chunker which relies on the Aho-Corasick string matching algorithm [34] with longest case-insensitive match.

2.2.2 Candidate selection

The aim of this phase is to map resource names from spotting to candidate disambiguation. The DBpedia Lexicalization dataset was used for determining candidate disambiguation for each surface form. This phase can also be viewed as a way to pre-rank the candidates for disambiguation before observing a surface form in the context of a paragraph. The DBpedia resource with the highest prior probability for a surface form is selected as the default sense according to its usage in Wikipedia.

DBpedia resource contains:

- Common names for resource
- Redirects from other resources – alternative spellings, aliases
- Disambiguation pages – link a common term to other resources

2.2.3 Disambiguation

The Inverse Candidate Frequency (ICF) was introduced in Spotlight [33] for disambiguation – see Formula 2.1. This measure supposes that the discriminative power of a word is inversely proportional to the number of DBpedia resources it is associated with.

$$ICF(w_j) = \log \frac{|R_s|}{n(w_j)} = \log|R_s| - \log n\ (w_j) \qquad (2.1)$$

Where:

- R_s is set of candidates for a surface form *s*
- $n(w_j)$ is total number of resources in *Rs* that are associated with the word w_j

[10] http://alias-i.com/lingpipe/

2.3 Named entity recognition using Linked Data

In this section, previously published in [30], we describe a number of practical approaches [35] that commonly constitute annotations in different documents. The aim of this section is to demonstrate the relation between phrase recognition and Linked Data. These approaches are used as part of the DBpedia Spotlight webservice therefore is essential to shortly introduce these techniques right after DBpedia Spotlight introduction. We will start from the simplest and go through the most powerful ones that take into account the word context obtained from Linked Data. These approaches will be used directly and discussed in the next chapters.

2.3.1 Lexicon-based phrase recognition

A simple approach for phrase recognition is the usage of a string matching algorithm that relies on a lexicon of name variations for the target terms in the knowledge base. The lexicon-based phrase recognition does not select phrases with regard to their context but just searches for any phrases known as possible DBpedia entities. This method produces a high number of false positives. One example is the set of function words that have entries on Wikipedia, but whose annotation would be undesirable in use cases such as blog annotation, because it would confuse the reader with too many unnecessary links. However, eliminating those phrases from the lexicon upfront is not an option, as they may have other significant meanings. For example the word ‘up’ can be a fiction word in some contexts or name of a movie “UP” by Pixar.

2.3.2 Noun-phrase chunk heuristic

In many cases, the objective of annotation is to mark the things being talked about in text. Therefore, a simple heuristic to eliminate false positives early in the process is to only annotate terms that are within noun phrases. We therefore extended the Lexicon-based phrase recognizer with a simple heuristic that only allows phrases that contain at least one noun.

2.3.3 Noun-phrase chunking with probabilistic dictionary

This method is similar to previous one. It assumes that we are considering only noun-phrases. However, instead of heuristic, it uses NP chunks extracted by a NP chunker. For each NP chunk it chooses the longest expression contained in the set of acceptable phrases (in the lexicon).

The method can be enriched for detecting common words like “do” or “make”. The Wikipedia guidelines explicitly instruct users to “avoid linking plain English words”. Therefore it will be a good idea attempt the detection of common words at phrase recognition time.

2.3.4 Named entity recognition (NER)

In use cases such as the concept tagging in blog posts or online newspapers, we are usually focused on specific types like people and places. In these cases it is viable to apply named entity recognizers as a strategy for phrase recognition.

NER extended by noun phrase n-grams was proposed by [36]. It is a hybrid approach mixing named entities and more general terms within noun phrase chunks. It consider as phrases only the expressions marked as named entities by the NER phrase recognizer, the noun-phrase chunks extracted by a NP chunker, and all sub-expressions of up to 5 tokens of the noun-phrase chunks. This increases the coverage of the NER phrase recognizer, which tends to generate fewer phrases.

2.3.5 NER with Linked Data

We distinguish two types of named entity extractors as in [37]. First type is able to identify interesting keywords and classify them in taxonomy. Second type is able to additionally provide a link pointing to URI that disambiguates the named entity.

The most widely used Linked Data named entity extractors are: AlchemyAPI[11], DBpedia Spotlight[12], Lupedia[13], OpenCalais[14], Saplo[15] and Zemanta[16]. Next we will compare some of these tools.

AlchemyAPI, OpenCalais and Zemanta offer both free and commercial versions of access to their web services. Commercial and free versions are basically the same, but there are limitations in a number of free requests per day. On the other hand DBpedia Spotlight is completely free without any restrictions. Table 2.1 contains basic information about these popular Linked Data tools.

Table 2.1: Basic information about selected LD tools.

	Alchemy API	Spotlight	OpenCalais	Zemantha
No languages	8	3	3	1
Entity types	272	272	39	81
No LOD dataset	7	1	9	1

[11] http://www.alchemyapi.com
[12] http://dbpedia-spotlight.github.com/demo/
[13] http://lupedia.ontotext.com/
[14] http://www.opencalais.com
[15] http://saplo.com/
[16] http://www.zemanta.com

The creators of the DBpedia Spotlight have compared their web service with a number of other NER extractors according to a particular annotation task [33]. The experiment consisted in evaluating 35 paragraphs from 10 articles in 8 categories selected from "The New York Times" and has been performed by 4 human raters. The final goal was to create wiki links. The experiment showed how DBpedia Spotlight overcomes the performance of other services to complete this task. Therefore it was used as a primary annotation tool in our case.

3 Keywords extraction

This chapter is focused not just on problematic of keywords extraction but rather on keyphrase extraction. The main difference between these two terms stands in the result of the algorithm. The output of keywords extraction algorithm is the list of important words with no regard to other words. On the other hand the output of keyphrase extraction algorithm is list of phrases that consists of two or more words that occurs together.

The last part of this chapter will cover the application of keyphrase extraction for an automatic tagging based on Linked Data. The methods for keywords and keyprase extraction were used as a first step for the auto-tagging approach.

3.1 Related work to keyphrase extraction

Keyphrase extraction is used to extract most frequent words which are significant for the document with respect to the application. Keyphrase extraction is frequently used in search engines and other text mining tasks. Some of the common tools and approaches for key phrase extractions are:

- Carrot 2 – it uses two algorithms: STC (Suffix Tree Clustering) and lingo. STC [38] is an incremental, linear time algorithm, which creates clusters based on phrases shared between documents. Lingo [39] is based on SVD and search complete keyphrase with some other constraints keyphrases. With STC, lingo also uses TF-IDF and LSA. Lingo provides flexibility in input. Indexed documents can be obtained by Nabble, Solr or google search desktop.
- KEA [40] – it is a standard algorithm for keyphrase extraction. It provides provision of learning from RDF dictionary in SKOS format. The dictionary contains hierarchical taxonomy and it also gives options for machine learning by a tool Weka [40]. It is a common prediction algorithm for Named Entity Recognition.
- Maui – it is basic KEA tool but also gives options to boost taxonomy from Wikipedia.
- Stanford topic modeling tool – this tool uses LDA for learning topic. It takes input and output in CSV format. It also provides options for Machine learning.

3.2 Related graph algorithms

In this section, we would like to introduce graph-based ranking algorithms, especially Google's PageRank [41] and its implementation for text document processing.

Google's PageRank [41] is the first and most popular graph-based ranking algorithm which has been successfully used by social networks, citation analysis, and link-structure analysis of the World Wide Web. This algorithm is a way of deciding on the importance of a node within a graph. The importance of a node is evaluated based on global information recursively drawn

from the graph. The graph node is important when it is often recommended by other nodes. In this special case, if other web documents contain links in the form of URI to this one.

The TextRank graph-based algorithm is a ranking model for graphs extracted from text documents. The text is split into tokens which represent the nodes of the graph. Nodes are connected with weighted edges based on the lexical or semantic relations, for example. TextRank algorithms use a co-occurrence relation controlled by the distance between word occurrences within a sliding window of maximum N words. The best results were achieved for a maximum of two words which correspond with N-gram based algorithms.

3.3 Automatic keyphrase extraction based on NLP and statistical methods

In this section, previously published in [42], we would like to present our experimental approach to automatic keyphrase extraction based on statistical methods and Wordnet-based pattern evaluation.

Automatic keyphrases are important for automatic tagging and clustering because manually assigned keyphrases are not sufficient in most cases. Keyphrase candidates are extracted in a new way derived from a combination of graph methods (TextRank) and statistical methods (TF*IDF). Keyword candidates are merged with named entities and stop words according to NL POS (Part Of a Speech) patterns. Automatic keyphrases are generated as TF*IDF weighted unigrams. Keyphrases describe the main ideas of documents in a human-readable way. Evaluation of this approach is presented in articles extracted from News web sites. Each article contains manually assigned topics/categories which are used for keyword evaluation.

Our approach is very useful in cases where we don't have manual keywords assigned by the author or where these keywords are not sufficient. Our approach builds on ideas from TextRank [43] and RAKE [44] with a combination of statistical (TF*IDF) and NLP methods.

Keywords are defined as a sequence of one or more words and provide a compact description of a document's content. Keywords are often used to define queries within information retrieval systems because they are easy to define, remember, and share. Keywords are usually corpus independent and can be applied across multiple different systems. For example, the Phrasier [45] system lists documents related to a primary document's keywords and supports keyword anchors as hyperlinks between documents. The Keyphind system [46] uses keywords as the basic building block for an IR system especially for summarization and clustering tasks. Hulth [47] (2004) describes Keegle, a system that provides extracted keywords for web pages found by a Google search engine.

Keyphrases consist of two or more keywords and named entities. In our approach, keyphrases consist of keywords, named entities and stop words. Stop words can be an important part of a keyphrase, which increase the readability and intelligibility of a phrase in natural

language. This idea was inspired by the RAKE system for automatic keyword extraction from individual documents.

3.3.1 Algorithm for automatic keyword extraction

TextRank can be used in individual documents without any other knowledge. Graph nodes ranking is made upon co-occurrences of tokens and the score of the other nodes with edges to the current one. In our algorithm, the TF*IDF score is used for better text token evaluation instead of the measure based only on n-grams. The next idea is based on two versions of the keyword characteristic. The keyword extraction problem can be divided into:

- **Individual keyword extraction** – individual words with a special and important meaning, generally in the form of a noun or named entity.

- **Keyphrase extraction and derivation** – phrases contain two or more keywords and other information in a human-readable form. This phrase can contain verbs and stop words for better readability. Short phrases can be joined by their co-occurrence percentage number.

Our algorithm can be divided into three main steps:

1. Text preprocessing
2. Keyword extraction
3. Keyphrase extraction

During the text preprocessing phase, the article's content is divided into tokens and non-significant characters are removed. Named entities are recognized by the elementary method: the first upper-case letter with corpus-based statistic is good enough for this recognition. The tokens are divided by their POS tag and generally only nouns and adjectives can be declared as potential keywords. The next idea is to choose only common words instead of unique ones. Keywords are often used for clustering and a keyword is useless when it is assigned to only a few articles. This is the reason why we remove tokens with low frequency and set up rules for general nouns. There is no such rule for named entities. The remaining tokens and named entities are declared as keywords candidates. We calculate the TF*IDF score only for these keyword candidates.

The keyword extraction phase contains only the method for removing useless candidates whose TF*IDF score is lower than 1/5 of a maximal value. This boundary can be changed by the number of requested automatic keywords.

The keyphrase extraction part can be described by these steps:

- NLP method – interesting n-grams are chosen. The choice is based on their POS tag patterns and the corpus frequency is counted only for these n-grams. These n-grams can be marked as keyphrase candidates.

- A score of importance is counted for each keyphrase candidate. This score contains the n-gram corpus frequency and TF*IDF score for each word. The score is used for document keyphrase selection.

- Derivation – keyphrase candidates are merged with named entities or individual keywords if their co-occurrence is significant for this document.

Used POS patterns:

1) POS patterns for 3-grams:

- (N or named entity), (V or A or stop word), (N or named entity)
- 3x (named entity)
- example: Tim Berners Lee

2) POS patterns for 2-grams

- A, (N or named entity)
- 2x (named entity)
- example: Bill Gates

Used tags:

- N – noun,
- V – verb,
- A – adjective,
- "stop word" – a word from stop list,
- "named entity" – a potential named entity based on simple patterns like word in the middle of the sentence with first capital letter.

Keywords and keyphrases are ordered by their TF*IDF score and only the most important keywords and keyphrases are used. The number of used items naturally depends on the requested number of these items. This number can be chosen directly, or by percentage measure from the score of the best item.

3.3.2 Evaluation of keyphrase extraction

To evaluate performance, we tested our system against a collection of newspaper articles about technology that were extracted from the Web. These articles contain manually assigned keywords from a controlled dictionary. These keywords were chosen by the author of the article. Such keywords are marked as topics in our case. These topics cover the article content very sketchily and capture only the basic idea of the article. For example, the content of the article contains the word "Google", but the manually assigned topic was "Google Inc". This can be enough for article classification, but for automatic keyword evaluation it is a further challenge.

We have decided that our news corpus is a collection from the "real world" and if the results are satisfying, it would be usable for other general data collections. Another reason was that this approach is very experimental and we had no better article collection for evaluation. Initially, we thought that these results would not be satisfactory, but we were surprised by their high relevance. The size of the data sets was chosen based on the number of the articles that were available for the each subset evaluation.

We have used three data sets:

A) Corpus of 500 articles with a small number of manually assigned topics (600 different topics) by the author.

B) Corpus of 50 random articles with a small number of manually assigned topics. Each article contains approximately 3 manual topics.

C) Corpus of 50 random articles with manually assigned topics (by author) and expanded, by 2-3 another human annotators, to other important topics covered by the article. Each article contains approximately 5 manual topics.

The statistics of precision and recall for part A) are shown in Table 3.1. These values are calculated for a different boundary configuration of accepted keywords. This threshold is set as the percentage difference from the score of the most important keyword. Precision and recall are reduced because of topic classification difficulties. There are about 600 various topics in this data set and some of them are very similar for the human annotator, but not for our topic classification module.

The topic classification is done only based on the name of the topic, so for example: topics "Google Inc." and "Google operating system" have the same score for these automatic keywords: "Google" and "Android".

Table 3.1: Precision and recall for the corpus of 500 articles.

Boundary	**1%**	**10%**	**30%**	**50%**	**70%**	**90%**
Precision	13.2%	18.9%	27%	31.8%	38.2%	40.8%
Recall	46.1%	33%	22.6%	16.5%	12.8%	10.53%

The statistics of precision and recall for part B) are shown in Table 3.2. This corpus contains 50 random original articles. The main difference between evaluation A) and B) is in the number of classification topics. There are about 120 topics used for classification so the precision and recall are not distorted as much as in part A).

Table 3.2: Precision and recall for the corpus of 50 articles.

Boundary	**1%**	**10%**	**20%**	**30%**	**40%**	**50%**	**70%**	**90%**
Precision	30%	38.6%	42.4%	48%	50%	49.4%	50.7%	55.2%
Recall	49%	33%	26.9%	23.7%	22%	18.5%	13.6%	12.9%

The statistics of precision and recall for part C) are shown in Table 3.3. This corpus contains 50 articles with 2-3 additional human-annotated topics. The total number of manually added topics is about 5.

Table 3.3: The corpus of 50 articles with additional human annotations.

Boundary	**1%**	**10%**	**20%**	**30%**	**40%**	**50%**	**70%**	**90%**
Precision	37.4%	47.4%	53.9%	55.8%	59.12%	59.4%	60.6%	64%
Recall	54.6%	35.9%	29.3%	23.7%	22.4%	18.8%	14.1%	13.5%

3.3.3 Discussion for keyphrase extraction

The proposed approach seems to be efficient enough to be comparable with other automatic keyword extraction systems. For example, the RAKE system achieved 33.7% precision with 41.5% recall and the undirected TextRank achieved 31.2% precision with 43.1% recall. Our approach achieved 37.4% precision and 54.6% recall for a small corpus with expanded number of annotations, including the problem of keyword generation and automatic clustering. We can assume that precision and recall will be a little bit lower for a bigger corpus. The most significant feature of the corpus is the number of exact manual annotations which are used for performance tests.

Automatic keywords will be used in next chapters for mapping the Linked Data topics to the articles and graph-based knowledge extraction.

3.4 Automatic tagging based on Linked Data

This section was previously published as [29] in 2010. In this section we describe unsupervised methods for automatic tagging based on information extracted from Linked Data. These methods are usable in situations where we have to tag unknown data and we have no corpus for learning methods.

We have created a web agent for collecting Call for Papers (CFP) announcements. Our web agent obtains CFP announcements from websites or from mailbox. The most important information is extracted and published on my own special website tmrg.cz in a user and machine readable way. One of the most important problems is event classification and clustering. Each event is marked with a group of tags which were literally found in the article's content. Other tags can be discovered from Linked data based on their relations.

Tagged data can have the form of short messages from RSS, short blog posts or emails. The automatic tags can be used for classifying the conferences. Users can use our web service to search for interesting events and sort them by their own preferences. We obtain tags with their relationship parameters and we can use them for automatic clustering of collected events.

3.4.1 Introduction to automatic tagging

Nowadays, there has been an explosion in user-created content on the Web. It can be blog posts, comments, announcements or just short messages on Twitter[17]. Social messages and statuses on Facebook[18] can be found or lost forever, because they are out of date. This information has no value for the common user, but it can be a valuable source of information for business intelligence. Social messages and posts can contain subjective opinions about new products or global services. It can save or earn a lot of money if the company owner is able to extract interesting information from these comments automatically. This company can change commercials, design or product images, that can increase popularity and public meaning.

For blog owners and users, categorization is very subjective and it is almost impossible to find the best solution which can be usable for all visitors and authors. Tagging is much more intuitive for the average user. This is the reason why many tagging systems are rising from the ashes. The user does not need to think, he just expresses his emotions and feelings about the content. He can tag for himself or for others, but the result is the same. New metadata can be used for searching, browsing and organizing. Even subjective information such as "*I like it*" or "*It is funny*" can be useable for other users. If somebody finds something funny, it can be funny for others as well.

[17] http://twitter.com
[18] http://www.facebook.com

Different types of tagging systems can be used for different types of content and applications. Collaborative tagging services such as Technorati[19] and Delicious[20] can be used for sharing tags and resources between users as a result of public knowledge. These services are valuable helpers, especially for bloggers.

3.4.2 Related work for automatic tagging.

Collaborative tagging systems allow users to create arbitrary tags for marking content with their ideas and opinions. Users can use tags from a shared repository or add their own. This is one reason why these systems cannot use fixed or controlled vocabularies. Golder et al. [48] described three main problems with collaborative tagging systems:

- polysemy,
- synonymy,
- different levels of specificity.

A polysemous word is one that has many related senses. For example, a "jaguar" may refer to an animal, or to a British luxury car manufacturer. Polysemy is a reason why we can get related but inapplicable items in full text searches.

Multiple words having the same or closely related meaning are referred to as synonymy. Synonymy represents one of the greatest problems in current tagging systems. Multiple words with the same meaning cause inconsistency in tagging systems. Inconsistent tagging leads to a lack of certainty that all relevant records have been found. Authors must rely on their expertise and intuition to choose the best possibility. Some problems can be solved by conventions. Some conventions can affect for example abbreviations. Generally known abbreviations should be used without comments, but others should not, for example, RDF or OWL. These words are well known in their abbreviated form and almost everybody uses them. If the abbreviation causes uncertainty, other tags should be used to eliminate the polysemy in the context, or we can add an explanation.

The next problem is that people use terms at different levels of specificity. This can range from very general to very specific. This is based on the author's expertise and requirements. Somebody describes an entity on the 'basic level', for example, as a 'car', somebody else as a 'jaguar' or a 'jaguar model xk'.

3.4.2.1 Systems for auto-tagging and clustering

Systems for auto-tagging are usually based on one of two principles: methods based on corpuses, which are used for learning, or methods based on statistics. The corpus-based methods compare the current document with others using TFIDF scoring. Statistical methods are usually based on extraction of the most frequent words.

[19] http://www.technorati.com
[20] https://delicious.com/

Data clustering is a popular technique for statistical analysis. This technique provides decomposition of datasets into subsets with similar features with regard to the current level of description. The clustering algorithm can be based on counting the number of co-occurring tags [49] (tags used for the same article). One important task is to set a cut-off point which will decide when the co-occurrence count is significant enough to be used. The obtained co-tags can represent linked nodes in the undirected graph. The main goal [49] is to divide the set of tags into non-intersecting groups of semantically-related tags.

Another approach [50] is to group documents that share the same tags into clusters. All documents in the cluster can be compared using TFIDF. The main idea is to compare documents with the same tag clouds instead of comparing a randomly chosen set of documents. In more focused clusters, this technique provides only unigram tags that literally appear in the blog posts.

Another approach [51] is based on tag suggestion. This system is called TagAssist. It is based on an annotated corpus and compares the document with others in the corpus using TFIDF. Automatic tag suggestions are based on tags from the documents with the best TFIDF score. This method enables automatic learning, but the corpus of similar annotated documents is needed.

3.4.2.2 Systems for manual tagging with Semantic Web techniques

Next, we would like to mention two projects based on publishing manual tagging results. The first one is called Int.ere.st [52]. Int.ere.st combines tags and the Semantic Web. Documents are tagged manually using SCOT (Social Semantic Cloud of Tags) ontology. Int.ere.st is also an open tagging platform with public API. This platform collects tagging information from different sources and publishes it in a machine readable way. The main problem with this approach is that we are dependent on the web service which gives us access to the tagging data.

Another similar approach is based on the MOAT [53] (Meaning Of A Tag) framework, which provides a collaborative way for authors to give correct meanings to their tags in a machine readable way [53]. The meaning of a tag is expressed by a triple in the ontology. Our tag will be a subject, we can use the predicate *moat:meaningURI* and the object will be identified as URI from the Linked Data. We use Php5 class implementation as the client for connection to the web service. This web service can offer different meanings and lets authors choose a URI from an existing set in a given tagging context [53].

3.4.3 Approach for automatic tagging

Our web agent specializes in computer conferences, so we started to crawl RDF pages about Computer science. This solution can be spread to other topics very easily. It depends just on the user's interests. Methods for automatic tagging have to be easy to implement and web service independent. The ideas have to be generally usable without special software or web services.

Every tag will be identified with the URI. The best option is to use some existing entities from Linked data, for example, entities from DBpedia[21], GeoNames[22] or FreeBase[23]. Each of these communities allows users to use web services or special API for direct communication with the knowledge base. If the global meaning is not usable, the URI will be generated for this tag automatically.

The relations between tags will be extracted from Linked data and stored as triples in the local database. The predicate will be some popular URI from a well-known vocabulary. We can build a hierarchical structure from the relations between the tags.

We can extract hidden information from the resources linked by our selected tags. It is possible to extract synonyms or a broader significance of our tag. This will solve the usual problems with tagging. Other information can be extracted from the cloud of automatic tags. It can specify the area of interest of the content.

3.4.3.1 Unsupervised automatic tagging without domain corpus

One of the main problems of automatic tagging is the fact that we do not know how to choose the best keywords for tagging. Other systems mostly use different types of clustering usually based on counting the TFIDF score between documents. The most important terms are usually extracted statistically. Unigrams [50] usually appear as the result of this effort. As unigrams are not the best terms for tagging, some systems try to count co-occurrence tags [49] [51] to enrich the basic term. These efforts are demanding, problematic and inefficient. But it is one way of finding a solution.

Our solution is to combine easy and fast techniques to reach the same target: annotated content with metadata in the form of tags. Automatic tags may not be completely correct; nevertheless, they will provide significant information about the topic of the article. This information can be used for automatic classification, for example. It is a common fact that we cannot rely on the correctness of every used tag if we want to derive new information.

We will start automatic tagging only with basic information: having many short articles about computer science, we want to divide them into an unknown number of classes with unknown intersections. We do not insist on independent classes. Common methods are usually based on clustering and the result is a hierarchical structure at best. This structure is based on statistical methods and does not give any valuable information about the domain in the real world. Our idea is the opposite of this solution. We want to take entities from the real world and map them to our articles. We can extract entities from Linked data because they contain disambiguated information about the entities and relations between them. We start from "Computer science" on DBpedia and extract other related entities to this topic. The entities are stored in the MySql database. Base on the [50] results, we started directly with a full-text

[21] http://www.dbpedia.com
[22] http://www.geonames.org
[23] http://www.freebase.com

search of the common form of the entity. The tag was assigned to the article when it was literally found in the content. This method is very effective even for large amounts of data:

This simple full-text search method was tested on 800 short articles about computer science (Call for papers emails) and 90% of them were tagged with at least 5 terms. These terms were directly found in the text and we did not use the relations between tags. We can append a parent entity if we find a few children entities, for example.

The average number of manual tags for article is about 2-3 in the public blogosphere. We were able to get 5 tags using the automatic way even without the use of relations between the tags.

3.4.3.2 Manual tagging with automatic expansion

Our method for automatic tagging does not have to be the best solution in all cases. We can use it as automatic expansion of the user tag, for example. We can take the manual tag and extract its synonyms or parents tag from Linked data. These tags can be used automatically or for the tag suggestion method only. It is possible to use them together with some other methods based on TFIDF.

3.4.4 Experimental design

Our methods were implemented as part of an experimental application for automatic classification of conferences based on their *Call for papers* emails. The reason behind automatic classification is to choose the best conferences for the user with regard to his preferences. We can simply show the automatic tag list to the user and we can let him choose the areas he is interested in. His preferences can be stored in the same way as the tags - in the form of URIs.

Figure 3.1 shows one of the automatic tags extracted from DBpedia. The title of the tag is in the form of a URI.

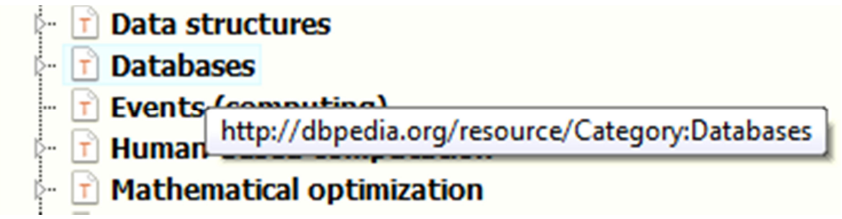

Figure 3.1: Description of the tag in the form of a URI.

Figure 3.2 shows the dynamic tree of automatic tags. The user is able to walk through the java script tree. The tag tree was implemented with JsTree[24] plug-in for jQuery[25]. This tree is loaded dynamically using Web 2.0 methods. Ajax from jQuery was specifically used for this purpose.

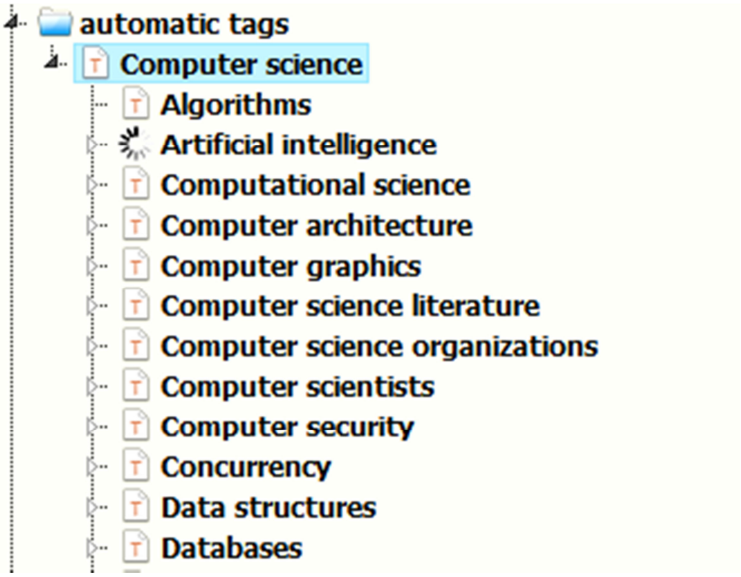

Figure 3.2: Hierarchical structure of the tags.

Figure 3.3 shows the possibilities of Web 2.0 techniques. There is a trigger on mouse right-click. This trigger shows the tag context menu. It can contain the options of moving, deleting, renaming, and adding to favourites, or it can open up a window with a description in a machine and user readable way.

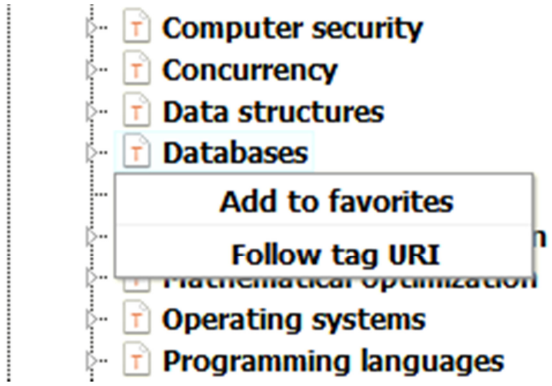

Figure 3.3: Tag context menu.

Users can add interesting tags to their personal favourites tag lists and use them for tagging and searching. Users do not need to create tags themselves. They can just pick them out from a global list. The second way they can get the URI for a tag is to use a public web service for searching entities. For example, the DBpedia lookup web service can suggest entities very

[24] http://www.jstree.com
[25] http://jquery.com

well. We can use Web 2.0 methods and we can use public web services with Ajax. The result is that the user can obtain a list of possible entities while writing the tag corresponding to the article.

Figure 3.4 shows the structure of manual tags. Each user can mark tags as favourites or he can use his own self-reference tags. For example: "*my conferences*", "*interesting conferences*", "*I like it*", "*not popular*", etc. We can choose some of these self-reference words and use them for the automatic counting value of a conference. For example, if five people mark the conference as "*interesting*", it could be interesting for others too.

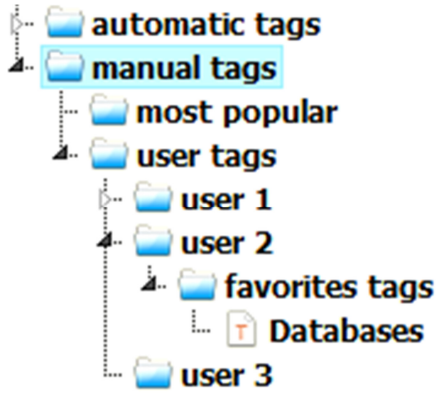

Figure 3.4: Structure of manual tags.

3.4.5 Discussion for tagging with Linked Data

This chapter has discussed a number of issues relevant to automatic tagging. We compared existing methods for automatic tagging and clustering and evaluated their benefits and disadvantages. The proposed methods for automatic tagging and automatic tag suggestions are based on Linked data and Web 2.0 techniques. These methods are unsupervised, domain independent and fully automatic without a learning phase based on a marked corpus.

These methods are very useful for large amounts of data and for time restricted computations. The results can be used for approximate classification into an unknown number of classification classes. The classification classes can be defined by the number of entities or by the number of levels of variations. We can use these methods for automatic classifying of manual tags based on level variation. The leaves of tag trees will be very specific and the local topic roots will be superficial. We can set the optimal 'basic level' of the description based on the statistical method used on the tree of tags.

Future work can be focused on other methods for extracting URI resources and for revealing hidden information in unknown groups of connected statements.

4 Clustering

This chapter will cover:

- Introduction to clustering,
- Definition of clustering algorithm,
- Feature selection with Linked Data and it`s evaluation with clustering algorithm,
- Cluster labeling with Linked Data.

The volume of available electronic documents is rapidly increasing, which leads to a requirement for their ingenious digital processing. One of the popular approaches to organizing textual data is the use of clustering algorithms, which divide documents into coherent clusters. The goal of clustering methods is to identify distinct groups in a dataset. The basic implementation of clustering puts together documents that share many terms. More generally, it puts together documents with similar features. The cluster hypothesis [3] says that documents in the same cluster behave similarly with respect to the relevance to information needs. The hypothesis states that if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant.

Clustering is primarily used in searching and browsing, which may be utilized either for internal processing documents, or even better for the presentation of results to the user. The main pioneers of semantic search using clustering include search engines Yippy[26] and Dogpile[27]. In the first case, it is already deployed in practice very good multilingual clustering for automatic processing of the Google, Yahoo! and Bing search results. This method is very convenient, because the user is not forced to go through hundreds of search results. He just limits search results to a cluster, which intrigued him.

The exploration of Internet resources is even more challenging than local text-document processing because of uncertainty as far as the quality of documents. Instead of making high quality web pages, some authors aim to make their pages rank highly by playing with the Web page features that search engine ranking algorithms are based on. This behavior is usually called search engine spam [4] [5]. Very often the unwanted pages have a higher ranking than the expected information sources. This is caused by the preparation of texts with regard to routine statistical analysis methods that should increase the position of an unwanted page in a search results. Search engine ranking systems were designed to prevent these search engine optimization (SEO) techniques.

The most common ranking algorithms are PageRank [6] and HITS [7]. Those algorithms are based on the theory that the most important pages on the Internet are those pages with

[26] Yippy search engine. http://yippy.com.
[27] Dogpile search engine. http://dogpile.com.

the most links leading to them. Links can be marked as votes. The importance of the page that contains the link and the number of outgoing links is also considered.

New kinds of spam aiming at links have appeared in the form of link farms. Building link farms is one technique that can deteriorate link-based ranking algorithms. Additional precautions have had to be taken by search engines in the form of identifying link farm spam pages [54].

4.1 Definition of clustering algorithm

The aim of the clustering algorithm is the automatic allocation of documents into smaller sets of similar documents called clusters. In the case of clustering algorithm, we can define the following input and output:

Input - The number of required clusters K and a set of documents with clear title. Unique identification of the document can be implemented using the ID, or simply by a document number.

Output - The output of the algorithm is a set of documents assigned to clusters. Each cluster is defined by a list of identifiers of documents that belong to it.

In the case of clustering algorithm, we can choose from the following approaches:

- Hard clustering - each document is assigned to exactly one cluster.
- Soft clustering - each document can be assigned to multiple clusters.

While most clustering algorithms use soft rather than hard clustering, we focus just on hard clustering, where it is possible to simplify the measurement of accuracy and completeness of the method.

According to the number of levels, we can distinguish these types of clustering algorithms:

- Flat clustering - all clusters are on the same level. Partitions are independent of each other. Number of required clusters is part of the input.
- Hierarchical clustering - clusters are combined to form a hierarchical structure automatically or according to the number of clusters. In most cases it is not necessary to specify the desired amount of clusters, since the algorithms are able to choose the number of clusters alone.

We will focus on flat clustering due to its easy, fast and reliable evaluation. In the case of hierarchical clustering evaluation some fundamental problems occur. One of the main issues is to say when the clustering is correct and where is a breaking point for the wrong result. Especially in the case of placement in a very similar cluster is not necessary to rule on the situation determining right or wrong classification, but the rate of correct classification. This fact is often neglected because there is no effective and unique solution [55].

The principle of clustering methods is called clustering hypothesis, which asserts that all documents within a cluster behave similarly and have similar characteristics with regard to the requested information. It means to contain the largest number of matching terms or topics.

4.2 Feature selection with Linked Data

In this section, previously published in Czech language [56], we would like to present our approach to feature selection with application of Linked Data. The importance of Linked Data is in links, so that a person or machine can explore the web of data. We used these techniques and resources for automatic feature selection and applied it to clustering method. We will demonstrate this approach with data collection related to IT area. These articles were automatically expanded by tags defined as resources from Linked Data. We will explain the transformation of these tags into features that can be used for clustering.

The issue of feature extraction to distinguish groups of documents is a very hot topic. It's the most common application includes various clustering methods. The aim of this chapter is to introduce methods that allow automatic feature selection using Linked Data. In order to evaluate the quality of feature selection, we apply it to the simplest and most popular clustering methods K-means.

4.2.1 Approach for feature selection with Linked Data

Currently there are a number of different methods for feature selection [8]. Some of them are correlation coefficients [8], mutual information [9] or SVM [10]. Now we introduce the technique for feature selection using Linked Data. The algorithm can be informally described as follows:

1. **Input** - *D* is a set of articles numbered from 1 to *N* and a set of nodes L from Linked Data, identified using the URI thematically covering articles. As an alternative, it is possible to use all the nodes from Linked Data as of DBpedia[28] covering most areas of interest. In the IT area there is currently about 20,000 nodes.

2. **Labels assignment** - the Articles are automatically enhanced by nodes from Linked Data in the form of labels. Assigning labels can be realized using full-text search of node name in combination with well-known methods such as tokenization, lemmatization, removal of stop-words, etc. The label is assigned to an article where the name of a label is found in the article's content at least once. Each label can contain data identifying its weight to the document. In the simplest case, it can be specified as the number of occurrences of the label name in the content of the article. The aim of this step is to estimate topics contained within articles.

[28] DBpedia. http://dbpedia.org.

3. **Analysis of the labels** - Labels (nodes from Linked Data) contain a wealth of interesting information that can be used for further analysis of the topics contained within article. The aim of this analysis is to determine the optimal features for clustering. The labels are directly considered as required characteristics. This analysis is performed on two different levels:

    - Local analysis - in this case, we will analyze a set of labels for one article, regardless of the labels assigned to other articles.

    - Global Analysis - handling labels according to their prevalence among articles.

4. **Output** - The result is the set of all labels contained in all the articles that best defines the articles with regard to others. These labels are directly used as features for clustering.

Next, we will explain the main principles of analysis labels from local and global perspective. In the case of local analysis, we investigate tags assigned to an article regardless of their frequency in all analyzed articles. Therefore we are dealing only the optimal choice of labels that would best describe the topics contained in the article. Standard techniques are:

- **Replacement label of its parent** - in this case, too specific label is replaced by a more general term. This substitution is particularly useful when the article is associated with a plurality of labels with the same parent. This substitution leads to a significant reduction of associated labels. Instead of labels with punctual meaning, a new label of great significance can be assigned to the article.

In the case of global analysis we examine all labels assigned to all articles. During the analysis, it is of course possible to use the basic techniques of local analysis, but we have to be careful with feature generalization. In most cases, it is appropriate to use the weighted characteristics that determine the significance of newly assigned labels. Now we briefly introduce the options and highlight the most important issues:

- **Replacement of synonyms** - in the case of synonyms we can choose the possibility that more frequently occurs in the articles. All other synonyms are replaced by this synonym.

- **Replacement label of its parent and vice versa** - in this case, it is convenient to find out whether some labels would not be appropriate to combine, divide, or vice versa, in order to achieve a reasonable level of description. In the first phase, it is good idea to replace label by its parent and determine whether the esulting set of articles that share this characteristic does not create a cluster approaching the expected size. If not, we should try to swap for a child to test whether the cluster will be created in the event of a combination of several labels. In this case we must take care that the child is or any of his followers in the text actually occurred.

- **Removal of unique labels** – unique labels are assigned very often to only one article. It is appropriate to remove these labels because there are insignificant for classification and clustering algorithms. These labels can occurs in the form of different remote topics the paper deals with, but those are missing in all the other articles or not adequately described in the context of Linked Data.

4.2.2 Clustering evaluation

In the previous chapter we introduced the technique for feature selection, and now we look at the problems of its evaluation. There are two basic options:

- **Direct comparison of the two sets of features** - a set of automatically selected features is compared with set of manually selected features. In other words, we try to automatically select the same features characterizing the articles that would a man choose.

- **Comparison of the results of an algorithm** – we will compare two sets of outputs of a clustering algorithm. Automatically selected features will be used as input for the first test-case and hand-selected features will be used as input for the second clustering test-case. The results of the clustering algorithms will be compared.

In a case of direct comparison of the two sets of properties, we cannot consider the evaluation as accurate and reliable, which is mainly due to the subjective selection of features in case of manual selection. One may omit some major features and vice versa add another less relevant one. The actual comparison of the two sets is indeed very simple, but very inaccurate. You can only test incidence and absence of the term and not the measure of proximity between manually and automatic selected property.

When comparing the results of the algorithm with two sets of features, we can examine its influence on the resulting data. We do not matter to the specific features, but rather on how they affect the same algorithm such as clustering.

We chose the second option, as it shows that manual selection of properties may not always be appropriate and may not achieve the best possible results. In addition, the manual selection of properties encumbered subjective approach. It has, in the case of clustering, significantly less effect and the result is thus better comparable.

As a basic clustering method the K -means algorithm was used. It enables the distribution of documents into a specified number of clusters. We will use three basic techniques for feature selection:

1. Manual feature selection - features are selected manually in a way, that most distinguish individual sample set of documents. The choice of properties is carried out manually by the user base of the knowledge of articles and the names of manual clusters, by which we evaluate clustering. The user is sufficiently acquainted with the clustering area.

2. Feature selection using statistical methods – we can use the simplest statistical approach in the form of TFIDF. In this case, we want to demonstrate that the use of statistical methods is dependent on the number of training documents. In the case of a small number of documents returned as a search result shows this method to be inappropriate and in addition very computationally intensive. In order to use this technique we had to increase the number of training documents to 500. The set of 25 training documents was completely unusable for TFIDF.

3. Feature selection using Linked Data - our method for feature selection using labels obtained from Linked Data.

For evaluation we used a set of 10 times 25 manually annotated articles. The clustering algorithm should correctly divide them into five different clusters. Evaluation of clustering was based on comparison with manual division of the documents into clusters. For the sake of credibility the clustering algorithm was performed ten times. Due to random component used for centroid initialization of K-means that is used for initial setup of centroids. This initialization significantly affects the quality of the clustering algorithm.

The following Table 4.1 shows the results of clustering using the manually selected properties. Meaning of the columns is following:

- Precision – standard calculation of precision.
- Recall – standard enumeration of recall.
- Match – number of articles with direct match according to manually selelected clusters.
- Extra – number of articles that were contained in automatic cluster and that do not belong there.
- Missing – number of articles that were missing in automatic clusters and that were included in manual clusters.

Table 4.1: Clustering with manual features

Test	**Precision**	**Recall**	**Match**	**Extra**	**Missing**
1	0.64	0.64	16	9	9
2	0.76	0.76	19	6	6
3	0.457	0.64	16	19	9
4	0.48	0.48	12	13	13
5	0.4	0.4	10	15	15
6	0.321	0.36	9	19	16
7	0.351	0.52	13	24	12
8	0.645	0.8	20	11	5
9	0.613	0.76	19	12	6
10	0.56	0.56	14	11	11
overall:			**148**	**139**	**102**

The following Table 4.2 shows the results of clustering using statistically selected properties. The meaning of the columns is the same as in Table 4.1.

Table 4.2: Clustering with statistically selected features (TFIDF)

Test	**Precision**	**Recall**	**Match**	**Extra**	**Missing**
1	0.4	0.4	10	15	15
2	0.52	0.52	13	12	12
3	0.64	0.64	16	9	9
4	0.32	0.32	8	17	17
5	0.48	0.48	12	13	13
6	0.56	0.56	14	11	11
7	0.5	0.6	15	15	10
8	0.423	0.44	11	15	14
9	0.4	0.4	10	15	15
10	0.464	0.52	13	15	12
overall:			**122**	**137**	**128**

The following Table 4.3 shows the results of clustering using properties obtained from our algorithm using information from Linked Data. Meaning of columns is identical to the previous tables.

Table 4.3: Clustering with features obtained from Linked Data

Test	**Precision**	**Recall**	**Match**	**Extra**	**Missing**
1	0.375	0.36	9	15	16
2	0.4	0.48	12	18	13
3	0.583	0.56	14	10	11

4	0.5	0.48	12	12	13
5	0.542	0.52	13	11	12
6	0.375	0.6	15	25	10
7	0.436	0.68	17	22	8
8	0.583	0.56	14	10	11
9	0.469	0.6	15	17	10
10	0.472	0.68	17	19	8
overall:			**138**	**159**	**112**

Now we compare the overall results achieved for all three types of clustering. The following Table 4.4 shows that the best choice of features is of course handmade, which was expected. Our method for selecting the features is, however, only about 8% worse if we use F-measure. Statistical selection of properties using TFIDF achieved very good results, but only through the use of two important modifications:

- TFIDF were trained on 20x greater set of articles, as in the training set to the same size, this technique proved to be unusable.
- Resulting labels chosen by TFIDF were adjusted in a way that was dropped 5% of tags with the highest TFIDF score, as they were very unique and were not statistically significant in the context of clustering. In Addition, a number of properties obtained using TFIDF was limited to the top 30%, while the maximum number of selected features was limited to 40 pieces.

Table 4.4: Evaluation of clustering with different feature selection methods

Feature selection:	Manual	TFIDF	Linked Data
Precision:	0,516	0,471	0,465
Recall:	0,592	0,488	0,552
F-measure:	0,551	0,479	0,505

4.2.3 Suggestions for future work with feature selection

Further work in this area can be focused on the evaluation of this technique of feature selection using clustering to larger collection of documents. For this purpose it is necessary to use a collection of documents from IT area containing the information for clustering evaluation. This set was not available in the time of evaluation of this approach. Based on the inspiration of article [10], it would be nice to verify this technique with the Open Directory Project[29], which contains a hierarchical structure of categories including classified articles. According to [10] it is also possible to use Delicious service [30] to get these articles enriched for labels in the form of short text strings, which might be more interesting and useful information. For reasons of comparability with [10] it would be good to use the same transformation of hierarchical structure of categories from ODP to flat and hard categories.

The next chapters will cover methods for soft clustering directly using the Linked Data. This algorithm, which is directly based on information from Linked Data suggests the optimal number of clusters and articles divided into groups according to the connections between topics. With this idea we will have to deal with very large graphs.

4.2.4 Discussion for feature selection with Linked Data

In this section we present a technique that allows automatic feature selection using machine readable information from Linked Data. We have described our own selection algorithm of properties and approached the main problems of evaluating the features quality.

As a suitable approach to evaluate feature selection it appears to be clustering algorithm that uses these properties. Therefore we do not rate directly acquired feature, but rather their importance for further processing, which is much more meaningful value.

[29] Open Directory Project. http://dmoz.org.
[30] Delicious. http://delicious.com.

This technique of feature selection using Linked Data reached in this case, only 8% poorer results when comparing the F-measure to the manual selection of features.

4.3 Cluster labeling with Linked Data

In this section, previously published in [28], we would like to introduce our approach to cluster labeling with Linked Data. Clustering web pages into semantically related groups promises better performance in searching the Web. Nowadays, only special semantic search engines provide clustering of results. Other engines are doubtful as far as the quality of clusters and moreover a dependable system for labeling these clusters is lacking.

Linked Data principles enable to connect data from different sources and to query them over the Internet. The information from Linked Data can be used for preliminary estimates of topics covered by a set of documents. These topics are represented as resources from Linked Data and they are used for smooth human-readable labeling of clusters.

4.3.1 Introduction to cluster labeling

Search engines are able to find related documents with regards to the content and quality of web pages, but the main problem remains untouched. It is hidden in the problem of a user's query. It is not difficult to imagine a situation in which it is hard, if not impossible, to formulate a query precisely. It is impossible to find related documents, if the topic of a requested document is not very common and the user is not familiar with the vocabulary appropriate for describing a topic of interest. Clustering can be the answer to this demand. There are many applications of clustering:

- Search result clustering,
- Scatter-Gather,
- Collection clustering,
- Cluster-based retrieval.

The principle behind search result clustering is to divide results into distinct clusters, so the user is able to choose a more specific subset of documents for listing. The search query is matched against clusters and the content of the best scoring clusters is returned as a result. This content can be divided into other clusters and the user can choose the appropriate cluster again. This approach is part of the Dogpile [57] and Yippy [58] search engines. The user can walk through clusters and use them as a filter for search results. This application is shown in Figure 4.1.

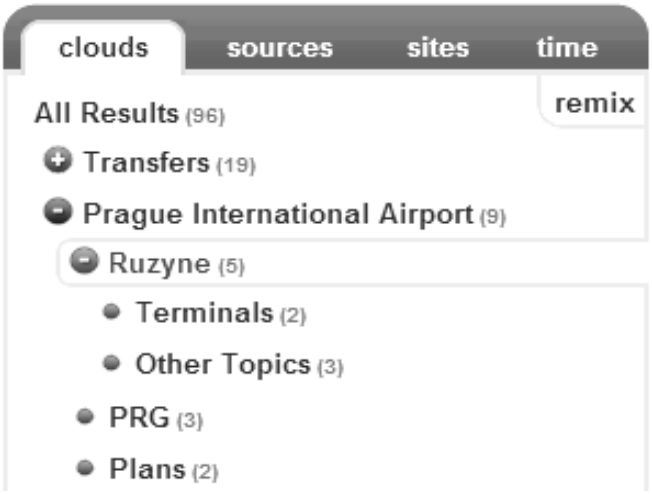

Figure 4.1: Yippy – clusters with labels

Scatter-Gather [26] is a technique for document browsing that employs document clustering as its primary operation for browsing large document collections. It is based on interactivity with the user. Initially, the system scatters the collection into a small number of clusters and presents them to the user. The user selects one or more of the groups for further browsing. The selected groups are gathered together to form a sub-collection and clustered again. Selection of the related clusters can be done based on the summaries or cluster labels.

Collection clustering is focused on dividing the documents in a set into smaller subsets for better performance of further information retrieval.

Cluster-based retrieval is based on the idea that a collection of documents can be divided into smaller subsets and a search can be done only by matching the query to one or more clusters. For example, when a query is set, the best document is found with common techniques and other related documents are chosen from the same cluster as the first one.

Fundamental clustering methods are based on the number of shared terms and other statistic-based approaches. Web 2.0 brings some additional possibilities on how we can improve clustering methods. Tags can be used as an additional feature for clustering methods [49] [50]. In [59] an approach was proposed where tags significantly increase the F-measure (1) for K-means from 0.139 to 0.225 in a test dataset built from ODP[31].

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{4.1}$$

Tags can increase the F-measure for the latent Dirichlet allocation (LDA) clustering algorithm from 0.260 to 0.307 in the same dataset. Many tagging services [60] [61] have been introduced with promising results. There are systems for assisting in tag selection [51] that help to solve some problems like synonyms and different levels of specificity. Tags can be

[31] http://dmoz.org/

used as keywords but rather in the form of Linked Data resources [29] with additional semantic information.

4.3.2 Previous work to cluster labeling

Nowadays, there exist four basic approaches to cluster labeling:

- Differential cluster labeling,
- cluster internal labeling,
- combination of inter-cluster and intra-cluster labeling,
- label extraction from taxonomies or corporas.

Differential cluster labeling compares clusters with each other and chooses terms or keywords that maximally distinguish the individual clusters. Common methods for feature selection can be used for this approach. Popular is mutual information, X^2-test or information gain (IG).

The output of this labeling approach is a set of terms or keywords for each cluster. However, a list of significant keywords will many times fail to provide a meaningful readable label for a set of documents. In many cases, the suggested terms tend to represent different aspects of the topic underlying the cluster. In other cases, a good label may not occur directly in the text and it is not possible to extract it. User intervention can be required to choose a proper label from the suggested terms to successfully describe the cluster's topic.

Cluster internal labeling computes a label that depends on the cluster itself, not on other clusters. The common method is to label a cluster with the title of the document closest to the centroid based on the cosine similarity. Titles of documents are easier to read than a list of keywords. The title of a document can contain an important context or a topic that was not mentioned in the text directly or that was not chosen by statistical methods.

Another approach [25] uses a combination of intra-cluster and inter-cluster term extraction, based on a modified version of the information gain measure. This approach tries to capture the most significant and discriminative words for each cluster.

Other work investigates the contribution of external knowledge bases for cluster labeling. In [62] Wikipedia is used to enhance the quality of cluster labeling. A general framework for cluster labeling extracts candidate labels from Wikipedia in addition to important terms that are extracted directly from the text. The labeling quality of each candidate is then evaluated by several independent judges and the best evaluated candidates are recommended for labeling.

4.3.3 Approach to cluster labeling with Linked Data

In this section, we would like to discuss our approach to internal cluster labeling with Linked Data. First, we will explain the basic principles behind the Linked Data application, than we

will look at a graph expansion and scoring illustration. The labeling algorithm will be presented by a pseudo code with additional comments.

4.3.3.1 Application of Linked Data

Linked Data contains information about a resource and moreover links to other related resources. The resources are applied as tags to documents. There are two basic types of links that we can directly use:

- Parent-child relation,
- links to synonyms.

These connections are bidirectional so a child can find his parent and a parent can find his children. Relations are described by ontology predicates. For example: "*dbpedia-owl:genre*", "*skos:broader*", "*dcterms:subject*". The meaning of these predicates differs slightly, but we can use it in the same way. An example of these relations between resources is shown in Figure 4.2.

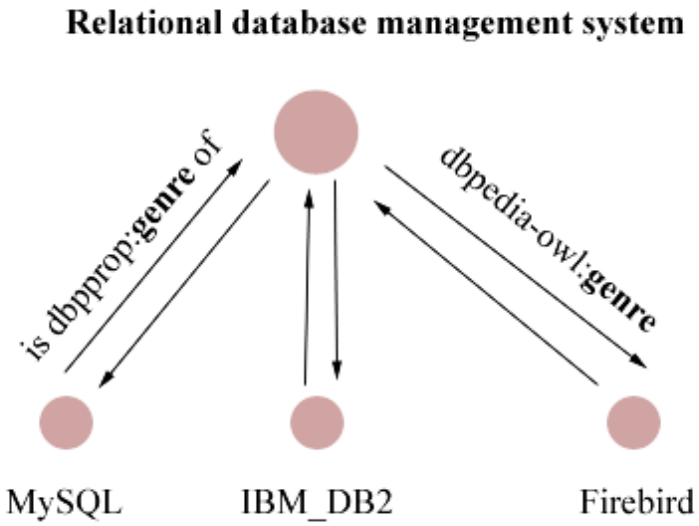

Figure 4.2: Scheme of hierarchical relations between nodes in LD

Synonyms are designated by the ontology relation: "*owl:sameAs*", which indicates true synonyms, and the relation "*skos:related*", which indicates related concepts.

4.3.3.2 Graph expansion and scoring

Each tag *t* is rated by a score. This score consists of two parts: the internal score and the gathered score (see equation 4.2):

$$t_i^{SC} = t_i^{SCI} + t_i^{SCG} \quad (4.2)$$

First, we have to prepare and expand a graph of tags. Then, we set the tags' internal score in the preparatory phase. The tags' gathered score will be computed in the calculation phase. Graph expansion is implemented using the following procedure:

1. Tags are assigned to documents of a cluster.
2. The tags' internal score for each cluster is computed based on the number of assigned documents.
3. Relationships from Linked Data are used for graph expansion. Parents are added automatically, children are added in case they occur in the content of the documents. The internal score for those tags is computed as explained above.
4. Synonyms are replaced by one representative. This action removes cycles and a spanning tree is created.
5. The choice of the best representing tag that will be used for labeling is done in the following algorithm. In case more than just one tag is required, the terminate condition of *removeMinNodes* can be updated.

The gathered score of a tag will be determined by using equation 4.3:

$$t_i^{SCG} = \sum_{j=1}^{count(t_i^{NB})} \frac{t_j^{SC}}{t_j^{OE}} \quad (4.3)$$

Where *count(t^{NB})* is the number of neighbors of the tag, t_i^{SC} is the score of a tag and t_i^{OE} is the number of output edges from the tag.

The tag's internal score t_i^{SCI} can be computed in many different ways based on the statistical approach. The easiest way is to set the value of a tag's internal score as the number of documents that this tag is assigned.

4.3.3.3 Description of labeling method

Input:

- T = set of tags (synonyms are grouped)
- E = set of edges (used by tags)
- TS = set of scores (for tags)

Initialization:

```
For each tag t in T do
    t.score = TS[t.id];        // tag is an object
```

Algorithm:

```
Function GetBestNode
{
	For step from 1 to N do // linear complexity
	{
		norm = 0;

		For all tag t in T do
		For all tag.tn in t.incommingNeighbours do
		{
			t.score += tn.score / tn.numberOutEdges;
			norm += square(t.score);
			norm = sqrt(norm);
		}

		// normalization
		For all tag t in T do
			t.score = t.score / norm;
	} // End from 1 to N

	/* remove all nodes without input edges
	if there is more than one node */
	T.removeMinNodes(); // there are no cycles
} // End function GetBestNode
```

Output:

The tag (node) with the highest score will be used for labeling:

$$t_{best} = \arg max_k \; t_k^{SC} \tag{4.4}$$

Each tag contains a huge amount of semantic information about its name in different languages, a description in different languages and links to other related resources with additional semantic relations. If we only need a short label, we can use the title of a tag in an appropriate language. If we need the semantic meaning of a tag, we can use its description in machine or human readable form.

4.3.4 Evaluation of cluster labeling

A set of experiments was conducted to evaluate the correctness and reliability of our approach to the label selection problem. The data and evaluation measures will be described first, followed by descriptions of the experiments and their results.

4.3.4.1 Data collections and evaluation measures

Three different data collections were used as a source of documents for clustering and labeling:

- Conferences – our collection consists of 15 000 calls for papers.
- News Groups – specifically 20 News Groups dataset [63].
- ODP – Open Directory Project [64], which is often use for the evaluation of text-mining problems.

Evaluation of the cluster labeling method is a difficult problem, for which no established methodology has gained wide acceptance. We have set up user evaluation of the cluster labeling as follows:

- Direct match (DM).
- Correct is close (CIC) – correct tag is at a distance of under two.
- HJ correct – human judge claims that label is completely correct.
- HJ wrong – human judge claims that label is completely wrong.
- HJ acceptable – human judge is able to accept this label as a relatively correct description of the document subset.

4.3.4.2 Experimental setup and results

The experiment consists of these steps:

1. Use one of the document sources: Conferences, News groups or ODP.
2. The human judge will create 10 clusters with the same number of documents. In our case it was 10 documents. This human will choose the best node from Linked Data for labeling these clusters.
3. Script takes those documents and applies one of the common clustering algorithms. K-means was used in our case.
4. Our labeling algorithm is applied to the clusters and the results are presented in Table 4.5.

Table 4.5: Evaluation of our cluster labeling algorithm

	Conferences	20 News Groups	ODP
DM	5	1	2
CIC	2	1	0
HJ correct	6	4	3
HJ wrong	2	3	3
HJ accept.	8	5	6

4.3.4.3 Discussion for cluster labeling

Our approach to cluster labeling is very intuitive and easy to use. Linked Data has great potential as an external source of free knowledge and promises better results with the growing number of electronic documents with free and open access. The information from Linked Data can be used directly in the form of labels or indirectly as ontology for information extraction. The labels from Linked Data are short and precise, which enables better utilization and understanding.

Further research in this area can be focused on differential labeling with maximum effort placed on cluster distinction. It is important to find a good measure for label distinction because the distance between tags in a graph may not be sufficient enough.

Another area for further research can be focused on existing graph algorithms such as PageRank and HITS.

5 Classification

In this chapter, previously published in [65], I would like to present a new approach to classification using Linked Data and PageRank. My research is focused on classification methods that are enhanced by semantic information.

The semantic information can be obtained from ontology or from Linked Data. DBpedia was used as a source of Linked Data in our case. The feature selection method is semantically based so features can be recognized by non-professional users as they are in a human readable and understandable form.

PageRank is used during the feature selection and generation phase for the expansion of basic features into more general representatives. This means that feature selection and PageRank processing is based on network relations obtained from Linked Data. The discovered features can be used by standard classification algorithms. We will present promising results that show the simple applicability of this approach to two different datasets.

5.1 Introduction to document classification with Linked Data and PageRank

Document classification is an important part of document management systems and other text processing services. Today`s methods are usually statistically oriented, which means a large amount of data is required for the training phase of these classification algorithms. The preparation of sufficient classification training sets and proper feature selection methods is a challenging and time consuming task even for domain specialists. So the common solution to this problem is based on relatively comprehensive corpuses that contain a lot of documents divided into different classification classes. Statistical methods try to discover relations between terms and classification classes during the training phase.

Our approach recognizes interesting keywords and expands them using semantic information obtained from Linked Data. For example, based on a feature *MySql* we can do the feature expansion into databases without the explicit occurrence of this word in the document content. The classification phase can process these parent concepts and use them for the correct pairing of documents and classification classes.

5.2 Previous work

Document classification can be defined as content-based assignment of one or more predefined categories (classification classes) to documents. This method can be used for document filtering to topic-specific processing such as labeling, keywords extraction and automatic translation. Common applications are e.g. spam filtering, news processing,

vulgarities detection and child protection on the Internet. We can distinguish two phases in document classification processing:

- learning phase,
- classification phase.

In the learning phase, the user defines categories by giving training documents for each of the interested categories. Methods for document classification require sample documents that belong to the appropriate classification class. The quality of the classification process usually grows with an increasing number of training documents. This is a weakness of document classification, because a solid training collection is required.

In the classification phase, previously unseen documents are given to the classifier, which returns a class association. The result of this phase is one or a set of documents with scores or probabilities for the categories.

There have been many supervised learning techniques for document classification. Some of these techniques include Naive Bayes, k-nearest neighbor, vector approaches e.g. Rochio, support vector machines, boosting [14], rule learning algorithms [15], Maximum Entropy and Latent semantic analysis.

DBPedia was used as a source of Linked Data presented in this chapter. We use a local copy of Linked Data stored in our relation database for performance purposes, but SPARQL endpoint could also be used. DBPedia is semantically enriched Wikipedia that was successfully employed previously for computing semantic relatedness of documents. WikiRelate! [66] combines path based measures, information content based measures, and text overlap based measures. Explicit Semantic Analysis [67] uses machine learning techniques to explicitly represent the meaning of a text as a weighted vector of Wikipedia-based concepts.

Another approach to document classification [18] proposed term graph model as improved version of the vector space model. The aim of this model is to represent the content of a document with relationship between keywords. This model enables to define similarity functions and PageRank-style algorithm. The vectors of PageRank score values were created for each document. The rank correlation and the term distance were used as a similarity measures to assign document to classification class. An alternative approach to document classification uses hypernyms and other directly related concepts [19] [20]. Next step in document classification can be marked as feature expansion with additional semantic information from ontology [21]. This approach [21] is exploiting the external knowledge for mapping terms to regions of concepts. For exploration of related concepts, the traversal graph algorithm is used.

5.2.1 PageRank

The PageRank algorithm was developed in 1998 by Page and Brine [41] as approach to web pages ranking through exploring hyperlinks structure on the Internet. The importance of each web page depends on the number and PageRank value of all web pages that link to it. This approach also known as "Random surfer walking" has been studied and improved for citation analysis [68].

Our modified version of PageRank 5.1 corresponds to its matrix definition [69] where:

- *Px(a)* is a value of node *a* in iteration *x*,
- *d* is a damping factor usually set to 0.85,
- *V* is a set of all nodes in the graph,
- *U* is a set of nodes with link to node *a*,
- *D* is a set of all dangling nodes,
- w_{ij} is weight of link from node *i* to node *j*.

$$P_{x+1}(a) = \frac{(1-d)}{|V|} + d * ((\sum_{u \in U} \frac{P_x(u) * w_{ua}}{\sum_{v \in V} w_{uv}}) + \frac{\sum_{s \in D} P_x(s)}{|V|}) \quad (5.1)$$

All modifications are primary focused on initialization step and they will be discussed later in this section.

5.3 Feature selection for document classification

Feature selection is the most important part of our approach. The whole approach for document classification consists of the following steps:

1. *Training* – basic features are selected and used for the definition of each category;
2. *Features processing* with Linked Data and PageRank;
3. *Classification* – subset from 20 News groups collection [63] was used;
4. *Evaluation* – it will be shortly described in the next section.

Now we will describe the method for feature selection and their processing with Linked Data and PageRank. Basic features are selected from documents in a common way. We can use TFIDF, X^2 or any other method explained before in the chapter Keywords extraction. These features are mapped to nodes in Linked Data, so each feature can be identified with URI and it has clear position in a graph of Linked Data. Other features are discarded as insignificant waste. The mapping is based on the full or partial compliance between feature (term) and name of the node in a corresponding language. The nodes are usually labeled in several languages with regard to the significance and popularity of a node.

The next step consists in processing of these nodes with PageRank and other related nodes from Linked Data. The objective of this effort is to get related nodes with high relevance to the document, even if they were not seen in the content of the document explicitly. Graph expansion is illustrated on the Figure 5.1. The original (basic) nodes are marked as *I* and derived nodes are marked as *II*.

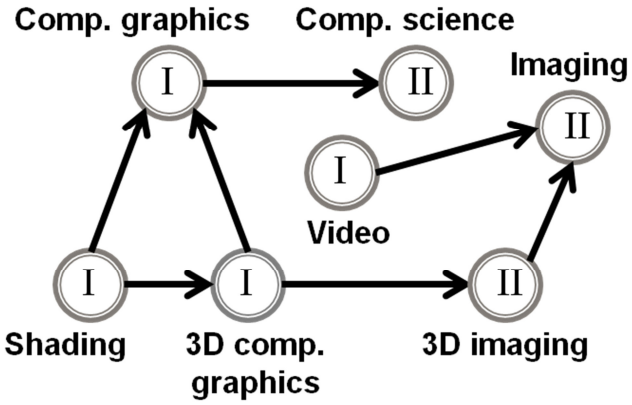

Figure 5.1: Graph expansion with PageRank

We have investigated three options for initialization of the PageRank algorithm (see Figure 5.2), where the steps of node expansion in a graph are marked with *I*, *II* and *III*:

a) All edges are assigned the same weight equal to 1.

b) The basic nodes are advantaged with self-citation edge with increased weight.

c) The basic nodes are advantaged with self-citation edge. The edges to the new nodes are penalized base on the quadratic distance of path from the basic node.

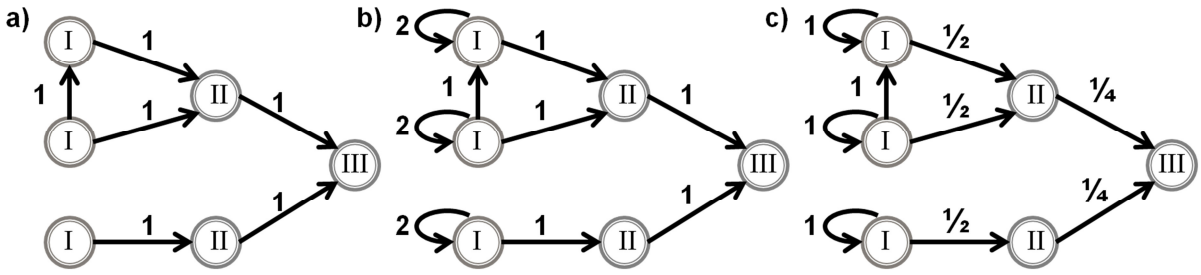

Figure 5.2: Initialization of the PageRank

Table 5.1: Dependency of PageRank score on nodes expansion and weights of the edges

Step exp.	Node	var a)			var b)		var c)
		I + II	I + II + III	I + II + III + IV	I + II	I + II + III	I + II
I	1	0.08	0.07	0.027	0.16	0.14	0.12
I	2	0.18	0.15	0.059	0.20	**0.18**	**0.22**
I	3	0.13	0.11	0.043	0.14	0.13	0.17
II	4	0.13	0.11	0.043	0.10	0.09	0.10
II	5	0.27	0.18	0.071	**0.21**	0.15	0.21
II	6	0.11	0.09	0.036	0.09	0.08	0.09
II	7	**0.11**	0.09	0.036	0.09	0.08	0.09
III	8		**0.19**	**0.355**		0.16	
IV	9			0.329			

As expected, our evaluation of the expanded graph in Figure 5.3 shows, that the variant *c)* achieves the best results (see Tab. 1) due to effective limitation in expansion of the basic nodes. The other versions requires explicit limiting criterion for graph expansion. In the next section, this variant was used for evaluation of our feature selection approach.

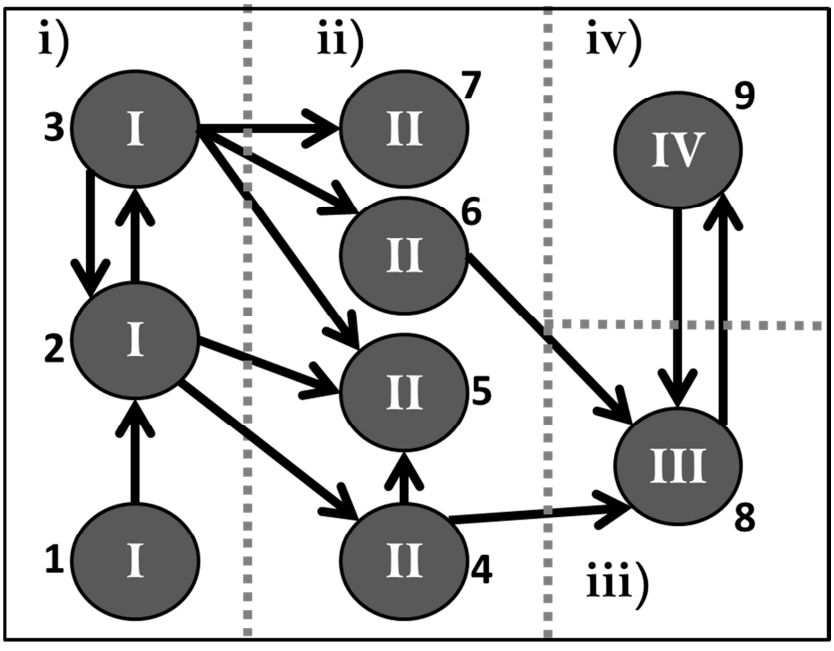

Figure 5.3: Graph expansion with PageRank

5.4 Evaluation of document classification with Linked Data and PageRank

Two different data collections were used for evaluation purposes:

- Conferences,
- 20 News Groups dataset.

Our collection of conferences consists of 18 000 calls for papers. Conferences can be relatively easily divided into single and multi-topic cases. Multi-topic conferences (approx. 1300) were discarded so we were able to apply single label document classification. Therefore this collection was chosen as a basic evaluation tool.

The remaining conferences were assigned to the corresponding topics based on the lists found on the Internet. Almost every document from this collection consists of a large number of keywords related to the category. Therefore, we were able to train categories relatively quickly. Approximately 10 solid documents were enough to train 1 category with almost constant macro-averaging F1(β=0.9). This F1 measure was almost constant from 10 to 100 solid training documents for 1 category. After 100 training documents or 2000 features, a problem with over-training was noticed.

For evaluation purposes the subset of 20 News groups collection was used. The reason was that our source of Linked Data was not sufficient to distinguish between similar categories in 20 News groups collection like *comp.os.ms-windows.misc* and *comp.windows*. Another problem was overtraining that occurs with approximately 100 training documents for one category.

For evaluation purposes we decided to compare our features selection method and the standard statistical approach with the same vector space classification algorithm (Rocchio). The comparison (see Table 5.1) is done with macro-averaging F_1 measure. The number of testing documents is determined as 20% of training documents.

5.5 Recommendations for further research

Our method for document classification with Linked Data is promising especially in those tasks with insufficient training sets or for quick filtering of existing documents. In those cases, the training phase could be very expensive and a waste of time for the user. Our method allows the definition of assigning categories using only a small number of training documents or a single node from Linked Data with automatic expansion on both sites - category definition and feature selection. In common tasks of document classification, the existence of keywords from interesting categories in Linked Data is required.

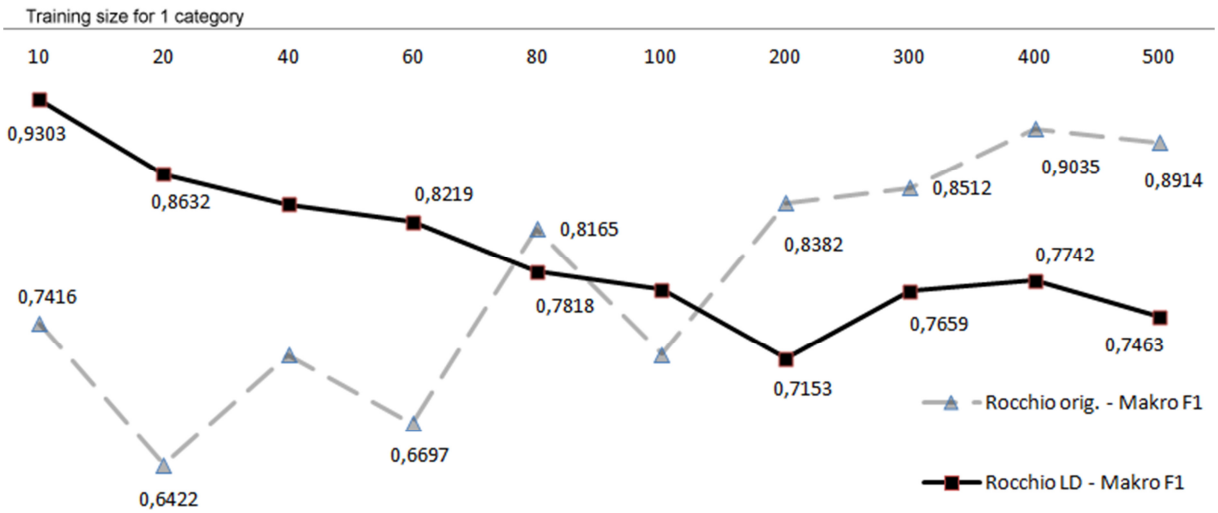

Figure 5.4: Graph – F1 measure for Rocchio classification algorithm

One of the biggest problems is to eliminate the over-training. Another possibility is to create a method for document classification directly based on the graph analysis reconstructed from Linked Data resources and their relations.

6 Semantic analysis of software specifications with Linked Data

This chapter was previously published in [30]. This chapter proposes application of our techniques, introduced in previous chapters, on the complex analytic task and demonstrate the benefits brought by combination of Linked Data with text-mining.

Software development life cycle is the process involved in the design, development and improvement of a software application. Nowadays especially component systems are used due to possibility of implementing reusable independent modules. The individual module, known as a software component, can be implemented in a form of a software package, a web service or a web resource that encapsulates a set of related functions. Software products are derived in a configuration process by composing different components. Moreover a software product line enables stakeholders to derive a different software products based on their needs. This fact and need for validation of the software product and its components requires methods for software specification processing and matching with concrete sw properties. In this chapter we will propose an approach for semantic analysis of software specifications with Linked Data.

6.1 Introduction to software specifications

Essentially software is determined by its functional and non-functional characteristics [24]. Functional characteristics describe software behavior that can be directly implemented and evaluated by common programmer. Furthermore, software specification documents are usually focused on functional characteristics, because it is easy to understand and describe application behavior to the customer. Unfortunately, there has been a lop-sided emphasis in the functionality of the software, even though the functionality is not usable without the necessary non-functional requirements.

These requirements are usually hidden and they occur at the time when it is least suitable. It includes usability, interoperability, flexibility, performance and software security. Real-world problems are more non-functionally oriented than they are functionally oriented, so even great application could be unusable if it disregards the typical use. These problems include high cost and slow processing, poor productivity, lower profit and it leads to unhappy customer.

In this chapter we will discuss our approach for semantic analysis of software specification using Linked Data (LD). The Web of Data is often illustrated as a fast growing cloud of interconnected datasets representing information about barely everything [70]. Linked Data refers to the Web of Data in contrast to the Web of Documents. Linked Data extends the current web that consists of documents and the computer-meaningless links between

documents. In the case of Linked Data, not just documents but also data elements (things) and the links between these data elements exist. More over the links in LD are expressive, unambiguous and identified by URI. LD is therefore more structured and machine process able than Web of Documents [71]. An example of these relations between resources is shown in Figure 6.1.

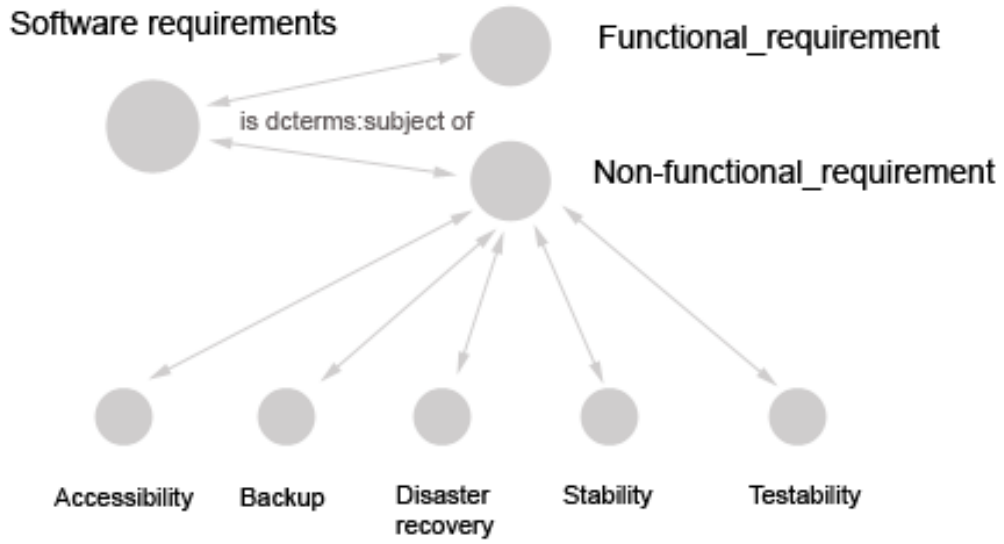

Figure 6.1: Example of hierarchical relations between nodes in LD

Relations from Figure 6.1 are bidirectional so child node can access its parent node and vice versa. There is a relation with predicate “dcterms:subject of” on one site and predicate “is dcterms: subject of” on the other site. Part “dcterms:” is a namespace identified by URI in the declaration part of a resource. There are other semantic links between siblings, so the potential node expansion can be processed through siblings instead of direct parent node. Example of data from “Non-functional requirement“ in plaintext format:

Non-functional requirement (URI: http://dbpedia.org/page/Non-functional_requirement)

- dbpedia-owl:abstract: In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture …
- dcterms:subject:
    - category:Systems_engineering
    - category:Software_requirements (link to parent)
- rdf:type:
    - yago:Requirement105892651

    - yago:SoftwareRequirements (link to parent)
- rdfs:comment: In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that …
- owl:sameAs:
    - fbase:Non-functional requirement (http://rdf.freebase.com/ns/m.08pc1k)
    - http://wikidata.dbpedia.org/resource/Q3254666
    - http://pt.dbpedia.org/resource/Requisito_não-funcional
    - http://yago-knowledge.org/resource/Non-functional_requirement
- is dcterms:subject of (links to children):
    - Accessibility
    - Audit and control
    - Availability (see service level agreement)
    - Backup
    - Capacity, current and forecast
    - Certification
    - Compliance
    - Configuration management
    - Dependency on other parties
    - Deployment
    - Documentation
    - Disaster recovery
    - …

6.2 Related work

In this section we will introduce Requirements engineering (RE) and Linked Data applications. We will propose our approach to text analysis using Linked Data thus we need to explain the basics about Linked Data and related web services such as DBpedia Spotlight [33]. This web service is used for semantic enrichment of a text and it consists of methods for entity detection, candidate selection and disambiguation based on related concepts extracted from Linked Data.

6.2.1 Requirements engineering

Requirements engineering (RE) is the process of developing requirements through an iterative cooperative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained [72]. Requirements engineering identify the purpose of a software system, and documents it in a form that is suitable to analysis, communication and subsequent implementation [73]. If

we want to build a software system it has to be described in some way before the design and implementation process will be started. Typically, these descriptions, known as requirement documents, are far from representing the real business logic.

Instead, we have a set of statements that is:

a) incomplete (forgotten features),
b) inconsistent (contains contradictions),
c) ambiguous (more possible interpretations).

Elements of the RE are elicitation, specification and validation:

- Elicitation – the aim is to gain knowledge relevant to a problem in order to produce a requirements model.
- Specification – Software Requirement Specification (SRS) will be described later in this chapter.
- Validation – the purpose is to check whether the requirements specification complies with stakeholders' intentions.

The requirements elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders. Requirements elicitation practices include:

- interviews,
- user observation,
- brainstorming,
- use cases,
- role playing,
- and prototyping.

Problems that indicate the challenges for requirements elicitation identified by [74] are:

1. **Problems of scope** - the boundary of the system is ill-defined or the users specify unnecessary design information. It means the technical detail that may confuse, rather than clarify system objectives and features.
2. **Problems of understanding**
    a. users have incomplete understanding of their needs,
    b. users have poor understanding of computer capabilities and limitations,
    c. users don't have a full understanding of the problem domain,
    d. analysts have poor knowledge of problem domain,
    e. user and analyst speak different languages,
    f. ease of omitting "obvious" information on both sites,
    g. conflicting views of different users and their needs,
    h. requirements are often vague and untestable, e.g., "user friendly" and "robust".

3. **Problems of volatility** - the requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility.

The aim of requirements elicitation is to find, analyze and describe customers' needs known as requirements. Before software can be implemented, we need to analyze these requirements and we distinguish the following kinds:

- Application requirements
  - Functional requirements
  - Non-functional or Extra-functional requirements
- Domain requirements – requirements that come from the characteristics of the system domain.

Benefits on applying ontologies in RE leads to reduce the negative effects of the previous factors on the RE processes. The potential uses of ontologies in RE include the representation of:

- The requirements model – requirements ontology,
- acquisition structures for domain knowledge – software requirements specification document ontology,
- the knowledge of the application domain – application domain ontology.

6.3 SW specification analysis

Our approach to semantic analysis of software specifications with Linked Data consists of these steps:

1) Extraction and normalization of use-cases,
2) extraction of functional and extra-functional properties,
3) actors detection.

6.3.1 Extraction and normalization of use-cases

Extraction of use-cases is based on combination of related keywords detection and use-case patterns recognition. The simplest case is to find words "use-case" or "UseCase" in a header or paragraph followed by a list of potential steps. The example of regular expression just for this header detection is in Formula 6.1.

$$\b([Uu]se\-?[Cc]ase) \quad (6.1)$$

Common software specifications usually involves structured lists of steps starting with a serial number or a special character like "-", "*", "#" etc. The example of regular expression for step detection is in Formula 6.2. This regular expression detects the start of potential step and marks it for additional processing.

$$[\-|*|\#|A\text{-}Za\text{-}z|1\text{-}9*]+[\.|\)]* \quad (6.2)$$

Complete regular expression for extraction of text from steps is in Formula 6.3.

$$[\-|*|\#|A\text{-}Za\text{-}z|1\text{-}9*]+[\.|\)]*(\w+) \quad (6.3)$$

Normalization consists in replacing the starting character by a number and in the creation of a new list by merging several smaller. Example of use-case processing and its transformation into XML is in Table 6.1.

Table 6.1: Example of use-case processing

Input	Output
1. first step 2. second step	<use_case_step start_with="1." no="1">1. first step </use_case_step> <use_case_step start_with="2." no="2" >2. second step </use_case_step>
A. first step B. second step	<use_case_step start_with="A." no="1">A. first step</use_case_step> <use_case_step start_with="B." no="2" >B. second step </use_case_step>
- first step - second step	<use_case_step start_with="-" no="1"> - first step</use_case_step> <use_case_step start_with="-" no="2"> - second step</use_case_step>

6.3.2 Extraction of functional and extra-functional properties

Software Requirements Specification (SRS) documents contains all the requirements specifications for a software system, typically separated into functional requirements (FR) and non-functional requirements (NFR). NFRs usually impact the system as a whole and interact both with each other and with the functional requirements.

Figure 6.2 illustrates mappings from requirements items in a SRS document to elements in ontology. Mapping from requirements items to thesaurus can be written formally – formula 6.4:

$$F_{int} : \text{ReqItem} \rightarrow 2^{Con \cup Rel} \quad (6.4)$$

Where

- OntologySystem = (*Con*, *Rel*, Rules).
- *Con* is a set of concepts.
- *Rel* is a set of relationships.
- *ReqItem* is a set of requirements items in a document.
- F_{int} is the interpretation function. In case of Fig. 6.2: F_{int}(bbb) = {A, B, aggregate(A,B)} and F_{int} (ccc) = {D, E}

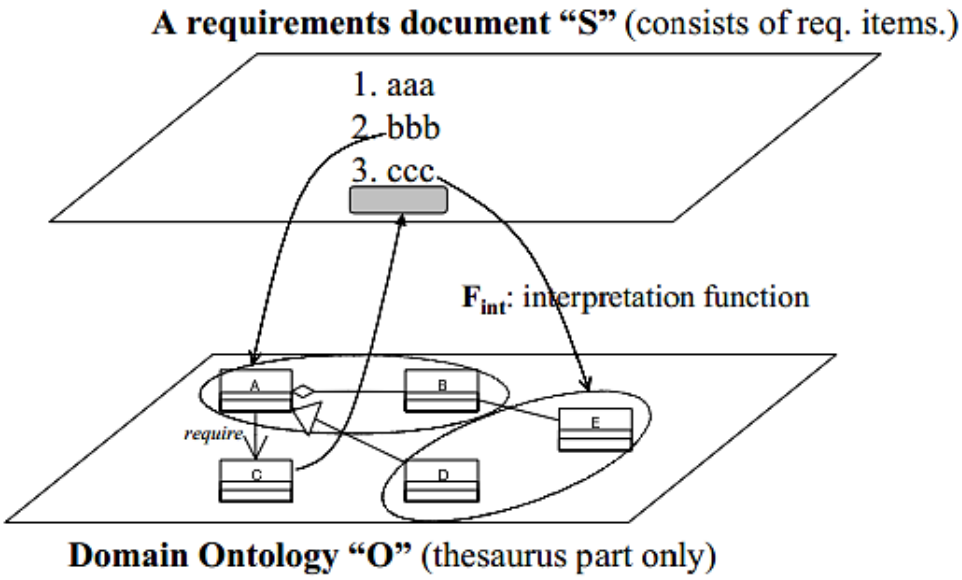

Figure 6.2: Mapping from SRS to ontology [75]

The mapping between the statements and ontology can also be done by using a frame of natural language. This approach is similar to NER but ontology is used instead of a lexicon.

This approach provides following validation options:

- Inconsistency – approach tries to detect mutually contradicting elements where requirement items are mapped. For example relation between *run-time loading* and *short response time*.
- Completeness - completeness of a SRS document can be measured by number of ontology elements related to items in a document. Missing requirement items can be suggested to the user.

6.3.3 Actors detection

Actors detection is combination of methods for NER and requirements extractions proposed above. Examples of entities extracted from case in Figure 6.3 are:

- Concepts: search, search results, search criteria
- Actors:
  - Buyer – synonyms for this actor will be proposed and confirmed by user.
  - System – named entity "Information System" will be proposed and linked to this actor.
- Potential named entity: Information System – probably same as actor "system"
- Extension or variation of use-case:
  - Is probably under the headline "Extensions".
  - Change in the list of enumerated items indicates extension or variation of the use-case.
- Other entities: matches

UseCase: **Buyer** searches for an **offer**
Scope: Marketplace
SuD: Marketplace **Information System**
Primary actor: Buyer
Main **success scenario specification**:

5. **Buyer** enters basic **search criteria**
6. **System** responds with the list of **matches**
7. **Buyer** requests the complete listing of a selected **offer**
8. **System** responds with the requested information

Extensions:
2a. No **matches** found
2a1. Use case aborted
Sub-**variations**:
2b. The amount of **matches** i stoo high
2b1. **Buyer** narrows the **search results** with additional criteria

Figure 6.3: DBpedia Spotlight annotation of sw specification

6.3.4 Evaluation

Nowadays there is no public corpus for evaluation of approaches for SRS analysis. Therefore we had to evaluate our approach based on publicly available specifications found on Internet. We used a random set of 100 pages extracted from software specification documents. This set contained 100 use-cases, 351 actors and 235 features. Our simple ontology contained 120 triples in the form of a subject, predicate and object. Each of these resources was identified by URI.

The results are below:

- NER and actors detection – our experimental system found these entities with F1 = 0.8.
- Normalization of use-cases – 70% of use-cases was normalized correctly. The rest was not recognized due to missing titles like “Use case” and missing lists. Generally these use-cases were in the form of natural language sentences.
- Extraction of functional and extra-functional properties ended with F1 = 0,38 due to lacking ontology for these specifications. Domain ontologies are the best choice for related domain SRS. General ontologies are usually insufficient. It seems us as a good idea to use extraction ontology for a common combination of requirement property with SI base unit.

6.4 Discussion

In this chapter we proposed an approach for semantic analysis of software specifications with Linked Data and ontologies. We introduced Linked Data, ontologies and relevant public web services like DBpedia Spotlight. We evaluated application of Linked Data and ontologies to requirements engineering. The proposed methods for actors identification, extraction of functional and extra-functional properties based on Linked Data, standard ontologies and extraction ontologies are quite promising.

The evaluation of this approach was carried out using a subset from real life SRS documents. NER, actors detection and normalization of use-cases worked very well. On the other hand the requirement extraction is highly dependent on the quality of the used ontology. Only small, simple and general ontology was used in our case therefore the result in the form of F1 measure was relatively bad.

In the future it is essential to evaluate this approach with robust public corpus, stable external libraries and web services. This evaluation was realized with non-public data from private sector, several libraries and web services were in beta version.

7 PageRank variants in the evaluation of citation networks

This chapter, previously published in cooperation with Michal Nykl [27], explores a possible approach to a research evaluation, by calculating the renown of authors of scientific papers. The evaluation is based on the citation analysis and its results should be close to a human viewpoint. The PageRank algorithm and its modifications were used for the evaluation of various types of citation networks that can be processed in a similar way as networks obtained from Linked Data.

Our main research question was whether better evaluation results were based directly on an author network or on a publication network. Other issues concerned, for example, the determination of weights in the author network and the distribution of publication scores among their authors.

The citation networks were extracted from the computer science domain in the ISI Web of Science database. The influence of self-citations was also explored. To find the best network for a research evaluation, the outputs of PageRank were compared with lists of prestigious awards in computer science such as the Turing and Codd award, ISI Highly Cited and ACM Fellows.

Our experiments proved that the best ranking of authors was obtained by using a publication citation network from which self-citations were eliminated, and by distributing the same proportional parts of the publications' values to their authors. The ranking can be used as a criterion for the financial support of research teams, for identifying leaders of such teams, etc.

7.1 Introduction

The evaluation of universities' prestige usually covers several areas such as research results, education, student satisfaction and others. When evaluating research, publications play an important role. Publications and their citations can best show the top researcher in the selected field of science. This evaluation is usually based on the number of publications indexed in e.g. the ISI Web of Science[32] (hereafter WoS) with regard to the number of citations and Journal Impact Factor [76]. The Impact Factor[33] of journal *J* in a given year (e.g. 2011) is the number of citations in this year (2011) to all items published in journal *J* two years before (2010 and 2009) divided by the number of journal J's citable items (i.e.

[32] ISI Web of Science - http://www.webofknowledge.com
[33] Computation of Journal Impact Factor in the WoS database, http://admin-apps.webofknowledge.com/JCR/help/h_impfact.htm

excluding notes, editorials, etc.) published in those two years (2010 and 2009). Note that in the evaluation, the impact factor of citing journals is not taken into account.

Our main method for the evaluation of citation networks is the PageRank algorithm, which uses the impact of citing nodes (articles, authors and so on) for determining the importance of cited nodes. PageRank was introduced by Brin and Page [41] to rank websites and became part of the Google search engine. From its introduction, PageRank has been examined for convergence, acceleration, rating prediction, etc. For example, Langville and Meyer [69] is a good starting point for its deeper study.

PageRank has been frequently used for citation analysis. Fiala [77] worked with the publication citation network and the authorship network to create an author citation network. The determination of edge weights with regard to the publication date and co-authorship is also solved. Other variants of bibliographic network evaluations (comprising e.g. co-citation or co-authorship network) are compared by Yan and Ding [78]. Sidiropoulos and Manolopoulos [79] used the list of ACM SIGMOD E. F. *Codd Innovation Award* holders to compare the results of human and machine rankings of authors. We used the same approach to determine the quality of author rankings but also used some other human evaluation methods. Yu et al. [80] explored a network which combines information from citations, reviews, comments, and information on the reputation of social network users who read articles and comment on them. A comparison and combination of PageRank and the journal impact factor are presented by Bollen et al. [81].

Our main research question was whether better evaluation results were based directly on an author network or on a publication network. We investigate several variants of author or publication citation networks. The influence of self-citations is explored and a further two variants of author ratings are proposed and studied. The author's rating can be obtained either from the weighted author citation networks or as a distribution of publication values among their authors. Other questions, therefore, concern, for example, how to determine the weights in the author network and how to distribute the publication scores among their authors. The evaluation results are compared with lists of the holders of four prestigious computer science awards. Our main contribution demonstrates that the best ranking of authors is obtained by using a publication citation network from which self-citations are eliminated and by distributing the same proportional parts of the publications' values to their authors.

The following section describes the data from the WoS collection, the used lists of prestigious awards and the construction of citation networks of papers or authors. The Types of citation networks section provides information on how to add weights to edges in the author citation network and how to distribute the publication scores among authors. The next section is devoted to our modifications of the PageRank algorithm. The experiments and their results are summarized in the Experiments section and discussed in the Discussion section. The conclusion and recommendation are presented in the last section.

7.2 Data used in our experiments

All of our experiments can be run on an arbitrary bibliographic data collection, but we used the already purchased Thomson Reuters collection employed in our previous studies [77]. This collection consists of all publications classified as "article" published in Journal Citation Reports 2009 in the computer science category between 1996 and 2005. This category covers all seven WoS subcategories: Artificial Intelligence, Cybernetics, Hardware & Architecture, Information Systems, Interdisciplinary Applications, Software Engineering and Theory & Methods.

Using this data, we create two citation networks – the publication network and the author network. The networks can consider various types of self-citations (see Figure 7.1). The first variant, marked *ALL*, takes into account all citations and is, therefore, the most benevolent. The second variant, marked *NOT*, removes citations between publications having at least one common author. For this reason, it is the strictest variant. The last variant is marked *PART*. It is applicable only to author networks and is created from the *ALL* variant by removing all self-loops. Other variants of self-citations are mentioned by Yan, Ding, and Sugimoto [82], who eventually used self-citations with lower weights.

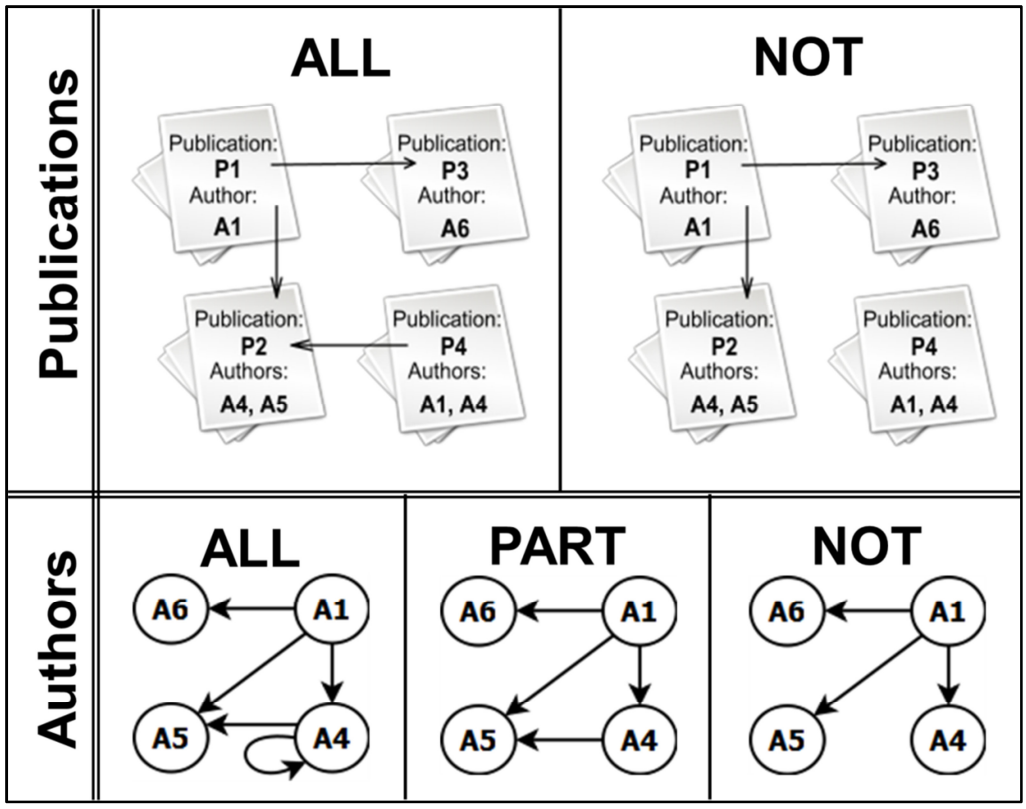

Figure 7.1: Types of self-citations variants used

Our data collection contains 149,347 articles from 386 journals. The average publication was written by 2.5 authors and has 1.3 citations in the *ALL* variant and 1 citation in the *NOT*

variant. The average author has 2.4 publications and 6.8 citations in the *ALL* variant, 6.6 citations in the *PART* variant and 5.4 citations in the *NOT* variant.

In the further described networks we use the following notions:

- *nodes* represent publications or authors
- *edges* represent citations
- *dangling nodes* (marked DN) are nodes which lack an outgoing edge
- *uncited* nodes are nodes which lack an incoming edge
- *isolated* are nodes which lack both outgoing and incoming edges

The numbers of nodes, edges, DNs, uncited nodes, and isolated nodes in our networks are shown in Table 7.1.

Table 7.1: Numbers of elements in the citation networks created from WoS

Types of networks	Nodes	Self-cit.	Edges	DN	Uncited	Isolated
Publication	149 347	ALL	191 447	79 571	90 901	49 774
		NOT	145 372	92 694	103 312	64 517
Author	157 440	ALL	1 062 886	71 354	83 146	48 333
		PART	1 039 339	71 843	83 662	48 482
		NOT	852 356	82 170	94 094	56 907

The comparison of human-made and machine-made author rankings is evaluated with the help of several lists of scientists who are holders of prestigious awards in computer science (mentioned below). From these lists names were removed which, due to their incompleteness, were ambiguous. This approach is similar to that mentioned by Sidiropoulos and Manolopoulos [79] and Lin et al [75]. Name disambiguation and unification has not been performed.

We used lists of the following prestigious awards:

- ***ACM A. M. Turing Award***[34] – ACM's most prestigious technical award is given for major contributions of lasting importance to computing. We use 39 well-distinguishable names from the period 1966–2010.

- ***ACM SIGMOD Edgar F. Codd Innovations Award***[35] – This is given for innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. We use 15 well-distinguishable names from the period 1992–2010.

- ***ACM Fellows***[36] – The ACM Fellows program was established in 1993 to recognize and honor outstanding ACM members for their achievements in computer science and information technology and for their significant contributions to the mission of ACM. We use 576 well-distinguishable names from the period 1994–2011.

- ***ISI Highly Cited***[37] – These highly cited researchers were identified by the Thomson Reuters team between 2000 and 2008 based on an analysis of papers covered in WoS from 1981 to 2008. We use 280 well-distinguishable names.

These unified lists contain 805 scientists who were found in WoS.

7.3 Types of citation networks

This section presents several types of citation networks with regard to the weights of the edges. We will start with the difference between publication networks and author networks.

Whereas publication networks contain information on the time sequences of publications, author networks lack this information. This difference may give an advantage to some authors (e.g. author C in Figure 7.2). We suppose that the publication network better describes the author's influence and is more useful for the evaluation of scientists than the author network. As shown in our experiments, this assumption was proven.

[34] *ACM Turing Award* - http://amturing.acm.org
[35] *ACM SIGMOD Codd* Award - http://www.sigmod.org/sigmod-awards/
[36] *ACM Fellows* - http://fellows.acm.org
[37] *ISI Highly Cited* - http://www.isihighlycited.com

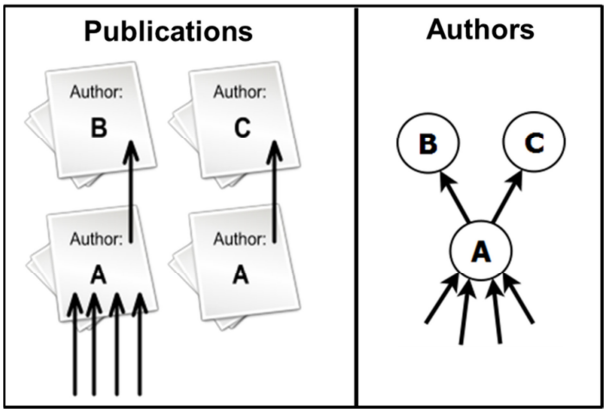

Figure 7.2: Difference between the citation networks of publications and of authors

Other network variants can be obtained by using weights of edges in author networks. Table 7.2 shows three variants of assigning weights to the author networks from Table 7.2. In the first case, marked N, weights expressing the numbers of authors' citations in the publication network are assigned to the graph edges.

For example, if author A has two publications both citing a publication by author B, then in the author network there exists an edge from A to B with weight 2. In the second case, marked 1/N, the publications' values are uniformly distributed to outgoing citations. This means that if a publication by author A1 cites a publication by two authors A4 and A5, then in the author network there exist edges from A1 to A4 and A1 to A5 both with weight 1/2. In the last case, marked 1, weight 1 is assigned to all edges.

Table 7.2: Variants of assigning weights to the author networks

Edge	ALL			PART			NOT		
	N	**1/N**	**1**	**N**	**1/N**	**1**	**N**	**1/N**	**1**
(A1, A6)	1	1	1	1	1	1	1	1	1
(A1, A5)	2	1	1	2	1	1	1	0.5	1
(A1, A4)	2	1	1	2	1	1	1	0.5	1
(A4, A5)	1	0.5	1	1	0.5	1	---	---	---
(A4, A4)	1	0.5	1	---	---	---	---	---	---

A rating of authors can also be obtained from the values of their publications. In this case, we evaluated the PageRank scores for all publications. Consequently, we distributed the publication values to their authors. The distribution can be done by assigning sums of publication values to all their authors, either regardless of the number of co-authors

(marked SUM), or as a proportional part of the publication value depending on the number of authors (marked DIV). For example, if the author has three co-authors in his first publication P1 and five co-authors in his second publication P2, then in variant DIV he receives 1/4 of the P1 score plus 1/6 of the P2 score.

Other variants of network evaluations, not explored in this paper, provide various ways of including the author's position in the list of co-authors below the paper's title. For example, Zhao [83] investigated the influence of authors' positions in the list of co-authors, but he used only the author network, where only the first authors, first N authors, or all authors of a publication were considered. Also, some score distribution variants, like those shown by Assimakis [84], can determine the distribution of publication non-proportional parts with regard to the author's position in the list of co-authors. With respect to the DIV variant results, we aim to test this approach in our next experiments.

7.4 Evaluation of the networks and experiments

The main algorithm that we use for the evaluations of citation networks is the PageRank algorithm [41] [69]. PageRank was designed for the ranking of websites. In this area it was the first recursive algorithm which used in the computation of a website score the scores of the referring (citing) websites. The second algorithm, developed at about the same time, was HITS by Kleinberg [7]. However, unlike HITS, PageRank was widely used and became part of the Google search engine.

We use Formula 7.1:

$$PR_{x+1}(A) = (1 - d) \cdot F(A) + d \cdot L_x(A) \tag{7.1}$$

Where:

- *PRx(A)* is the PageRank score of node *A* in iteration *x*,
- *d* is the damping factor.

The damping factor determines to what extent the final score is computed using the citing nodes' scores (in the PageRank definition this is described as how many times a web user follows hyperlinks to other websites when browsing) and to what extent a random teleport is used (in the definition the random teleport is equal to a situation, in which the user types the address of a random website in the web browser address bar and does not follow any hyperlinks).

By observing web users of the Google, the PageRank authors identified that users used the random teleport once in six steps on average. Therefore, they recommended setting the damping factor to 0.85. The first part of Formula 7.1, marked *F(A)*, represents the random teleport and assigns the same probability of a random jump to all nodes in the network (see Formula 7.2), where |V| is the cardinality of the nodes set *V*.

$$F(A) = 1 \,/\, |V| \tag{7.2}$$

The random teleport is important for convergence when the PageRank values of all nodes in the set are computed iteratively. The second part of the formula, marked *Lx(A)*, represents following the hyperlinks (i.e. graph edges) and uses the values of the citing nodes to determine the values of the cited nodes. Our variant contains weights of edges and resolves the problem of dangling nodes (nodes which lack an outgoing edge), connecting them to all nodes in the network (see Formula 7.3).

$$L_x(A) = (\sum_{u \in U} \frac{PR_x(u) \cdot w_{u\text{toA}}}{w_{u\text{out}}} + \frac{1}{|V|} \sum_{s \in D} PR_x(s)) \tag{7.3}$$

Where:

- U is the set of nodes which have an edge incoming to node A,
- $w_{u\text{toA}}$ is the weight of the edge from node u to node A,
- $w_{u\text{out}}$ is the sum of weights of all outgoing edges from node u,
- D is the set of all dangling nodes in the network.

In general, the random teleport gives the same teleportation probability to all nodes. A non-uniform teleportation is called personalization [41]. We experimented with two personalizations. In the first case, we replaced the uniform distribution of the random jumping to all nodes with a distribution favoring authors with more publications (see Formula 7.4):

$$F(A) = A_{Pub} / \sum_{a \in V} a_{Pub} \tag{7.4}$$

Where:

- A_{Pub} is the number of author A's publications,
- V represents the set of all authors.

High publication numbers may identify authors who are popular and, as was mentioned by Ding [85], "being popular" is necessary for "being prestigious". Therefore, we assume that using publication numbers as personalization can provide better results of author evaluations. We can also say that authors are rewarded for their productivity.

As was shown by Glänzel [86], publications written by more authors coming from more countries are cited more frequently. Our question is whether more authors of a publication can contribute to a higher publication quality and, consequently, to a better evaluation of authors. Therefore, in the second case, we use personalization favoring publications with a higher number of authors (see Formula 7.5).

$$F(P) = P_{Aut} / \sum_{p \in V} p_{Aut} \tag{7.5}$$

Where:

- P_{Aut} is the number of publication P's authors,

- *V* represents the set of all publications.

We experimented with Formula 7.2, 7.4, and 7.5 as the first part of Formula 7.1. To compare PageRank results with a less ingenious method, we used in-degree ranking calculating node values according to Formula 7.6.

$$InD(A) = \sum_{u \in U} w_{u\text{toA}} \tag{7.6}$$

Where:

- *InD(A)* is the in-degree score of node *A*,
- *U* is the set of nodes which have an edge incoming to node *A*,
- w_{utoA} is the weight of the edge leading from node u to node *A*.

Other ways of automatic author ranking can be found in Sidiropoulos and Manolopoulos [79].

The aim of our experiments was to find a way of automatic scientist ranking which produces a ranking closest to the established rankings used by humans. Therefore, we use all the above described variants of citation network evaluations and compared the obtained resulting lists of authors with the previously mentioned lists of awarded scientists. The results obtained are contained in Table 7.3, where the rows combine the use of:

- author/publication network
- methods of evaluation (*in-degree* counting or *PageRank* algorithm)
- exclusion/inclusion of self-citations (marked as *NOT* or *ALL* in the case of publication networks and *NOT*, *PART* or *ALL* in the author network evaluation)
- variants of assigned weights in the author networks (*1* or *1/N* or *N*) and distribution of publication values to their authors (*DIV* or *SUM*)
- variant of the first part of the PageRank Formula 7.1 – Formula 7.2 or Formula 7.4 in the case of author networks and Formula 7.2 or Formula 7.5 in the case of publication networks.

In this table, the value contained in each cell of the column “sum” represents the sum of positions of awarded authors in the respective ranking variant. The authors with the highest ranks occupy the first positions in the ranking; therefore, the lowest sum of positions is the best. If two authors have the same scores and are, for example, on the second and third position in the ranking, then their position is determined as 2.5. The columns “r” show the ranks of all 39 evaluation variants for five different author sets when they are sorted in ascending order by the respective sums. Again, the lower the value of “r”, the better the specific variant is. For a better illustration of results, the best seven ranking variants in Table 7.3 are highlighted with a gray background and the worst twelve variants are highlighted in bold text. The author ranking based on a pure citation count (we may call it a baseline ranking) is the in-degree variant working with network ALL and weights N. In Table 7.3, this

variant is marked with an asterisk (see the row "N*") and highlighted with a black background.

Table 7.3: Comparison of the resulting author rankings
with the lists of prestigious award winners

network	method	self-cit.	weight	formula	ACM Turing *(39)*		ACM Codd *(15)*		ACM Fellows *(576)*		ISI Highly Cited *(280)*		Unification *(805)*	
					sum	r	sum	r	sum	r	sum	r	sum	r
Author	In-degree	NOT	1		1,60E+06	16	**5,94E+05**	**34**	1,76E+07	25	**7,30E+06**	**34**	**2,54E+07**	**30**
			1/N		1,51E+06	10	**5,79E+05**	**31**	1,74E+07	22	**7,16E+06**	**31**	2,49E+07	23
			N		1,61E+06	19	**6,02E+05**	**35**	1,76E+07	26	**7,30E+06**	**33**	**2,54E+07**	**31**
		PART	1		**1,73E+06**	**31**	5,17E+05	16	**1,80E+07**	**35**	**7,44E+06**	**36**	**2,59E+07**	**36**
			1/N		1,64E+06	24	5,43E+05	21	**1,78E+07**	**29**	**7,28E+06**	**32**	**2,55E+07**	**32**
			N		**1,76E+06**	**34**	5,36E+05	18	**1,82E+07**	**38**	**7,50E+06**	**38**	**2,62E+07**	**38**
		ALL	1		**1,74E+06**	**33**	5,25E+05	17	**1,81E+07**	**36**	**7,46E+06**	**37**	**2,60E+07**	**37**
			1/N		1,66E+06	27	5,51E+05	24	**1,79E+07**	**32**	**7,32E+06**	**35**	**2,56E+07**	**35**
			N*		**1,77E+06**	**35**	5,45E+05	23	**1,83E+07**	**39**	**7,52E+06**	**39**	**2,62E+07**	**39**
	PageRank	NOT	1	(2)	1,50E+06	7	**6,23E+05**	**38**	1,72E+07	19	6,79E+06	19	2,45E+07	19
				(4)	1,57E+06	11	4,98E+05	12	1,56E+07	7	6,60E+06	10	2,27E+07	8
			1/N	(2)	1,45E+06	5	**6,06E+05**	**37**	1,71E+07	18	6,79E+06	18	2,44E+07	18
				(4)	1,50E+06	8	4,81E+05	11	1,54E+07	6	6,58E+06	8	2,25E+07	6
			N	(2)	1,50E+06	6	**6,28E+05**	**39**	1,72E+07	20	6,78E+06	17	2,45E+07	20
				(4)	1,57E+06	12	5,02E+05	13	1,56E+07	8	6,58E+06	9	2,27E+07	7
		PART	1	(2)	1,61E+06	18	5,45E+05	22	**1,78E+07**	**28**	6,87E+06	21	2,51E+07	26
				(4)	1,64E+06	25	3,91E+05	4	1,62E+07	14	6,69E+06	13	2,34E+07	14
			1/N	(2)	1,57E+06	13	**5,76E+05**	**30**	1,76E+07	27	6,89E+06	22	2,50E+07	24
				(4)	1,59E+06	14	4,48E+05	8	1,60E+07	12	6,69E+06	12	2,32E+07	11
			N	(2)	1,63E+06	23	**5,58E+05**	**28**	**1,79E+07**	**33**	6,92E+06	23	**2,53E+07**	**28**
				(4)	**1,67E+06**	**29**	4,01E+05	6	1,63E+07	15	6,74E+06	14	2,35E+07	15
		ALL	1	(2)	1,63E+06	21	5,56E+05	26	**1,79E+07**	**34**	6,95E+06	25	**2,53E+07**	**29**
				(4)	**1,66E+06**	**28**	4,00E+05	5	1,64E+07	16	6,75E+06	16	2,36E+07	16
			1/N	(2)	1,60E+06	15	**5,88E+05**	**33**	**1,78E+07**	**30**	6,98E+06	26	2,52E+07	27
				(4)	1,61E+06	20	4,54E+05	9	1,62E+07	13	6,74E+06	15	2,34E+07	13
			N	(2)	1,65E+06	26	**5,70E+05**	**29**	**1,81E+07**	**37**	**7,00E+06**	**28**	**2,55E+07**	**34**
				(4)	**1,68E+06**	**30**	4,10E+05	7	1,64E+07	17	6,80E+06	20	2,37E+07	17
Publication	In-degree	NOT	DIV		1,51E+06	9	**5,85E+05**	**32**	1,73E+07	21	6,94E+06	24	2,46E+07	21
			SUM		1,60E+06	17	**6,05E+05**	**36**	1,74E+07	23	6,98E+06	27	2,49E+07	22
		ALL	DIV		1,63E+06	22	5,58E+05	27	1,76E+07	24	**7,06E+06**	**29**	2,51E+07	25
			SUM		**1,73E+06**	**32**	5,54E+05	25	**1,79E+07**	**31**	**7,13E+06**	**30**	**2,55E+07**	**33**
	PageRank	NOT	DIV	(2)	1,24E+06	1	5,12E+05	15	1,47E+07	2	6,38E+06	2	2,14E+07	2
				(5)	1,28E+06	3	5,06E+05	14	1,46E+07	1	6,35E+06	1	2,14E+07	1
			SUM	(2)	**1,83E+06**	**36**	4,56E+05	10	1,52E+07	5	6,42E+06	3	2,24E+07	5
				(5)	**2,04E+06**	**38**	2,94E+05	1	1,56E+07	9	6,47E+06	6	2,27E+07	9
		ALL	DIV	(2)	1,27E+06	2	5,41E+05	20	1,50E+07	4	6,44E+06	5	2,18E+07	4
				(5)	1,33E+06	4	5,37E+05	19	1,49E+07	3	6,44E+06	4	2,18E+07	3
			SUM	(2)	**1,88E+06**	**37**	3,86E+05	3	1,56E+07	10	6,56E+06	7	2,28E+07	10
				(5)	**2,09E+06**	**39**	3,19E+05	2	1,60E+07	11	6,60E+06	11	2,33E+07	12

7.5 Discussion

Let us start by considering whether a better resulting order of authors can be obtained by evaluating author citation network or publication citation network. A comparison of the results is presented in Table 7.3. The average values of columns “r” for variants marked *Publication* give better results than the average values in columns “r” for variants marked Author. In the column *Unification* (unifying all the four used sets of awarded scientists), the average value of “r” is 12.3 for variants Publication and 23.4 for variants Author.

In the evaluation of the publication citation networks, the PageRank algorithm outperforms the simpler in-degree evaluation. This result was expected because PageRank is able to take into account the quality of citing publications. In the column Unification the average rank of the PageRank variants is 5.8 and of the in-degree variants 25.3.

From Table 7.3 we can see the better results for publication networks without self-citations (marked NOT). In the column Unification, an average rank of 4.3 is given, while those networks containing self-citations (marked ALL) give an average rank of 7.3. Further, we can see that it is better to distribute to the authors the proportional part than the full value of their publications (from the publication citation network without self-citations). Compare the variants marked DIV and SUM. For these variants the average ranks in the column Unification are 1.5 and 7. The only exception is the evaluation based on the Codd prize. The reason could be:

1) this prize is awarded only in the area of database systems and our collection covers the complete computer science area;
2) the number of awarded authors is too small (only 15 persons).

Finally, a better order of authors is provided by the PageRank variant which uses Formula 7.5 rather than Formula 7.2. Different results are yielded by the evaluation with the Turing and Codd prizes. However, these results are not substantially worse and are justified by the small number of awarded authors.

Table 7.4 and Table 7.5 show the differences between the most interesting variants of the evaluation. The variations contain the variants applying the PageRank algorithm to the publication citation network. The pure author citation rank and the best PageRank variant using the author network are added too. Table 7.4 contains the Spearman rank correlation coefficients. The lower correlation between the results of the publication network and the author network and the pure author citation count is also shown. Table 5 shows how many authors are included by different pairs of evaluation variants on the top 100 positions. We also tested authors’ top 1000 positions, but the results were similar to those shown in Table 7.5. The numbers presented indicate that the most similar rankings are produced by Formula 7.2 and 5 and that the pure author citation rank is the most distant from all other variants.

Table 7.4: Spearman rank correlation coefficients for some evaluation variants.

network	method	self-citations	weights	formula	Author		Publication							
					In-D.	PR	PageRank							
					ALL	NOT	NOT				ALL			
					N*	1/N	DIV		SUM		DIV		SUM	
					-	(4)	(2)	(5)	(2)	(5)	(2)	(5)	(2)	(5)
Author	In-D.	ALL	N*	-	1	0,608	0,561	0,562	0,661	0,629	0,555	0,559	0,644	0,615
	PR	NOT	1/N	(4)	0,608	1	0,841	0,860	0,876	0,870	0,829	0,847	0,857	0,844
Publication	PageRank	NOT	DIV	(2)	0,561	0,841	1	**0,997**	0,915	0,877	**0,985**	**0,977**	0,894	0,849
				(5)	0,562	0,860	**0,997**	1	0,932	0,903	**0,985**	**0,984**	0,914	0,877
			SUM	(2)	0,661	0,876	0,915	0,932	1	**0,986**	0,905	0,921	**0,979**	**0,959**
				(5)	0,629	0,870	0,877	0,903	**0,986**	1	0,873	0,898	**0,974**	**0,980**
		ALL	DIV	(2)	0,555	0,829	**0,985**	**0,985**	0,905	0,873	1	**0,996**	0,918	0,876
				(5)	0,559	0,847	**0,977**	**0,984**	0,921	0,898	**0,996**	1	0,938	0,906
			SUM	(2)	0,644	0,857	0,894	0,914	**0,979**	**0,974**	0,918	0,938	1	**0,987**
				(5)	0,615	0,844	0,849	0,877	**0,959**	**0,980**	0,876	0,906	**0,987**	1

The twelve best correlation coefficients of different pairs of evaluation variants are highlighted. The row marked with “N*” represents the author ranking based on the pure citation count.

Table 7.5: Numbers of authors included by both evaluation variants of a pair of rankings on the first 100 positions in the ranking

networks	methods	self-citations	weights	formula	Author		Publication							
					In-D.	PR	PageRank							
					ALL	NOT	NOT				ALL			
					N*	1/N	DIV		SUM		DIV		SUM	
					-	(4)	(2)	(5)	(2)	(5)	(2)	(5)	(2)	(5)
Author	In-D.	ALL	N*	-	100	41	47	50	56	55	47	50	57	58
	PR	NOT	1/N	(4)	41	100	51	52	40	39	40	41	37	36
Publication	PageRank	NOT	DIV	(2)	47	51	100	**96**	69	67	**79**	**79**	66	62
				(5)	50	52	**96**	100	72	70	**81**	**81**	70	66
			SUM	(2)	56	40	69	72	100	**96**	65	67	**86**	**85**
				(5)	55	39	67	70	**96**	100	63	65	**84**	**84**
		ALL	DIV	(2)	47	40	**79**	**81**	65	63	100	**97**	69	65
				(5)	50	41	**79**	**81**	67	65	**97**	100	72	68
			SUM	(2)	57	37	66	70	**86**	**84**	69	72	100	**96**
				(5)	58	36	62	66	**85**	**84**	65	68	**96**	100

The 12 best results are highlighted. The row marked with “N*” represents the author ranking based on the pure citation count.

The top 20 positions from the author rankings obtained by the five best variants are presented in Table 7.6. The authors who are in the first three positions in at least one ranking are highlighted by a gray background. Those who are not among the top 20 researchers in the relevant column but are among the top 20 in any of the other columns are mentioned in the bottom part of the table.

As we can see from Table 7.6, the top names do not vary substantially, e.g., the name Jain, AK permanently occupies one of the top three places. The reason could be that it represents more persons having the same surname. The other interesting name is Setiono, R. It is in first place in the variants using self-citations of authors (variants ALL) but in substantially worse positions in the other three variants (variants NOT). The most probable explanation is too frequent usage of self-citations by the author and his co-authors. The top places of Setiono, R in variant DIV compared with variant SUM indicates he published his articles with only a small number of co-authors.

Table 7.6: The top 20 positions from the author rankings obtained by the five best variants.

Position	Publication / PageRank				
	DIV				SUM
	NOT		ALL		NOT
	(2)	(5)	(2)	(5)	(2)
1	Simon, DR	Simon, DR	Setiono, R	Setiono, R	Jain, AK
2	Breiman, L	Breiman, L	Jain, AK	Jain, AK	Vazirani, U
3	Jain, AK	Jain, AK	Yager, RR	Breiman, L	Bernstein, E
4	Moltenbrey, K	Yager, RR	Breiman, L	Yager, RR	Simon, DR
5	Yager, RR	Moltenbrey, K	Simon, DR	Simon, DR	Breiman, L
6	Robertson, B	Vazirani, U	Moltenbrey, K	Vazirani, U	Tanaka, K
7	Vazirani, U	Bernstein, E	Vazirani, U	Moltenbrey, K	Yager, RR
8	Bernstein, E	Zadeh, LA	Bernstein, E	Bernstein, E	Kim, J
9	Zadeh, LA	Robertson, B	Pedrycz, W	Pedrycz, W	Lee, J
10	Pedrycz, W	Pedrycz, W	Robertson, B	Zadeh, LA	Chang, CC
11	Amari, S	Hyvarinen, A	Zadeh, LA	Robertson, B	Lee, S
12	Hyvarinen, A	Amari, S	Amari, S	Amari, S	Pedrycz, W
13	Chang, CC	Oja, E	Wang, J	Wang, J	Wang, J
14	Oja, E	Chang, CC	Hyvarinen, A	Hyvarinen, A	Osher, S
15	Wang, J	Tanaka, K	Oja, E	Oja, E	Kim, JH
16	Tanaka, K	Wang, J	Chang, CC	Lee, J	Wang, Y
17	Lee, J	Burges, CJC	Lee, J	Chang, CC	Moltenbrey, K
18	Burges, CJC	Lee, J	Tanaka, K	Tanaka, K	Bennett, CH
19	Lee, S	Lee, S	Egghe, L	Lee, S	Oja, E
20	Zhang, J	Kim, J	Dannenberg, RB	Picard, RW	Wang, HO
	(21) Kim, J	(21) Zhang, J	(22) Lee, S	(22) Dannen..	(24) Zhang, J
	(26) Kim, JH	(25) Kim, JH	(23) Picard, RW	(26) Kim, JH	(28) Amari, S
	(32) Picard, ..	(31) Wang, Y	(27) Zhang, J	(27) Egghe, L	(34) Picard, ..
	(33) Wang, Y	(33) Picard, ..	(28) Kim, JH	(28) Zhang, J	(41) Hyvarin..
	(41) Egghe, L	(52) Egghe, L	(31) Kim, J	(31) Kim, J	(46) Robert..
	(56) Wang, ..	(53) Wang, ..	(35) Burges, CJC	(32) Burges, ..	(57) Zadeh, ..
	(59) Osher, S	(56) Osher, S	(36) Wang, Y	(36) Wang, Y	(128) Burge..
	(68) Setiono, ..	(75) Setiono, ..	(52) Osher, S	(41) Osher, S	(172) Setion..
	(85) Bennett..	(92) Bennett..	(65) Wang, HO	(64) Wang, ..	(204) Egghe, ..
	(1401) Dann..	(1585) Dann..	(100) Bennett, ..	(106) Bennet..	(3948) Dann..

The authors who are in the top three positions in at least one ranking are highlighted by a gray background. Those who are not among the top 20 researchers in the relevant column but are among the top 20 in any of the other columns are mentioned in the bottom part of the table.

7.6 Suggestions for future work

In this chapter we introduced several variants of citation networks and their usage in the evaluation of authors' prestige. In our created orders of authors, those authors who are holders of prestigious awards granted by scientific societies were found. The goal was to determine those evaluation variants providing the closest ranking to the human opinion. Our main research question was whether better results are provided by an evaluation based directly on an author network or an evaluation based on a publication network. The other problems under study were how to determine the weights in the author network and how to distribute publication scores among their authors.

As the best variant we recognized the one applying the PageRank algorithm to the publication citation network without self-citations and distributing the PageRank values of publications proportionally to their authors. Briefly put, this is the variant Publication/PageRank/NOT/DIV/(5) from Table 7.3. The results by Formula 5 are of nearly the same quality as those by Formula (7.2). The pure citation count gives considerably worse results. We should underline that the results obtained are based on the data from the ISI Web of Science database (almost 150,000 computer science journal articles from 1996-2005) and that we also applied the same algorithms to data from other sources (DBLP and CiteSeer), but the results were not so convincing in order to be included in this paper. Our explanation of the results not presented here is the lower quality of the citation networks based on these data sources (only 0.21 citations per publication in DBLP and many indexing errors in CiteSeer [87].

In the future it would be nice to apply the PageRank algorithm to journal citation networks to compare our results with author rankings taking into account Journal Impact Factors of journals in which authors published. These values could consequently be integrated into our formulas to obtain author orderings. The distribution of publications' values to authors depending on the authors' order in the paper byline is also worthy of investigation. We consider the evaluation of prestige of workplaces and institutions as another interesting challenge.

8 Conclusions

This thesis aimed at application of additional semantic information to standard text-mining tasks on text documents obtained from Web. Moreover we realized experiments with PageRank and used it mainly for feature extraction and citation analysis. New modifications of PageRank were proposed and evaluated. The experiments in Scientometrics area [27] are supported by important international journal indexed by Web of Science with impact factor 4.153.

Looking back at the goals from Chapter 1, this thesis fulfilled goals summarized as follows:

G1 *Propose a feature selection method with additional semantic information.*

- The algorithm for feature selection with Linked Data was introduced in Chapter 4.
- The method for feature selection with PageRank was proposed in Chapter 5.

G2 *Application and evaluation of feature selection from goal G1 to classification and clustering.*

- The evaluation of clustering algorithm was introduced in Chapter 4.
- The evaluation of classification algorithm was proposed in Chapter 5.

G3 *Cluster labeling with Linked Data.*

- This approach was proposed in Chapter 4 section 4.3.

G4 *Software specification analysis with Linked Data.*

- This approach was proposed in Chapter 6. It consists of text-mining techniques for use-case identification, extraction of requirements, including both functional and extra-functional ones, and for actors detection.

G5 *Citation network analysis with PageRank.*

- This approach was introduced in Chapter 7.
- The Thomson Reuters subset was used with prestigious awards list for evaluation in this case.

8.1 Evaluation and data sets

The most important data sets used in this thesis for evaluation of introduced approaches:

- My collection of Call for papers[38] announcements – this collection currently contain 18 000 records with automatically extracted information: dates with classification to common classification classes related to this topic (send abstract, send article, send camera-ready version, start and end of a conference), URL of website and venue (extracted by NER).
- The 20 News groups dataset [63] – cleaned subset about information technology.
- Open Directory Project [64] – cleaned subset about information technology.
- I have collected several newspapers collection with our web-crawler. It consists of approximately 30 000 articles with basic category like economy or IT. Some of the sources are:
    - The New York Times[39] - information technology, economics
    - BBC[40] – disasters and information technology
- ISI Web of Science[41] - approximately 150 000 articles from 386 journals, a subset of this collection was compared to the Codd award (SIGMOD Edgar F. Codd Innovations Award[42] and other related awards)

The most important used sources of Linked Data:

- DBPedia[43] – English subset, approximately 2 GB of annotated data.
- Freebase[44] - English subset, approximately about 200 000 of locally indexed nodes, it was used especially for NER with Linked Data.
- Geo-names[45] – used together with DBPedia.

[38] http://tmrg.cz/
[39] The New York Times - http://www.nytimes.com/
[40] BBC - http://www.bbc.com/
[41] ISI Web of Science - http://www.webofknowledge.com
[42] Codd Award - http://www.sigmod.org/sigmod-awards
[43] DBPedia - http://dbpedia.org/
[44] Freebase - https://www.freebase.com/
[45] Geo names - http://www.geonames.org/

8.2 Contributions

Contribution to the field of information technology published in my previous thesis [88] proposal:

- Simple text-classifier – it is based on scores instead of probabilities was published on international conference ITAT and introduced in the doctoral thesis. The F-measure of this simple and fast classifier was approximately 0.92 for simple tasks like classification of dates in Call for papers announcements.
- An approach for transformation of plain-text documents into semantic web:
    - Indexing – our own simple and fast crawler,
    - Information extraction - named entities and dates,
    - Web 2.0 technologies – HTML 5 + CSS 3, Ajax + REST, JSON and XML,
    - Semantic web – documents were published in the form of Semantic Web (RDFa) and dates were marked for simple processing by Google Calendar.

Contribution published in this thesis:

- Text-mining methods enhanced with semantic information from Linked Data:
    - Feature selection with Linked Data and PageRank.
    - Application of feature selection to classification and clustering.
    - Method for cluster labeling with Linked Data.
    - An approach for software specification analysis with Linked Data.
    - Citation network analysis with PageRank.

All proposed methods were evaluated with well-known test collections if available. Other methods were evaluated manually. Semantic versions show significantly better results than statistical versions of text-mining methods. Therefore I suppose that these methods will be targeted in next research. This presumption is backed up with growing number of research articles related to Linked Data as source of publicly available and machine readable information. An approach for software specification analysis is completely new and first of its kind. Other related research articles are usually theoretically oriented.

These methods were previously published in Journals and proceedings of conferences. The list of the most important publications is in the last Section “Author's publications” as well as list of the current citations of these publications.

Bibliography

[1] D. Hand, . H. Mannila and P. Smyth, Principles of Data Mining, Cambridge, England: Mit Press, 2001.

[2] C. C. Aggarwal and C. Zhai, Eds., Mining Text Data, New York: Springer New York Dordrecht Heidelberg London, 2012.

[3] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[4] A. Perkins, "The Classification of Search Engine Spam," [Online]. Available: http://www.silverdisc.co.uk/articles/spam-classification. [Accessed 6 2012].

[5] H. Garcia-Molina and Z. Gyongyi, "Web spam taxonomy," Stanford, 2004.

[6] P. Lawrence, B. Sergey, M. Rajeev and W. Terry, "The PageRank Citation Ranking: Bringing Order to the Web," 1999.

[7] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," in *Journal of the ACM (JACM)*, New York, 1999.

[8] L. Wolf and A. Shashua, "Feature Selection for Unsupervised and Supervised Inference: the Emergence of Sparsity in a Weighted-Based Approach," *Proceedings of Ninth IEEE International Conference on Computer Vision*, pp. 378- 384, 2003.

[9] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Proceedings of Artificial Intelligence,* 1997.

[10] P. S. Bradley and O. Mangasarian, "Feature selection via concave minimization and support vector machines," *In: Proceedings of ICML,* 1998.

[11] T. Cover and J. Thomas, Elements of information theory, New York, USA: John Wiley & Sons, 1991.

[12] J. Paralič, K. Furdík and G. Tutoky et. al., Dolovanie znalostí z textov, Košice:

Technická univerzita v Košiciach, 2010.

[13] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *The American Statistician Vol. 46, Iss. 3,* pp. 175-185, 1992.

[14] Y. Schapire and R. Singer, "BoosTexter: A boosting-based system for text categorization," *Machine Learning*, 1999.

[15] W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization.," *ACM SIGIR ’96,* 1996.

[16] A. L. Berger, J. D. Vincent and A. Stephen, "A maximum entropy approach to natural language processing," *Computational linguistics 22.1*, pp. 39-71, 1996.

[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and H. R., "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science 41 (6)*, pp. 391-407, 1990.

[18] W. Wang et al, "Term Graph Model for Text Classification.," *ADMA’05,* pp. 19-30, 2005.

[19] S. Bloedhorn and A. Hotho, "Boosting for Text Classification with Semantic Features," *WebKDD’04*, 2004.

[20] G. Ramakrishnanan and P. Bhattacharyya, "Text Representation with WordNet Synsets using Soft Sense Disambiguation," *Proceedings of the 8th NLDB,* 2003.

[21] G. De Melo and S. Siersdorfer, "Multilingual text classification using ontologies," *Advances in Information Retrieval,* pp. 541-548, 2007.

[22] D. Sutton, "Linguistic problems with requirements and knowledge elicitation," *Requirements Engineering, 5(2),* pp. 114-124, 2000.

[23] J. Coughlan and R. Macredie, "Effective communication in requirements elicitation: A comparison of methodologies," *Requirements Engineering, 7(2),* pp. 47-60, 2002.

[24] L. Chung and J. P. Leite, "On non-functional requirements in software engineering," in *Conceptual modeling: Foundations and applications,* 2009.

[25] F. Geraci, M. Pellegrini, M. Maggini and F. Sebastiani, "Cluster Generation and Cluster Labelling for Web Snippets," in *Lecture Notes in Computer Science*, 2006.

[26] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey, "Scatter/Gather: a cluster-based approach to browsing large document collections," in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, 1992.

[27] M. Nykl, K. Ježek, D. Fiala and M. Dostal, "PageRank variants in the evaluation of citation networks," *Journal of Informetrics*, pp. 683-692, 2014.

[28] M. Dostal, M. Nykl and K. Ježek, "Cluster labeling with Linked Data," *Journal of Theoretical and Applied Information Technology 53 (3)*, pp. 340-345, 2013.

[29] M. Dostal and K. Ježek, "Automatic tagging based on Linked Data," in *IEEE International Conference on Service-Oriented Computing and Applications*, Perth, 2010.

[30] M. Dostal, M. Nykl and K. Ježek, "Semantic analysis of software specifications with Linked Data," *Journal of Theoretical and Applied Information Technology*, pp. 368-376, 2014.

[31] T. B. Lee, "Linked Data – Design Issues," 2009. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html. [Accessed 20 dubna 2011].

[32] A. Jaffri, H. Glaser and I. Millard, "URI Disambiguation in the Context of Linked Data," in *LDOW 2008*, Beijing, China, 2008.

[33] P. N. Mendes, M. Jakob, A. Garcia-Silva and C. Bizer, "DBpedia Spotlight: Shedding Light on the Web of Documents," in *7th International Conference on Semantic Systems (I-Semantics)*, 2011.

[34] A. V. Corasick and M. J. Aho, "Efficient string matching: an aid to bibliographic search," in *Commun ACM*, 1975.

[35] P. Mendes, J. Daiber, R. Rajapakse, F. Sasaki and C. Bizer, "Evaluating the Impact of Phrase Recognition on Concept Tagging," in *Proceedings of the Eight International Conference on Language Resources and Evaluation LREC'12*, Istanbul, Turkey, 2012.

[36] L.-A. Ratinov et al, "Local and Global Algorithms for Disambiguation to Wikipedia," in *ACL vol. 11*, 2011.

[37] G. Rizzo and R. Troncy, "Nerd: evaluating named entity recognition tools in the web of data," in *Workshop on Web Scale Knowledge Extraction*, 2011.

[38] O. Zamir and O. Etzioni, "Web document clustering: A feasibility demonstration," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998.

[39] S. Osiński, J. Stefanowsk and D. Weiss, "Lingo: Search results clustering algorithm based on singular value decomposition," in *Intelligent information processing and web mining*, 2004.

[40] F. Eibe et al., "Domain-specific keyphrase extraction [KEA]," in *Proc. of the 16th international joint conference on AI*, 1999.

[41] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems, 30(1-7)*, pp. 107-117, 1998.

[42] M. Dostal and K. Ježek, "Automatic keyphrase extraction based on NLP and statistical methods," in *Proceedings of the Dateso 2011*.

[43] P. Tarau and R. Mihalcea, "Textrank: Bringing order into texts," *Proceedings of EMNLP 2004*, pp. 404-411, 2004.

[44] S. Rose, D. Engel, N. Cramer and D. Cowley, "Automatic keyword extraction from individual documents," *Text mining applications and theory 2010*, pp. 3-19, 2010.

[45] G. Jones and S. Paynter, "Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications," *Journal of the American Society for Information Science and Technology 53.8*, pp. 653-677, 2002.

[46] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning and E. Frank, "Improving browsing in digital libraries with keyphrase indexes," *Decision Support Systems 27(1–2)*, pp. 81-104, 1999.

[47] A. Hulth, "Combining machine learning and natural language processing for automatic keyword extraction," Stockholm University, Faculty of Social Sciences, Department of

Computer and Systems Sciences (together with KTH), Stockholm, 2004.

[48] S. Golder and B. Huberman, "Usage Patterns of Collaborative Tagging Systems," *Journal of Information Science,* pp. 198-208, 2006.

[49] G. Begelman, P. Keller and F. Smadja, "Automated Tag Clustering: Improving search and exploration in the tag space," in *Proceedings of the Fifteenth International World Wide Web Conference,* Edinburgh, 2006.

[50] C. H. Brooks and N. Montanez, "Improved annotation of the blogosphere via autotagging and hierarchical clustering," in *Proceedings of the 15th International World Wide Web Conference,* Edinburgh, 2006.

[51] S. Sood et al, "TagAssist: Automatic Tag Suggestion for Blog Posts," *Proceedings of the International Conference on Weblogs and Social Media,* 2007.

[52] K. Haklae, Y. Sungkwon, J. Jiwoong, K. Kwangsub and J. G. Breslin, "Combining Tags and the SemanticWeb for Linked Tagging Data," in *Semantic Web Conference 2008,* Karlsruhe, 2008.

[53] A. Passant and P. Laublet, "Meaning Of A Tag: A Collaborative Approach to Bridge the Gap Between Tagging and Linked Data," *Proceedings of the Linked Data on the Web workshop at WWW2008.,* 2008.

[54] B. Wu and B. D. Davison, "Identifying link farm spam pages," in *Proceedings WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, 2005.

[55] B. Fischer and J. H. Buhmann, "Data Resampling for Path Based Clustering," *Pattern Recognition,* pp. 206 - 214, 2002.

[56] M. Dostal and K. Ježek, "Volba vlastností s využitím Linked Data," *Proceedings of Datakon,* 2012.

[57] "Dogpile search engine," [Online]. Available: http://www.dogpile.com/. [Accessed 5th June 2012].

[58] "Yippy search engine," [Online]. Available: http://yippy.com/. [Accessed 8th June 2012].

[59] D. Ramage, P. Heymann, C. D. Manning and H. Garcia-Molina, "Clustering the tagged web," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, New York, 2009.

[60] "Freebase knowledge base," 1st June 2012. [Online]. Available: http://www.freebase.com. [Accessed 1st June 2012].

[61] "Delicious," 2nd June 2012. [Online]. Available: http://www.delicious.com/. [Accessed 2nd June 2012].

[62] D. Carmel, H. Roitman and N. Zwerdling, "Enhancing Cluster Labeling Using Wikipedia," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, New York, 2009.

[63] K. Lang, "Newsweeder: Learning to filter netnews," *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331-339, 1995.

[64] "Open directory project," 2nd June 2012. [Online]. Available: http://dmoz.org/. [Accessed 20th June 2014].

[65] M. Dostal, M. Nykl and K. Ježek, "Exploration of Document Classification with Linked Data and PageRank," *Intelligent Distributed Computing VII*, 2013.

[66] M. Strube and S. Ponzetto, "WikiRelate! Computing semantic relatedness using Wikipedia," *AAAI'06, USA*, 2006.

[67] E. Gabrilovich et al, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," 2007.

[68] N. Ma et al, "Bringing PageRank to the citation analysis," *Information Processing & Management, 44*, pp. 800-810, 2008.

[69] A. N. Langville and C. D. Meyer, Google's PageRank and beyond the science of search engine rankings, Princeton: Princeton University Press, 2006.

[70] R. Cyganiak and A. Jentzsch, "Linking Open Data cloud diagram," 2011. [Online]. Available: http://lod-cloud.net. [Accessed 20tj June 2014].

[71] C. Bizer, T. Heath and T. Berners-Lee, "Linked data-thestory so far," in *Int. Journal on Semantic Web and InformationSystems, Special Issue on Linked Data*, 2009.

[72] P. Loucopoulos and V. Karakostas, System Requirements Engineering, 1995.

[73] B. Nuseibeh and S. Easterbrook, "Requirements Enginnering: a road map," in *Proceedings of ICSE'2000 - Future of Software Engineering Track*, 2000.

[74] M. Christel and K. C. Kang, "Issues in Requirements Elicitation," Technical Report, Pittsburgh, Pennsylvania 15213, 1992.

[75] L. Lin, Z. Xu, Y. Ding and X. Liu, "Finding topic-level experts in scholarly networks," *Scientometrics, 97(3),* pp. 797-819, 2013.

[76] E. Garfield, "Citation analysis as a tool in journal evaluation," *Science 178(60),* pp. 471-479, 1972.

[77] D. Fiala, "Time-aware PageRank for bibliographic networks," *Journal of Informetrics, 6(3),* pp. 370-380, 2012.

[78] E. Yan, Y. Ding and C. R. Sugimoto, "P-Rank: An indicator measuring prestige in heterogeneous scholarly networks," *Journal of the American Society for Information Science and Technology, 62(3),* pp. 467-477, 2010.

[79] A. Sidiropoulos and Y. Manolopoulos, "Generalized comparison of graph-based ranking algorithms for publications and authors," *Journal of Systems and Software, 79(12),* pp. 1679-1700, 2006.

[80] K. Yu, X. Chen and J. Chen, "A multidimensional PageRank algorithm of literatures. Journal of Theoretical and Applied Information Technology, 44(2)," pp. 308-315, 2012.

[81] J. Bollen, M. A. Rodriquez and H. Van De Sompel, "Journal status," *Scientometrics, 69(3),* pp. 669-687, 2006.

[82] E. Yan, Y. Ding and C. R. Sugimoto, "P-Rank: An indicator measuring prestige in heterogeneous scholarly networks," *Journal of the American Society for Information Science and Technology, 62(3),* pp. 467-477, 2010.

[83] D. Zhao, "Going beyond counting first authors in author co-citation analysis," *Proceedings of the American Society for Information Science and Technology, 42(1),* 2005.

[84] N. Assimakis and M. Adam, "A new author's productivity index: p-index," *Scientometrics, 85(2),* pp. 415-427, 2010.

[85] Y. Ding, "Applying weighted PageRank to author citation networks," *Journal of the American Society for Information Science and Technology, 62(2),* pp. 236-245, 2011.

[86] W. Glänzel, "National characteristics in international scientific co-authorship relations," *Scientometrics, 51(1),* pp. 69-115, 2001.

[87] D. Fiala, "Mining citation information from CiteSeer data," *Scientometrics, 86(3),* pp. 1-12, 2011.

[88] M. Dostal, "Znalostní nástroje pro analýzu a prohledávání Webu," University of West Bohemia in Pilsen, Pilsen, 2011.

[89] Unknown, "Clustering to Improve Merchandise Allocation, Testing, and Forecasting: An Application of the K-Medians Algorithm," 10th May 2011. [Online]. Available: http://espin086.wordpress.com/2011/02/27/clustering-to-improve-merchandise-allocation-testing-and-forecasting-an-application-of-the-k-medians-algorithm/. [Accessed 10th May 2011].

[90] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Turkey, "Scatter/gather: a cluster-based approach to browsing large document collections," *SIGIR 1992,* pp. 318-329, 1992.

Author's publications

Abbreviations used in this section:

- **ISI** – publication was indexed by Conference Proceedings Citation Index (ISI)
  - ISI – waiting for indexing
  - ~~ISI~~ – indexing was promised but has not been done
- **Scopus** – publication was indexed by Scopus
  - Scopus – waiting for indexing
- **IEEE Explore** – publication was indexed by IEEE Explore

Journals publications

1)	M. Dostal, M. Nykl and K. Ježek, “Cluster labeling with Linked Data” in *Journal of Theoretical and Applied Information Technology 53 (3)*, pp. 340-345, 2013. ISSN: 1992-8645. Status: published	**Scopus**
2)	D. Chmelařová, Z. Ambler, M. Dostal and V. Vobořilová, “Rehabilitace kognitivních funkcí u pacientů s roztroušenou sklerózou, přehledová práce” in *Cesk Slov Neurol N*, 2014. ISSN: 1802-4041. Status: published	ISI
3)	M. Dostal, M. Nykl and K. Ježek, “Semantic analysis of software specifications with Linked Data” in *Journal of Theoretical and Applied Information Technology*, pp. 368-376, 2014. ISSN: 1992-8645. Status: published	Scopus
4)	M. Nykl, K. Ježek, D. Fiala and M. Dostal, “PageRank variants in the evaluation of citation networks” in *Journal of Informetrics*, pp. 683-692, 2014. ISSN: 1751-1577. Status: published	**ISI**

Contributions to major conferences

1) M. Dostal, M. Nykl, K. Ježek, “Exploration of Document Classification with Linked Data and PageRank”, in *Intelligent Distributed Computing VII*, 2013. **ISI**, **Scopus**

2) M. Nykl, M. Dostal and K. Ježek, “Linked Data and PageRank based classification”, in *IADIS International Conference Theory and Practice in Modern Computing*, 2013. **Scopus**

3) M. Dostal and K. Ježek, “Automatic Keyphrase Extraction based on NLP and Statistical Methods”, in *Dateso*, 2011. **Scopus**

4) M. Dostal and K. Ježek, “Automatic tagging based on linked data: Unsupervised methods for the extraction of hidden information”, in *IEEE International Conference on Service-Oriented Computing and Applications* (SOCA), 2010. ~~ISI~~, **IEEE explore**

Contributions to other conferences

1) M. Dostal, D. Fiala and K. Ježek, “Semantic Markup for Web Applications”, in *Proceedings of the International Conference on Computer, Communication, Information Sciences, and Engineering* (ICCCISE 2013), Paris, France, pp. 506 - 509, 2013. World Academy of Science, Engineering and Technology, vol. 76, April 2013. ISSN: 2010-376X.

2) M. Dostal and K. Ježek, “Volba vlastností s využitím Linked Data”, in *Proceedings of the Datakon* (Datakon 2012), pp. 77-87, 2012. ISBN 978-80-553-1049-7.

Citations

M. Dostal and K. Ježek, “Automatic Keyphrase Extraction based on NLP and Statistical Methods”, in *Dateso*, 2011.

- C. L. Alvargonzález, “ACOTA: Tecnologías de etiquetado semiatomático y colabotarivo”, Master Thesis, 2013.
- R. Wang, W. Liu, and C. McDonald, “How Preprocessing Affects Unsupervised Keyphrase Extraction”. In *Computational Linguistics and Intelligent Text Processing*, pp. 163-176. Springer Berlin Heidelberg, 2014.
- B. Kaur, and B. K. Sidhu, “Methods for key phrase extraction from documents” in *Methods 1(5)*, 2014.
- N. Mitra, N. Goel, and S. Chakraverty, “Adaptive Content Based Textual Information Source Prioritization”. In *Special issue on Distributed Intelligent Systems and Applications*, pp. 829 - 835, 10/2014. ISSN: 2229-6956.

M. Dostal and K. Ježek, “Automatic tagging based on linked data: Unsupervised methods for the extraction of hidden information”, in *IEEE International Conference on Service-Oriented Computing and Applications* (SOCA), 2010.

- P. L. Araúz, P. Faber, and P. J. M. Redondo, “Linking Domain-Specific Knowledge to Encyclopedic Knowledge: an Initial Approach to Linked Data”. In *MSW 2011*, pp. 68-73, 2011.