Assessing the quality of Web via semi-supervised methods

Dávid Siklósi

Supervisor: András Benczúr Ph.D.

Eötvös Loránd University
Faculty of Informatics

Ph.D. School of Computer Science
Erzsébet Csuhaj-Varjú D.Sc.

Ph.D. Program of Basics and methodology of Informatics
János Demetrovics D.Sc.

Budapest, 2016.

Acknowledgments

I would like to express my gratitude to my supervisor András Benczúr who supplied me with guidance and infinite amount of help during my researches.

I also would like to thank the members of the Data Mining and Search Group of MTA SZTAKI for providing such an inspiring environment. Special thanks for the contribution of those whom I worked together on the experiments of this thesis: Károly Csalogány, Bálint Daróczy, András Garzó and Tamás Kiss.

Special thanks for friends and relatives who spent time with reading my dissertation and helped improving its quality: Júlia Biró, István Siklósi, Adrienn Szabó, Bálint Daróczy, Shailvi Wakhlu, Cucu.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In the last decades, Web has grown to be a central part of our lives and substantially changed the way we share information. Users of the Web are not only gathering information from different sources, but also actively editing it via wikis, blogs, forums, social networks and they are commenting, reviewing, tagging and giving opinions on existing content. The immense and continuously growing size of the Web combined with such diversity of information sources makes it nearly impossible to find relevant information without the help of search engines, Web archives and Web directories.

Search engines index billions of Web pages and provide an interface for users to find relevant documents by keyword search. The main task of a search engine is to rank relevant documents in a way that the users have to run through only the first few hits of the result list to find the desired information. In order to retrieve a "good quality" result list, search engines apply sophisticated ranking algorithms based on the keywords the user typed in, the textual content of the Web pages and the Web graph.

In the mid 90's, owners of Web pages started to realize that appearing in the top results of search engines translates into financial benefits. Thus they started to optimize their pages to influence the ranking algorithms of search engines and attract more and more visitors. While some of these activities help improving the quality of Web pages, a number of shifty webmasters started to stuff their pages with very popular query words and use other techniques to increase their ranking scores for query words that are not even relevant to the content of their Web pages. These pages are called Web spam pages. Since spamming activities deteriorate the quality of search results, it became a top priority goal for search engines to recognize and filter spam.

Besides filtering useless content, search engines also invest huge effort in personalizing their search results. Consider the case when a user types the query:

crane. It can refer to an animal, a machinery for lifting heavy weights or the American novelist, Stephen Crane. To decide which topic could be the most favorable for the user, search engines categorize the pages of the Web and create profiles of their users' category preferences based on the queries they typed in and the pages they visited earlier.

In this thesis I survey the existing results and introduce new techniques for automatically identifying spam pages and for classifying the topic of Web pages. I also introduce new aspects of Web quality, which could help assessing Web pages along more dimensions than spam and topical genre.

1.1 Thesis Outline

After this introduction, we continue with a chapter that contains all the fundamental knowledge needed to understand this thesis. I give a brief overview of basic methods, notions and algorithms in the topics of search engine ranking methods, graph similarities, machine learning, clustering algorithms and information theory.

Chapter 3 gives a discussion about different aspects of Web quality. I give a detailed description of what is Web spam and its typology. I introduce the DMOZ web directory and its huge role in assessing Web quality. I also introduce some relatively new aspects of quality such as trustworthiness, factuality and bias. I give a short overview of the baseline methods for classifying Web pages.

In Chapter 4 we compare a wide range of semi-supervised learning techniques both for Web spam filtering and telephone user churn classification [2]. Semi-supervised learning makes the assumption that the label of a node in a graph is similar to those of its neighbors. Our experiments demonstrated that stacked graphical learning in combination with graph similarity methods improves classification accuracy over the baseline methods.

In Chapter 5 we extend and compare the graph stacking features introduced in Chapter 4, with new graph stacking features that characterize the internal link structure of pages within a host. The motivation behind constructing new features is that certain types of web hosts, such as corporate sites, web directories and content hosting services have characteristic internal structures, as captured through features such as the depth of the link graph within the host, the number of pages at various levels, and the distribution of internal and external links from and to the levels. We use our methods both for Web spam filtering and DMOZ topical category classification. We observed that the extra information of internal site structure improves our results by 1-4% in AUC.

In chapter 6 we introduce a new semi-supervised technique for cross-lingual Web classification [4]. While English language training data exists for several

Web classification tasks, most notably for Web spam, we face an expensive human labeling procedure if we want to classify Web hosts in a language different from English. We overview how existing content- and link-based classification techniques work, how models can be translated from English into another language, and how language-dependent and language-independent methods combine. We describe our experiments both for Web spam filtering and DMOZ category classification.

In chapter 7 we analyze the results of the ECML/PKDD Discovery Challenge 2010, especially the results on the newly introduced quality categories: trust, factuality and bias [5]. As it turned out, these categories are very hard to automatically classify with the methods that work really well for Web spam and DMOZ category classification. We propose a new text classification method based on extracting features via bi-clustering documents. We obtained the first results convincingly above 0.6 in AUC for the new categories, which is comparable to the first results of Web spam filtering about a decade ago. Beside the DC2010 data set we also evaluate our new methods on the Web Quality 2015 Data Challenger data set.

Chapter 2

Preliminaries

2.1 Content-based ranking

Content-based ranking is the component of search engines that stems from classical Information Retrieval, the multidisciplinary field of investigating models for selecting and ranking documents that match the given keywords. Search engines assign relevance scores to each document that express relevance of the document to the query. Both the documents and the queries are treated as a sequence of words. The query fed to the ranking algorithm sometimes differs from the original query typed by the user. Some words may be omitted, and other words may be added to the query. For example very frequent words (*the*, *a*, *and* etc.) are often removed from the query, since they are not discriminative enough and may cause noise in ranking. On the other hand, related words such as synonyms or inflections may be added to the query in order to find documents that contain other forms of the query word.

In the following Sections we describe the two most known content-based ranking schemes: TFIDF and Okapi BM25. In both schemes each document is represented by a sparse vector where each entry corresponds to a word. Only the entries corresponding to the words in the document are nonzero. Similarly, the queries are represented as a sparse vector, and the relevance is defined by the similarity between the query and the document vectors. The difference between the two schemes is how we define the value of a word in a document vector to express the importance of the corresponding word.

2.1.1 TF-IDF

First let us assume that the query contains a single term t. How can we measure the relevance of a document d for this query? The first obvious idea is to use

the number of occurrences of t. However this approach is satisfactory only if all documents have similar size. If a short and a long document both contain t the same number of times, then one can feel that the shorter one is likely more relevant than the longer one because the longer one may contain additional unrelated content.

By the above reasoning, we define *term frequency* as the number of occurrences normalized with the total number of words in the document. Let $N_{t,d}$ be the number of times term t appears in document d. The term frequency of t in d is

$$TF_{t,d} = \frac{N_{t,d}}{\sum_{i \in d} N_{i,d}}. \tag{2.1}$$

Now let us turn to multi-term queries. Assume that our query is "The Big Apple" and we use the sum of the three term frequencies as relevance score. It is obvious that the three terms in the query are not equally important. A document that contains words "The" and "Big" many times but "Apple" only once may be less relevant than the document containing "Apple" many times but with possibly less occurrences of "The" and "Big". Terms that appear in many documents represent less information about the topic of the documents they appear in. Rare terms are much more useful to express a specific subject.

Let us define the *document frequency* of term t as the fraction of the documents that contain t at least once. Let N_t be the document frequency of term t, let D be the set of documents and $|D|$ be the number of documents. Then we can express the importance of a query term by *inverse document frequency* as follows:

$$IDF_t = \log \frac{|D|}{N_t}. \tag{2.2}$$

Finally, the TF-IDF score of document d for query q is the combination of the term frequency and the inverse document frequency:

$$\text{TF-IDF}_{d,q} = \sum_{t \in d, t \in q} TF_{t,d} \cdot IDF_t. \tag{2.3}$$

2.1.2 Okapi BM25

BM25 is very similar to TF-IDF but it computes both the TF and the IDF part in a slightly different way. For IDF, IDF^{okapi} gives smaller weights to popular words than IDF, moreover it can be negative if a term appears in more than half of the documents:

$$IDF_t^{okapi} = \log \frac{|D| - N_t + 0.5}{N_t + 0.5}. \tag{2.4}$$

To avoid negative weights, the IDF^{okapi} function can be given a floor of 0, or if we don't want completely remove popular words, a floor of a constant ϵ.

In BM25, the term frequency measure is also changed to a bounded function that gives decreasing importance to additional occurrences of a term if it already appears several times in the document. Let $avgdl$ be the average length of documents in the corpus and $|d|$ be the length of document d. Then we define the BM25 relevance score of the query q to document d as:

$$BM25_{d,q} = \sum_{t \in d, t \in q} IDF_t^{okapi} \cdot \frac{N_{t,d} \cdot (k_1 + 1)}{N_{t,d} + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}, \tag{2.5}$$

where usually $k_1 \in [1.2, 2.0]$ and $b = 0.75$.

Contrary to TF-IDF, here we use the number of occurrences of t instead of TF in the following formula:

$$\frac{N_{t,d} \cdot (k_1 + 1)}{N_{t,d} + k_1}. \tag{2.6}$$

This means that in BM25, the number of occurrences of a term has higher weight in relevance, but its maximum value depends on k_1, since the above function converges to $k_1 + 1$. With parameter b, we can set the reward of the shorter and the penalty of the longer documents.

2.1.3 Location based term weighting

Different parts of the documents may have different importance, hence to compute the relevance of a query term, one may also take the location (title, URL, file name, keywords section) or the type (font size, boldface) of the occurrence into account. It may be reasonable to assign higher scores for query terms that appear in the title than for terms that only appear in the body. Higher weight can applied for words written in larger font size or words appearing at the beginning of the document.

Web pages have a special feature of key importance in Web information retrieval: Hyperlinks on the Web pages are usually have an associated *anchor text*, which is only few words long. This short text usually contains characteristic information about the pointed (and not the containing) page. Web search engines include the text of the hyperlinks into the content of the target page, not just the page that contains the link. Even more, the weight at the target page is much higher in general.

2.2 Link-based ranking

The link structure of the Web can be represented as a directed graph $G = (V, E)$ where V is the set of Web pages and E is the set of hyperlinks between pages: $(u, v) \in E$ if page u links to page v. From now on let $d^-(u)$ denote the in-degree and $d^+(u)$ the out-degree of node u, i.e. the number of pages that link to or pointed by node u.

2.2.1 PageRank

PageRank was invented by Brin and Page [75] and it is probably the best known link-based ranking algorithm. In this section we will define PageRank. The discussion starts with the description of simpler ranking algorithms in order to highlight the considerations that lead to the invention of PageRank.

Link-based ranking algorithms compute scores to the pages based on the intuition that the existence of an edge $(u; v)$ implies that the author of page u votes for the quality of v. A straightforward implementation of the above intuition is to rank the pages by their in-degree $d^-(v)$, the number of pages linking to v. This simple method has several drawbacks:

- All links are treated equally, thus one can easily create lots of dummy pages that link to a target page in order to increase its in-degree. In other words, in-degree is easily spammable.
- A single link from a popular page such as *cnn.com* should carry more value than even several links from low quality pages, i.e. the contribution of good pages should be higher.
- Pages with high out-degree have higher impact on the rank of other pages. Thus, the contribution of link from page u should be proportional to $1/d^+(u)$.

These considerations lead to the following recursive definition of page quality:

$$p(v) = \sum_{u:(u,v)\in E} p(u)/d^+(u), \tag{2.7}$$

i.e. a page is of high quality if it is pointed to by high quality pages.

However we still have some problems with the above equation. To reveal these problems, we consider the so-called *bow-tie* structure of the Web [15] shown in figure 2.1. We can differentiate the following types of Web components:

- *Strongly Connected Component (SCC):* the largest strongly connected subgraph of the Web.

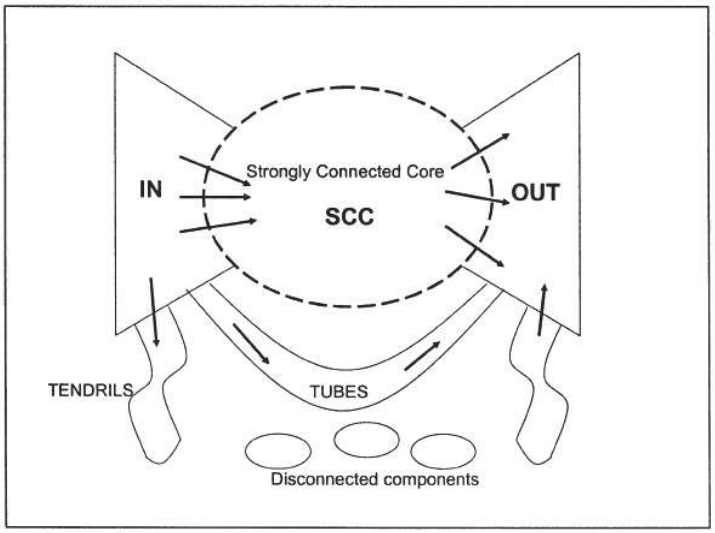


Figure 2.1: The bow-tie structure of the web.

- *The IN component:* consisting of nodes that can reach the SCC by following links, but are not reachable from the SCC.
- *The OUT component:* consisting of nodes from which the SCC is unreachable, but which themselves are reachable from the SCC.
- Other smaller parts that are not connected directly to the SCC: tendrils, tubes, disconnected components.

It is relatively easy to see that if we apply the recursive algorithm of Equation (2.7), the PageRank value of in-tendrils, the IN component and the SCC will all converge to 0, while the out-tendrils and the OUT component will sink all the available PageRank scores.

In order to overcome these difficulties, we add a complete graph with very small edge weights to the Web graph. We define the PageRank vector p with $p(i) \geq 0$ and $|p|_1 = 1$ as the solution of the following set of modified equations

$$p(v) = c \cdot r(v) + (1 - c) \cdot \sum_{u:(u,v)\in E} p(u)/d^+(u), \tag{2.8}$$

where $r = (r(v_1), \ldots, r(v_n))^T$ is the so-called teleportation or personalization vector with $r(v_i) \geq 0$, $|r|_1 = 1$ and c is the teleportation probability with a typical value of $c \approx 0.15$. If r is uniform, i.e. $r(v) = 1/n$ for all v, then p is the PageRank (PR).

In matrix notation, let A denote the adjacency matrix of the Web graph with normalized rows:

$$A_{(u,v)} = \begin{cases} 1/d^+(u) & \text{if page } u \text{ points to } v \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

Then equation (2.8) gives us

$$p = cr + (1-c)A^T p = M^T p. \tag{2.10}$$

In order to make Equation (2.10) meaningful, we have to ensure that there exists a unique solution. Notice that M describes the state transition matrix of a homogeneous Markov chain corresponding to the uniform independent random walk on the extended Web graph and p is the limit distribution of the Markov chain. It is a well-known fact from the theory of Markov chains that there exist a unique stationary distribution and thus a unique solution for the above system of equations, if the Markov chain is irreducible and aperiodic [68]. Moreover, the unique solution p is a probability distribution.

The matrix M is aperiodic, since nodes with $r(v) > 0$ have length one cycles with positive transition probability $M(vv) > 0$. If $r(v) > 0$ for all v in G, then all nodes are reachable from all nodes so all nodes form one single SCC, hence M is irreducible.

The idea of PageRank can explained through the random surfer model as well. The random surfer models a user who navigates through Web pages. He can perform two kinds of actions. Either he follows one of the hyperlinks of the current page or directly goes to a random page. He chooses the first and second action with probability $1-c$ and c. If he decided to follow one of the links, the link is chosen uniformly at random. In the other case, the target page v is selected with probability $r(v)$.

2.2.2 Personalized PageRank

If the teleportation probability vector r is uniform, we refer to it as uniform PageRank, otherwise as personalized PageRank. Unless otherwise stated, PageRank stands for uniform PageRank. To avoid misunderstanding, we will denote uniform PageRank by p and personalized PageRank by PPR(r) where r is the personalization vector. Uniform PageRank shows a general measure of goodness for each page. Personalized PageRank can introduce a bias towards a specific type of pages: e.g. r can be chosen such that $r(v)$ is positive for well known sports pages and 0 otherwise. This means that our random surfer will go to a random sports page if he gets bored with following the links. Pages that are reachable in a few steps from sports pages will get higher score. Other pages

that are not reachable in a few steps are unlikely related to sports and will get lower PageRank.

An important property of personalized PageRank is its linearity [59]. For any positive constants α_1, α_2 such that $\alpha_1 + \alpha_2 = 1$,

$$\text{PPR}(\alpha_1 r_1 + \alpha_2 r_2) = \alpha_1 \text{PPR}(r_1) + \alpha_2 \text{PPR}(r_2).$$

In an important special case, r consists of all 0 except for a single node v. Any personalized PageRank vector can be expressed as a linear combination of such 1-node personalized PageRank vectors:

$$PPR_i(r) = \sum_j r_j \cdot \text{PPR}_i(\chi_j), \tag{2.11}$$

where χ_j is the personalization vector consisting of all 0 except for node j where $\chi_j(j) = 1$.

In particular, uniform PageRank is the sum of n 1-node personalized PageRank vectors:

$$p_i = \frac{1}{N} \sum_j \text{PPR}_i(\chi_j) \tag{2.12}$$

By equation (2.12) we may say that the PageRank of page i arises as the contribution of personalization over certain pages v where $\text{PPR}_i(\chi_v)$ is high. We say that page v *supports* i to the above extent.

As noticed independently by [38, 59], the (personalized) PageRank of a vertex is equal to the probability of a random walk terminating at the given vertex where the length is from a geometric distribution: we terminate in step t with probability $c \cdot (1-c)^t$. To justify, notice that PageRank can be rewritten as a power series

$$\text{PPR}(r) = c \cdot \sum_{t=0}^{\infty} (1-c)^t r \cdot A^t. \tag{2.13}$$

By personalized PageRank, ranking can be tuned towards specific user interests. However, it is not easy to implement it in practice. Since the personalization vector can differ for each query, it would be very expensive to compute PageRank at query time.

2.2.3 HITS

Kleinberg [20] introduced an iterative ranking algorithm referred as *Kleinberg's algorithm* or *HITS*. HITS ranking is query-dependent, which implies that the computation must be fast as it is performed at query time. HITS computation is based on a topically focused small subgraph generated by the following steps:

- Acquire a seed set of search results S_q which are assumed to be relevant to the given query q. This could be the top n results of an existing search engine using text based ranking.
- Extend this seed set with other results that are likely to be relevant to the topic. Usually pages that are linked from S_q and pages that point to S_q are added. For high in-degree nodes, sampling can be applied to avoid adding too many pages.
- Build the vicinity graph G_q, the subgraph spanned by the extended seed set S'_q.

HITS computes two scores for each node in G_n. The *authority* score $a(v)$ reflects the relevance of the content of v to the query. The *hub* score $h(v)$ measures the quality of v as a link collection considering the given topic. The algorithm is based on the intuition that good hubs likely link to good authorities and good authorities appear in good hub pages. Thus, good hubs and authorities mutually reinforce each other.

In the iterative algorithm, let $h^{(k)}$ and $a^{(k)}$ denote the vectors of hub and authority scores. The first hub vector $h^{(0)}$ may contain arbitrary positive values.

The $(k+1)$-th authority score of v is computed from the previous hub vector $h(k)$ as the sum of the current hub scores of nodes pointing to v:

$$a^{(k+1)}(v) = \sum_{(u,v)\in E_q} h^{(k)}(u) \tag{2.14}$$

Similarly, the $(k+1)$-th hub score of v is the sum of the current authority scores of nodes pointed by v:

$$h^{(k+1)}(u) = \sum_{(u,v)\in E_q} a^{(k+1)}(v). \tag{2.15}$$

If A_q is the adjacency graph of G_q then the above equations can expressed in matrix form:

$$\begin{aligned} a^{(k+1)} &= A_q^T h^{(k)}, \\ h^{(k+1)} &= A_q a^{(k+1)}, \end{aligned}$$

whence

$$\begin{aligned} a^{(k+1)} &= A_q^T A_q a^{(k)}, \\ h^{(k+1)} &= A_q A_q^T h^{(k)}. \end{aligned}$$

It is shown in [20] that $h^{(k)}$ and $a^{(k)}$ can be normalized to converge to a unique h and a, the principal eigenvectors of $A^T A$ and AA^T, respectively. In practice, no more than a few hundred iterations suffice for convergence.

Unfortunately, HITS computation should be performed at query time, a disadvantage compared to PageRank. The computation includes fast subgraph construction, which requires random access to the Web graph.

2.3 Link-based similarity

The link-based similarity of nodes u and v in the Web graph is based on the neighboring link structure, usually the incoming links of u and v. One may assume that if two pages are pointed by similar other pages, then these two pages may be topically related.

2.3.1 Co-citation, Jaccard and cosine

The simplest and often the most effective link based similarity measure is *co-citation*, $\text{coc}(u, v)$ defined as the number of common in-neighbors of u and v. Co-citation may have directed and undirected variants. Co-citation is very easy to compute, but, similar to in-degree, it is vulnerable to spam. The Co-citation of any two pages, even if one is a page of high reputation and the other is a spam page, can be easily increased by creating pages that link to both of these pages.

The *Jaccard coefficient* is another metric that has the strength of emphasizing important connections and ignoring "accidental" unimportant connections. The Jaccard coefficient $\text{Jac}(u, v)$ is the ratio of common neighbors within all neighbors. The Jaccard coefficient has variants that use the reversed or undirected graphs. For a weighted graph we may divide the total weight to common neighbors by the total weight of edges from u and v. The measure performs poorly in the situation showed in Figure 2.2. Edges ux and vy have very high weights while uy and vx have low weights, since the Jaccard coefficient is high, while the actual similarity is very low.

Cosine similarity fixes the above problem of the Jaccard coefficient. Let us denote the row of the adjacency matrix corresponding to node u as vector $\overline{u}$. The cosine similarity of nodes u and v is simply

$$\cos(u, v) = \frac{\overline{u}^T \overline{v}}{||\overline{u}|| \cdot ||\overline{v}||}.$$

It has also transposed and undirected variants. Without normalization, cosine similarity would simply fall back to co-citation.

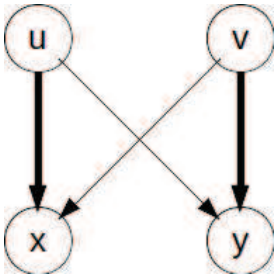


Figure 2.2: Example when Jaccard similarity performs poorly.

2.3.2 SimRank

SimRank was suggested by Jeh and Widom [58] as a measure of link-based similarity of two nodes in the graph. The basic idea is that two nodes are similar if they are pointed by similar nodes. SimRank is the iterative generalization of co-citation in the same way as PageRank generalizes in-degree.

SimRank is defined by the following equation: Initially

$$\text{Sim}^{(0)}(u_1, u_2) = \begin{cases} 1 & \text{if } u_1 = u_2 \\ 0 & \text{otherwise;} \end{cases} \tag{2.16}$$

then

$$\text{Sim}^{(i)}(u_1, u_2) = \begin{cases} (1-c) \cdot \frac{\sum \text{Sim}^{(i-1)}(v_1, v_2)}{d^-(u_1) \cdot d^-(u_2)} & \text{if } u_1 \neq u_2, \\ 1 & \text{if } u_1 = u_2. \end{cases} \tag{2.17}$$

where the summation is for all pairs $(v_1, u_1) \in E, (v_2, u_2) \in E$.

SimRank power iterations as in (2.17) are infeasible since they require quadratic space. We may use the algorithm of Sarlós et. al. from [89] instead.

2.4 Machine Learning

Machine learning is the field of designing and developing algorithms that "learn". In this case "learning" means that for a given task (which usually involves inference from data) the algorithm can improve its performance over time by using more data.

More formally, machine learning can be viewed as searching for some model that is a good approximation of an unknown function characterizing the system in question. We assume that there is a true function $y = f(x)$ which maps input x to output y. We do not know the real f but we have observations on the input

and sometimes on the output, the *training data* as well. Learning is the process of searching for an f' that matches the observations as good as possible. The function f' is often referred as the *model* and the set of functions F from which it is selected as the *hypothesis space*.

Machine learning methods can be categorized into three types based on the nature and the usage of the training data.

- In *supervised* learning, a set of examples with the "right" answer is given. For each input x in the training data, we know the correct output $y = f(x)$. The task is to predict the answer for future input. The model is finalized after processing the training data and not modified during the prediction for unseen data.

- In *unsupervised* learning, no "right" answer is given. The task is to explain the data in terms of a few classes or parameters. The algorithm has to discover how the data is organized and identify the classes. The best known form of unsupervised learning is *clustering* (see Section 2.6).

- *Semi-supervised* learning is the hybrid of supervised and unsupervised learning. A large set of data is given but the "right" answer is known for only a small fraction of them. The methods usually identifies certain regularity in the unlabeled data and uses the labeled data to leverage on that. Results show that using the unlabeled data can improve the learning accuracy. By using semi-supervised learning, the expensive production of labeled data can be reduced.

We shortly overview the machine learning techniques used in this thesis.

2.4.1 Base learners

Naive Bayes

The Naive Bayes classifier [56] is a generative approach based on Bayesian inference. Given a sample $x_k = (x_k^1, ..., x_k^m)$ and a class H_i, the Naive Bayes method computes the posterior probability of x_k belonging to H_i as

$$P(H_i|x_k^1, ..., x_k^m) = \frac{P(x_k^1, ..., x_k^m|H_i)P(H_i)}{P(x_k^1, ..., x_k^m)}. \tag{2.18}$$

Now for each x_i we want to determine the class H_i with the highest probability:

$$P(H_i|x_k) > P(H_l|x_k) \text{ for all } i : j, i \neq j. \tag{2.19}$$

Since the denominator in Equation (2.18) is constant, this reduces to

$$P(H_i|x_k^1, ..., x_k^m) = Z \cdot P(x_k^1, ..., x_k^m|H_i)P(H_i), \tag{2.20}$$

where Z is a constant independent of H_i.

Naive Bayes assumes all the features $x_k^1, ..., x_k^m$ are independent, so the conditional distribution can be computed as

$$P(H_i|x_k^1, ..., x_k^m) = Z \cdot P(H_i) \prod_{j=1} P(x_k^j|H_i). \tag{2.21}$$

A class label $H^* = H_i$ is assigned to each sample x_k with a decision rule that picks the most probable hypothesis:

$$i = \text{argmax} P(H_i) \prod_{j=1} P(x_k^j|H_i). \tag{2.22}$$

Naive Bayes is a quite simple and fast method. The model can be built in time linear in the size of the the training set. It works quite well for classifying text documents using words as features.

Decision Trees

Decision Trees are classifiers [84] presented in the form of binary trees where each node corresponds to a variable and edges represent possible realization of that variable. Given a sample $x_k = (x_k^1, ..., x_k^m)$, leaf nodes correspond to a possible class H. The main goal of a Decision Tree is to build class hypotheses based on the observed attributes of the training data. The output dichotomic decision tree can be used to determine the class label of an unclassified sample by considering its descriptive attribute realizations. Building a decision tree model from a training data set involves two phases. In the first phase, a splitting attribute and a split index are chosen. The second phase involves splitting the records among the child nodes based on the decision made in the first phase. This process is recursively continued until a stopping criterion is satisfied. Classification quality depends on the choice of the variable ordering (from the root to the leaf) and the values for the splitting rule. Two of the most widely used indexes for evaluating whether a node should be split or not are the *Gini Index* and the *Entropy*. Given a node j with hypothetical realizations t, the Gini Index is defined as

$$I_G = 1 - \sum_t f(j,t)^2,$$

while the Entropy is defined as

$$I_E = -\sum_t f(j,t) \log f(j,t),$$

where $f(j,t)$ represents the frequency of value t in node j.

Support Vector Machines

Support Vector Machines (SVM) [27] are linear learning techniques aimed at determining the optimal hyperplane that discriminates samples of different classes. SVM classifiers are in general considered especially effective for text classification.

Given a training set defined over the input space X and binary class labels Y defined by

$$(x_1, y_1), ..., (x_l, y_l) \in (X \times Y),$$

Support Vector Machines find the optimal hyperplane $O = \{x \in R^n : w^T x + b = 0\}$ with the maximum margin, i.e. with the maximum distance between class samples. When samples are not linearly separable, features are transformed into a higher dimensional space by a kernel function Φ that allow the samples to be separated.

The optimal hyperplane O is defined by learning two parameters from the data: the weight vector w and the bias b. Let S_1 and S_2 be two disjoint subsets of samples defined over the feature space F, where samples $(x_k, y_k) \in S_1$ have $y_k = -1$ and samples $(x_p, y_p) \in S_2$ have $y_p = +1$. For any sample x_t, since the maximum margin depends on w, the classification problem can be formulated as

$$\min \frac{1}{2}||w||^2 \quad \text{such that} \quad y_t(w^T x_t + b) - 1 \geq 0.$$

Linear regression

In linear regression we assume that the hypothesis space is linear. In contrast to the previous methods, linear regression does not classify the elements, but predicts or forecasts the value of the output.

Let our training set be $X \in R^{n \times k}$ and the output value for the training elements be $y \in R^k$. We are looking for the vector $w \in R^n$ and the bias $w_0 \in R^k$ that approximates the training data the best by the formula

$$X^T w + w_0 = y.$$

2.4.2 Ensemble learning

An ensemble learner [66] consists of multiple base learners to obtain stronger predictive performance than any of the individual algorithms.

Bagging

Bagging is designed to improve stability and accuracy of the base learner(s). It reduces variance and overfitting. Given a standard training set D, bagging generates new training sets D_i by taking a sample with uniform probability and with replacement from D. For each new training set D_i, a model is built by the base learner(s). The predictions of the models are averaged on the test set. As an example, the Random Forest algorithm [14] combines random decision trees with bagging.

Boosting

Boosting algorithms apply base learners iteratively. Every time a base learner is added, the elements of the training set are re-weighted. Elements that are misclassified gain weight and elements that are classified correctly lose weight. Thus, next base learners focus more on previously misclassified elements. Different types of boosting algorithms use different re-weighting schemes. The most popular boosting algorithms are AdaBoost [41] and LogitBoost [42].

Ensemble Selection

Bagging and Boosting techniques modify the training data. In contrast, Ensemble Selection methods weight the base learners used in the ensemble, based on their performance on the training set (cross-validation) or on a validation set. Various methods exist for defining the weight of the learning algorithms, including Bucket of models, Stacking, and Bayesian model combination. The following procedure is implemented in the Weka classification framework [101]: for initializing the ensemble, we choose the best k learning algorithms on the validation set. Then in every step, we choose the learning algorithm that adds the most to the current ensemble in performance on the validation set. Every learning algorithm can be chosen several times and performance can be evaluated by an arbitrary predefined quality measure.

2.4.3 Feature selection

Feature selection or attribute selection is a method to select a subset of features that are used for model building. Feature selection can filter redundant and irrelevant features. At the same time, it reduces both training time and overfitting. Feature selection techniques can be divided into two main groups: feature ranking and feature subset selection. The former tries to rank and select attributes based on their individual predictive power evaluated on the training set by statistical or information theoretic measures like correlation. The latter tries to select

a subset of attributes that together have good predictive power. These methods are usually initialized with an empty set of attributes or with all of the attributes and they are iteratively add or remove attributes, based on some heuristics.

2.5 Classification quality measures

In this section we describe the evaluation measures used in this thesis.

2.5.1 Precision and Recall

Assume we have a classifier C and a test set of documents D. Classifier C computes a label for each document $d \in D$. Let $C(d)$ denote the classified label (1 or 0) and $O(p)$ the original label. Let $S \subset D$ and $N \subset D$ denote the set of pages with original label 1 and 0, respectively ($S \cup N = D$). Let S_C and N_C denote the set of pages that are labeled as 1 and 0 by the classifier. In summary,

$$S = \{p \in D | O(p) = 1\},$$

$$N = \{p \in D | O(p) = 0\},$$

$$S_C = \{p \in D | C(p) = 1\},$$

$$N_C = \{p \in D | C(p) = 0\}.$$

Let us define the following metrics:

- *True positives* are the correctly detected documents where $O(d) = 1$, $TP_C = S \cap S_C$.
- *True negatives* are the correctly detected documents where $O(d) = 0$, $TN_C = N \cap N_C$.
- *False positives* are the incorrectly detected documents where $O(d) = 0$ and $C(d) = 1$, $FP_C = N \cap S_C$.
- *False negatives* are the incorrectly detected documents where $O(d) = 1$ and $C(d) = 0$, $FN_C = S \cap N_C$.

The two-by-two matrix of these values forms the *confusion matrix*:

	predicted positive	predicted negative
positive	true positives	true negatives
negative	false positives	false negatives

Precision is the fraction of correctly detected documents in the set of documents where $C(d) = 1$:

$$P = \frac{TP_C}{TP_C + FP_C}.$$

Recall is the fraction of correctly detected documents in the set of documents where $O(d) = 1$:

$$R = \frac{TP_C}{TP_C + FP_C}.$$

The *F-measure* is a frequently used combination of precision and recall as their harmonic mean,

$$F = \frac{2P_C R_C}{P_C + R_C}.$$

In most cases the classifier output is not binary but a predicted value or a probability. In this case we can apply a threshold on this value and label a page 1 if its value is higher than the threshold and 0 otherwise. Let us denote such an algorithm by C_Θ:

$$C_\Theta(p) = \begin{cases} 1 & \text{if } C(d) > \Theta, \\ 0 & \text{otherwise.} \end{cases}$$

For different Θ, the performance of the algorithm will be different.

Assume that $0 < C(d) \leq 1$ for all $d \in D$ and there is only one $q \in D$ with $C(q) = 1$. In the extreme case of $\Theta = 0$, C_0 will label every document 1, so its precision is $P_{C_0} = |S|/|D|$ and its recall is $R_{C_0} = 1$.

In the other extreme case, C_1 will classify document q as 1 and all others 0. If $q \in S$, then precision is $P_{C_1} = 1$ and recall is $R_{C_1} = 1/|S|$.

The reasonable value of Θ is somewhere between these two cases. However, it is not clear which Θ is the best. We can select the one where the corresponding F-measure is the highest, but we may prefer higher precision to higher recall or vice versa based on our needs. For example in case of a Web search engine a typical user would prefer every result to be relevant on the first page but isn't interested in all the relevant documents of the Web, therefor precision is more important than recall. On the other hand, in medical diagnostic applications a false negative result can have severe consequences meanwhile a false positive result only entail some unnecessary examinations, so in this case recall is more important than precision.

2.5.2 ROC curve

Besides the precision-recall curve of the previous subsection, another way to visualize the threshold-depended performance is the *ROC curve*. The name ROC

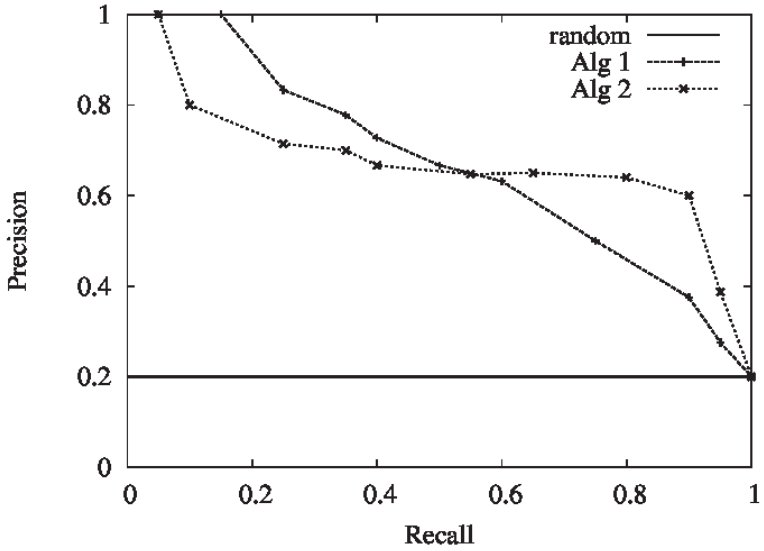


Figure 2.3: Precision and recall of two example algorithms.

originated in signal detection theory and it stands for "Receiver Operating Characteristic". The ROC curve depicts the *false positive rate*, $FPR_C = FP_C/N$, as the function of recall (also called *true positive* rate or *sensitivity*). Figure 2.4 shows the ROC-curves of the algorithms of Figure 2.3.

The *area under the ROC-curve* (AUC) is a frequently used classifier performance indicator. Obviously $0 \leq AUC \leq 1$. If $AUC = 1$, then the algorithm works perfectly for all possible thresholds. For the random algorithm, $AUC_R = 0.5$. If $AUC \leq 0.5$, the curve is below the random line and the algorithm can be improved to reach at least $0.5 - AUC$ by inverting its decision. It is shown in [35] that for randomly selected $p \in S$ and $q \in N$, AUC_C is the probability of classifier C output a pair of predictions with $C(p) > C(q)$.

2.5.3 NDCG

Normalized Discounted Cumulative Gain [57] is a commonly used measurement in information retrieval to evaluate the quality of search engines ranking based on the graded relevance of the ranked documents. NDCG makes the assumption that highly relevant documents are more useful when appearing earlier in the

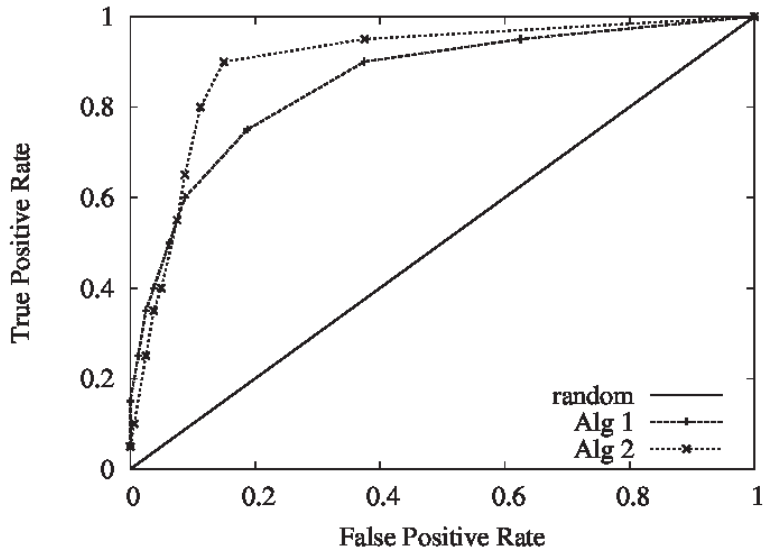


Figure 2.4: ROC curve of the two example algorithms.

result list and that the beginning of the result list is more important than the end:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)},$$

where rel_i is the graded relevance value of the ith ranked document. It is easy to see that the maximum value of DCG can be achieved by sorting the documents by their graded relevance value. The maximum DCG is called Ideal DCG (IDCG). If we want to compare the DCG value of result lists with different length or for different queries, normalization is needed:

$$NDCG_p = \frac{DCG_p}{IDCG_p}.$$

2.5.4 MAE, RMSE

The most common measures for evaluating regression methods is the Mean Absolute Error and the Root Mean Square Error. Let the predicted values for our

test set be f_i and the original values be y_i. Then

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$$

and

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - y_i)^2}$$

We can see that RMSE gives more penalty for larger errors than MAE.

2.6 Clustering

Clustering is an unsupervised machine learning technique that aims to group a set of objects in such a way that objects in the similar cluster (group) are more similar in some sense to each other than to those in other clusters. There is no unique definition for the notion of a cluster, which is one of the reasons why there are so many clustering algorithms. However the different clustering algorithms can be grouped into cluster models. In this thesis we use only one type of clustering algorithms described in Section 7.2. Here we tangibly describe the most prominent clustering models [104].

2.6.1 Centroid based clustering

The centroid method starts with a given number of clusters, which could be random or generated by some heuristics. Then it refines the clusters iteratively based on an objective function. In every iteration it recomputes the centroid of the clusters and moves every element into the cluster with the closest centroid.

The centroid models always find a local optimum. They are sensible for the initial cluster assignment. The most well known centroid algorithms are k-means and k-medoids.

2.6.2 Hierarchical clustering

The hierarchical methods represent clusters in a hierarchical tree structure called dendrogram. We can differentiate two types of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). The agglomerative models start with every element in a separate cluster and in every iteration they merge the two closest clusters until every element is in one cluster. In contrast, the divisive models start with every element in the same cluster and split recursively as

they move down the hierarchy. The most well known hierarchical algorithms are SLINK and CLINK.

2.6.3 Density-based clustering

Density-based algorithms define clusters as areas of density higher than the areas between the clusters. The most common density-based algorithm is DBSCAN. DBSCAN requires two parameters: ϵ and minPts. DBSCAN finds clusters where every point in the cluster has at least minPts number of points in an ϵ radius. The advantage of DBSCAN is that it is deterministic and does not depend on the initial configurations. The drawback is that DBSCAN assumes clusters of similar density. This problem is solved by the OPTICS algorithm, a generalization of DBSCAN. However both algorithms have problems with separating nearby clusters.

2.6.4 Distribution based clustering

The most popular distribution based clustering method is the Gaussian mixture model. This algorithm starts with a fixed number of Gaussian distributions. In every iteration, for every pair of elements and distributions, we compute the probability that the element is from the given distribution. Finally we refine the parameters of every distribution to fit better to the elements that are likely to come from the given distribution.

2.7 Information theory

Information theory was developed by Claude E. Shannon [92] to find fundamental limits on signal processing operations such as compressing, storing and communicating data. Since its inception, information theory has been applied to many other fields including linguistics, cryptography, and physics.

2.7.1 Entropy

A key measure of information is entropy, the amount of information in a random variable. It indicates how easily a message can be compressed or in other words, it expresses the average number of bits needed to store or communicate one symbol in a message. For example, specifying the outcome of a coin flip has lower entropy than specifying the outcome from a roll of a die. Let X denote a

random discrete variable. Then the entropy of X is computed as

$$H(X) = -\sum_{x \in X} p(x) \cdot log(p(x)). \tag{2.23}$$

2.7.2 Mutual Information

Mutual information measures the amount of information that can be obtained about one random variable by observing another. Let X and Y denote random discrete variables. Then mutual information of X and Y is computed as

$$I(X,Y) = \sum_{x \in X, y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}. \tag{2.24}$$

2.7.3 Kullback-Leibler divergence

Kullback-Leibler divergence (KL) is an asymmetric measure of the difference of two probability distributions. The KL of two probability distributions p and q is

$$KL(P \parallel Q) = \sum_{x} P(x) log \frac{P(x)}{Q(x)}. \tag{2.25}$$

The above formula is invalid if $Q(x) = 0$. The definition of KL is extended to the cases when $Q(x) = 0$ implies $P(x) = 0$. In this case, the expression $0 \cdot log(0)$ is interpreted as 0, since $\lim_{x \to 0} x log(x) = 0$.

It is easy to prove the following equation:

$$KL(P \parallel Q) = \sum_{x} P(x) log P(x) - \sum_{x} P(x) log Q(x) = H(P,Q) - H(P). \tag{2.26}$$

In other words, the Kullback-Leibler divergence measures the loss of information if we approximate P with Q.

2.7.4 Jensen-Shannon divergence

Jensen-Shannon divergence is the symmetric, smoothed version of Kullback-Leibler divergence, also known as total divergence to the average.

$$JS(P \parallel Q) = \frac{1}{2}\big(KL(P \parallel M) + KL(Q \parallel M)\big), \tag{2.27}$$

where

$$M = \frac{1}{2}(P + Q).$$

In contrast to KL divergence, JS is a metric, hence it makes more sense to use it as a distance measure.

2.8 Notes

A part of this chapter incorporates the basis of joint works with Károly Csalogány in the papers [9] [2]. He summarized the terminology in his thesis "Methods for Web Spam Filtering". I extended part of his descriptions for the specific needs of my results; for more information we refer the reader to his thesis[1].

[1] http://www.inf.elte.hu/karunkrol/szolgaltatasok/konyvtar/Lists/Doktori%20disszertcik%20adatbzisa/Attachments/74/Csalogany_Karoly_Ertekezes.pdf

Chapter 3

Web Quality

In the following sections I introduce the 3 main Web quality categories that this thesis examines and I discuss why we think these categories are considered to have an important role in assessing the quality of Web pages. I also give a brief overview of the common methods that are used for Web classification. Since the basis of Web classification is supervised machine learning and creating a good quality training set for quality classification is very expensive, here I collect and compare the few data sets that are commonly used for experiments in this field.

3.1 Spam

In the last decades, Web has grown to be a central part of our lives. Every day, millions of people use the Web for finding information, socializing, shopping goods, booking hotels or flights, etc. Due to the immense size of the Web, users generally locate the desired Web pages through search engines, which practically means that a very large portion of web traffic can be originated from search engines.

For many Web pages, increased traffic translates to increased profit. According to the statistics of eMarketer[1], an independent market research company, online business-to-consumer sales grew 21.1% and reached $1 trillion in 2012. In order to increase web traffic, it is essential for commercial web pages to appear in the top results of search engines. To acquire these top positions, webmasters design the content and link structure of their pages in a way to influence the ranking algorithms of search engines for certain queries. These techniques are called Search Engine Optimization (SEO).

[1] http://www.emarketer.com

We can distinguish white-hat SEO and black-hat SEO. White-hat SEO improves the quality of the content and the structure of the Web site and it is useful for the users. On the other hand black-hat SEO decreases the quality of search engines and misleads users by putting irrelevant pages in top positions for certain query terms. Pages created by black-hat SEO are also called Web spam pages. Due to the large and ever increasing financial gains resulting from top search engine positions, it is not surprising that spammers are devoting a huge effort to influence the ranking algorithms of search engines.

3.1.1 Types of Web spam

We can categorize Web spamming techniques based on which component of the search engine ranking algorithms are they targeting. Content spam manipulates the textual content of web pages, meanwhile link spam is the creation of special link structures between pages.

Content spam

Content based ranking methods apply the simple idea that a page is relevant to a query if it contains the terms of the query. The more the query terms occur, the more relevant the page is (see Section 2.1). This motivates spammers to create pages with a large number of different terms or to repeat specific popular query terms several times. According to Gyöngyi et al. [52], the most common content spamming techniques are:

- *Repetition of specific terms.* This way spammers increase the relevance of the document for a small number of specific query terms.
- *Dumping a large number of unrelated terms.* Sometimes spammers copy whole dictionaries to get a certain amount of relevance to many different queries.
- *Weaving of spam terms into copied content.* Spammers can copy high quality content available on the Web and insert spam terms at random positions. It is effective when the topic of the original text is so rare that only a few relevant pages exist.
- *Phrase stitching.* This technique is very similar to weaving but text is copied from different sources, which makes the page relevant to more queries while at the same time making detection more difficult.

We can also distinguish different types of content spam based on the location in the Web page:

- *Title, meta, URL spam.* Search engines usually give higher weights for terms that appear in the title and meta tags or in the URL of an HTML page. So it makes sense for spammers to put spam terms in the title or to create very long URLs.

- *Anchor spam.* Anchor text is the visible, clickable text in a hyperlink. Just like title text, anchor text also gets higher weights but contrary to title and URL, anchor text increases the relevance of the target page of the link, since it is usually a short summary of the referred page. Therefore, spam terms are sometimes included in the anchor text of referring pages.

Link spam

Beside content based ranking algorithms, search engines also use link based algorithms like PageRank and HITS (see Section 3.1.1) to evaluate the importance of Web pages. Hence, spammers often try to influence the relevance scores of their pages by creating special link structures between them. We can differentiate three types of Web pages from the aspect of a black hat SEO targeting at spamming the Web link structure:

1. *Inaccessible.* These are pages that cannot be controlled by spammers, however they can still insert links on their own pages pointing to inaccessible pages.

2. *Accessible pages.* These pages are owned by others but it is still possible to modify the content of them. For example a spammer can modify and insert some links on a wiki page, in a comment on a blog page, in a review on a webshop page, etc.

3. *Own pages.* These pages are maintained by the spammer and hence, has full control over the content of these pages.

Link spammers have two choices to increase the importance of their pages. The first one is adding outgoing links to popular Web pages to increase the hub score in the HITS algorithm. This strategy is quite easy, since spammers have full control over their own pages. The second possibility is gathering incoming links from popular Web pages, which can increase both HITS authority and PageRank scores. This is a little bit trickier but feasible with the help of accessible pages and other sophisticated methods. According to Gyöngyi et al. [52], the most wide-spread link spamming techniques are:

- *Directory cloning.* An easy way to place a huge number of outgoing links on a Web site is to copy large existing directories from the Web like Yahoo! Directory `dir.yahoo.com` or the Open Directory Project `dmoz.org`.

- *Honey pot.* A honey pot is a collection of pages with high quality content that links to spam pages. Useful looking content perhaps copied from authoritative sources can mislead users and can gather some incoming links from inaccessible pages. Note that the previously mentioned directory cloning is a good technique for creating honey pots.

- *Social spam.* As mentioned before, spammers can easily insert links into blog comments, wiki pages, product reviews, guest books that point to their own pages. Even if the social media site is moderated, sometimes it is quite hard to identify a well placed spam link.

- *Expired domains.* Spammers can buy expired domains and steal some PageRank from inaccessible pages through the still living incoming links to the expired domain.

- *Link farms.* Since these days maintaining a Web site is very affordable, spammers can control a large number of sites for a very low budget. They can combine all the above techniques and create a massive link structure between their sites to boost the relevance for only a number of target pages.

3.2 The Open Directory Project: DMOZ

A *web directory* or *link directory* is a special directory on the Web that lists Web sites organized into categories based on their topic. The purpose of web directories is to filter the Web from useless content and to maintain a listing of high quality Web sites. Most of the web directories allow site owners to submit their sites with a limited number of suggested categories. These submissions are reviewed and evaluated by the editors of the directory. Users of web directories can query the categories of a web site or they can browse the categories and find a list of web sites with a very specific topic.

The largest and most well known web directory is the Open Directory Project, also known as the DMOZ (**d**irectory.**moz**illa.org) web directory. Submitting a site to DMOZ or downloading their data set is publicly available and free. DMOZ is owned by AOL but it is maintained by a community of 90 thousand volunteer editors. The DMOZ database contains more than 4 million sites in different languages and about 1 million categories.

3.2.1 Maintenance

The majority of editors are maintaining only a few number of categories. All new editors who join the community edit only one category. As they get some editing experience, they can apply for editing additional categories. Editors are evaluating new submissions as well as checking the status of already listed sites. Editors have a very thorough and detailed guideline to evaluate the categories of submitted sites and to decide whether its content fits the quality expectations of DMOZ. Editors have a bunch of automatic tools which helps them in the evaluation process like link and spell checkers and the Robozilla Web crawler which periodically check the status of listed sites. The activity of editors is reviewed by meta editors. They are the managers of the community. They review new editor applications, new category requests, complains against editors.

3.2.2 Category hierarchy

DMOZ uses a hierarchical ontology scheme for organizing site listings. In most cases, sites are grouped together based on their topics, but language and regional categories are also exist. If a category contains too many sites, then it is divided into smaller subcategories. However DMOZ ontology is more complex than a simple tree. Some categories can be subdivided based on multiple criteria. The *business* category for example has subcategories based on the types of the organizations (cooperatives, small businesses, major companies, etc) and also based on business areas (automotive, health care, telecom, etc). Furthermore a site can appear in more than one category and categories can have various types of cross-reference links between them and even cycles are present. In this thesis we focus only on the largest top level categories (see Figure 3.1).

3.3 Other aspects of quality

In Section 3.1 we showed that we can categorize the web pages by their spamicity. However just because a web page is not spam does not mean that it has high quality content. Web quality has various factors. Some of them are objective, like spamicity and how well designed the content of a web page is, but some of them are very subjective, like the importance of the topic of the web page. Assessing the quality of web pages is more like prioritizing different factors depending on the person or organization who collects information from the Web. For example, if we are gathering news from the web, then we are not interested in educational or commercial sites. Moreover we are probably not interested in satire and fake news either. However satire news sites are not spam sites, and if

Figure 3.1: Main categories of DMOZ.

the content is well designed, they can be listed in the DMOZ web directory in the news category.

From the above example we can see that more factors are needed beyond spam and genre to determine the quality of web pages. In the Discovery Challenge 2010 (see Section 3.4.2), three new quality measures were introduced which are orthogonal to genre:

- *Trustworthiness.* How authoritative is the source of the Web site.
- *Neutrality.* Neutrality or factuality is based on the ratio of facts and opinions.
- *Biased.* The definition was adapted from Wikipedia. Flame, assault and dishonest opinion without facts were flagged as Biased.

3.4 Assessing the quality of Web

Large scale Web classification techniques are based on supervised machine learning algorithms (Section 2.4). Thus we can divide the basic classification methods into three steps:

1. Extract features from Web pages.
2. Compile a large manually labeled training set.
3. Use the features and the labeled training set as an input for a suitable machine learning method.

In the following sections we briefly overview the basic feature sets (Section 3.4.1), the existing labeled data sets (Section 3.4.2) and also the most effective machine learning methods for web classification (Section 3.4.3).

3.4.1 Feature sets

For baseline we can use the so called "public feature set" that can be originated from early Web spam filtering techniques. In [37], Fetterly et al. propose that certain kind of spam pages can be identified through statistical analysis. They examined several features generated from the content and linkage of Web pages and found that the outliers in the statistical distributions tend to be spam pages. Later on several authors [6, 16, 72] suggested additional link and content features that can be used as an input for machine learning methods. Most of these features were included in the WEBSPAM-UK-2006 data set and were made public for the Web Spam challenge [18] and for the Discovery challenge contests [10] where several teams used these feature sets as an input for different classification methods. We can differentiate three types of feature set: content, link and term features. In the next sections we will give a short overview of these features.

Content features

1. *Number of words in the page, in the title.*
2. *Average length of words.*
3. *Fraction of anchor text.*
4. *Fraction of visible text.*
5. *Compression rate.*
6. *Corpus precision, recall.* Precision and recall of the words of the page to the top 100, 200, 500, 1000 most popular words in the corpus.
7. *Query precision, recall.* Precision and recall of the words of the page to the top 100, 200, 500, 1000 most globally popular words. Usually a query log is used to determine the most globally popular words.

8. *Entropy of n-grams.* Where an n-gram is n consecutive words. Usually n is set to 3.

9. *Independent n-gram likelihoods.* It is defined as $-\frac{1}{|N|}\sum_{t\in N}\log P(n)$ where N is the set of n-grams and $P(n)$ is the probability of n in the page.

Link features

1. In-degree, out-degree.

2. Average in-degree, average out-degree.

3. *Assortativity.* The ratio of the degree of the page and the average degree of its neighbors.

4. Average in-degree of out-neighbors, average out-degree of in-neighbors.

5. Number of neighbors and supporters at distance 2, 3, 4.

6. *PageRank.* See Section 2.2.1

7. Standard deviation of the PageRank of in-neighbors.

8. Truncated PageRank at distance 1, 2, 3, 4.

9. *TrustRank.* Personalized PageRank from a trusted set of seed. See Section 2.2.2.

10. Numeric transformations of the above features. These transformations were found to work better for classification in practice than the raw link-based features. This includes mostly ratios between features such as in-degree/PageRank or TrustRank/PageRank, and the logarithm of several features.

Bag of words features

A well-known method for document classification is to use term frequencies as a feature vector. The method consists of three steps:

1. *Feature selection.* We throw away unimportant words from the corpus, usually those that appear in nearly all of the documents (stop words) and those that appear only in a few of the documents.

2. *Weighting.* The weight of a word is intended to reflect its importance to a document in the given corpus. For this purpose, usually a TF-IDF based weighting is used (see Section 2.1).

3. *Classification.* The feature vectors are used as input to machine learning algorithms.

3.4.2 Data sets

WEBSPAM-UK data sets

The WEBSPAM-UK2006 and WEBSPAM-UK2007 data sets were compiled by Castillo et al. [18] for the Web Spam Challenges. These data sets are based on crawls of the .uk Web domain done in May 2006 and May 2007 [13]. The first crawl consists of 77.9 million pages and over 3 billion links from 11400 hosts, the second crawl includes 105.9 million pages and over 3.7 billion links from 114529 hosts. The hosts were labeled as spam, non-spam and borderline by volunteers who work in the areas of Web mining or information retrieval. In our experiments we used only the subset of the labels where there was agreement between the raters and we extended the training set with non-spam domains using the DMOZ directory. Table 3.1 shows the distribution of spam hosts over the labeled hosts.

ClueWeb09 data set

The ClueWeb09 data set was created by the Language Technology Institute at Carnegie Mellon University to support research on information retrieval. The data set consists of 1 billion web pages in 10 languages, collected in January and February 2009. The English corpus consists of 20 million domains and 500 million pages. Spam labels were provided as part of the Waterloo Spam Rankings [25].

Portuguese data set

The .pt data set was crawled and labeled by the Portuguese Web Archive in 2009. The data set consists of 600 thousand domains and 70 million pages. The original labels include only positive instances of spam hosts that we have extended with negative instances from the DMOZ web directory.

	UK2006	UK2007	ClueWeb09	.pt	DC2010
hosts	11400	114529	500000	600000	190000
pages	77.9M	105.9M	1B	70M	23M
spam	124	439	439	124	423
non-spam	3375	8421	8421	3375	4982

Table 3.1: Comparison of the distribution of spam/non-spam labels on different data sets.

DC2010 data set

The Discovery Challenge 2010 data set is a multilingual corpus from the .eu domain crawled by the Internet Memory Foundation in 2010 and annotated by the Hungarian Academy of Sciences (English documents), Internet Memory Foundation (French) and L3S Hannover (German) [10]. The data set consists of 190 thousand hosts and 23 million pages. In addition to labeling hosts as spam and non-spam, DC2010 includes additional five categories based on genre: Editorial, Commercial, Educational, Discussion and Personal; as well as three new quality categories: trust, factuality and bias. For more details about the additional categories, see Section 7.4.1.

C3 data set

The C3 data set was released as part of the Web Quality 2015 Data Challenger. The data set was created in the Reconcile[2] project and contains 22325 evaluations (five dimensions, among them credibility) of 5704 pages given by 2499 people. It also contains some additional information about website characteristics and basic demographic features of users.

3.4.3 Classification methods

The area of the Adversarial Information Retrieval attracted a large number of researchers for yearly workshops and a number of data challenges. The AIRWeb workshop series ran for five years [36], and evaluation campaigns included the Web Spam Challenges [17] and the ECML/PKDD Discovery Challenge 2010 [34]. While Web Spam Challenges focused only on detecting Web spam, the Discovery Challenge extended the scope by introducing labels for genre and quality. The baseline classification methods can be collected by analyzing the results [24, 1, 43] of these challenges.

[2] `http://reconcile.pjwstk.edu.pl/`

A key ingredient of the Web Spam Challenge 2008 best result [43] was ensemble undersampling. They extended the "public" content- and link-features by custom host-graph features based on PageRank, TrustRank and Truncated PageRank. To handle class imbalances, they used bagging with Ensemble Random Under-Sampling Strategy (ERUS) over a C4.5 decision tree. Other strong results were achieved by using a semi-supervised version of SVM [1] or random forest [96] (see Section 2.4). Best results either used bag of words representations or the "public" features sets or both.

The Discovery Challenge 2010 best results show that variants of bag of words representation are very strong and in contrary to Web Spam Challenges there is only a little use of content- and link-based features. A possible reason is that the DC2010 training and test set were constructed in such a way that no IP and domain was allowed to be split between them. Best results applied a wide selection of classification techniques including decision trees, random forest, SVM, boosting, bagging, oversampling, ensemble selection. It also turned out that in categories with high class imbalance feature selection methods like Fisher, Wilcoxon, Information Gain, works very well [47, 4, 71].

The other lesson learned from DC2010 is the hardness of the new quality tasks (neutrality, bias, trust). While the best results achieved an average AUC of 0.8 on genre categories and 0.83 on spam they produced nearly random results on quality categories.

Chapter 4

Graph stacking

Semi-supervised learning, a class of machine learning surveyed e.g. in [106], also exploits information from unlabeled besides labeled data during the learning phase. We focus on the applicability of classifying Web spam and telephone churn, i.e. users who cancel their telephone line subscription. Both Web hosts and telephone users can be represented as a graph (see Section 2.2), where in case of Web the nodes are the hosts and the edges are links between the hosts, while in the case of telephone churn the nodes are the users and the edges are the calls between users. Our assumption is that the label (spam and churn, respectively) of a node in this graph is similar to those of its neighbors.

We compare various means of stacked graphical learning, a meta-learning scheme in which a base learner is augmented by expanding the features of one node with predictions on other related nodes in a graph. This class of algorithms is introduced by Kou and Cohen in [61]. The methodology is used with success for Web spam detection in [16]: they use the average label of the neighbors as a new feature for the classifier.

We run our tests on the WEBSPAM-UK2006 data set. The baseline decision tree utilized all graph based features related to a node (i.e. features related to the "home page" or the "maximum PageRank node within site" are not computed) [16] and a Naive Bayes classifier of the machine learning toolkit Weka [101] over the content based features of the Web Spam Challenge 2006 Phases I and II data. The Web Spam Challenge 2006 evaluation target was the F-measure. Depending on the data set, the best forms of graph stacking improve the F-measure by 1-10%, as shown in Section 4.3.2.

Beside the spam data set we also test our graph labeling methods on a telephone call graph, a data type that appears less in the publications of the data mining community. Closely related to our work are the churn prediction results by machine learning methods on real data [100, 5, etc.]; these results however

do not exploit neighborhood information embedded in the call graph.

The telephone call graph is formed from the call detail record, a log of all calls within a time period including caller and callee id, duration, cost and time stamp. The vertex set consists of all nodes that appear at least once as caller or callee; over this set calls form directed edges from caller to callee.

Churn classification uses customer information (price package, time since in service etc.) as well as traffic aggregates in various call zones and directions. We use one year call detail record and all customer information up to a given time. The classification target consists of users who leave service in the fourth month "in future" (in a time period with no information available for the classifier). Due to the sparsity of positive instances (below 1% churn in a month) and a large amount of churn explained by external reasons, such as the customer moves, churn classification is a hard task. Baseline reaches $F = 0.08$ and this is improved to 0.1 by stacked graphical learning. In the industrial practice the goodness of the churn classification is measured by the recall of the top list of 10% of the customers, i.e. they are willing to involve a maximum of 10% of their customers in direct marketing campaigns and want to maximize the potential churn reached. In this sense our baseline classification has a recall of 40.8%, improved to 47% by stacked graphical learning.

In this chapter we concentrate on spreading trust (or no churn) and distrust (churn) information from known nodes with the help of hyperlink based similarity measures. Our main goal is to identify those graph based similarity features that can be used to classify unknown pages. We propose a set of spam and churn classification methods that combine graph based similarity to labeled nodes [9] with trust and distrust propagation methods, both backward and forward. For example given a link farm alliance [50] with one known target labeled as spam, similarity based features will automatically label other targets as spam as well.

Our stacked graphical learning algorithms generate features by averaging known and predicted labels for similar nodes of the graph by the measures in Section 4.2.1. We compare various similarity measures, including simple and multi-step neighborhood, co-citation, cosine and Jaccard similarity of the neighborhood as well as their multi-step variants [39] described in detail in Section 4.2. For the purposes of evaluation we consider these algorithms separately, by performing one classification experiment for each feature.

4.1 Related results

Several results has appeared that apply rank propagation to extend initial trust or distrust judgments over a small set of seed pages or sites to the entire web, such as trust [53, 103], distrust [81, 32] propagation in the neighborhood or

their combination [102] as well as graph based similarity measures [9]. These methods are either based on propagating trust forward or distrust backwards along the hyperlinks based on the idea that honest pages predominantly point to honest ones, or, stated the other way, spam pages are pointed by spam pages. Trust and distrust propagation originates from Guha et al. [48] for trust networks. Wu et al. [102] is the first to show its applicability for Web spam classification.

Trust and distrust propagation are in fact forms of semi-supervised learning surveyed by Zhu [106], a methodology to exploit unlabeled instances in supervised classification. Stacked graphical learning introduced by Kou and Cohen [61] is a simple implementation that outperforms the computationally expensive variants [61, 16].

Identifying spam pages is somewhat analogous to classifying web documents into multiple topics. Several results [83, and the references] demonstrate that classification accuracy can be significantly increased by taking into account the class labels assigned to neighboring nodes. In accordance with [9], Qi and Davison [83] found that most of the improvement comes from the neighborhood defined by co-citation.

Several link-based algorithms were designed to evaluate node-to-node similarities in networks that can be used to give alternate, similarity based weights to node pairs. We refer to [62] for an exhaustive list of the available methods ranging from co-citation to more complex measures such as max-flow/min-cut-based similarities of [65] in the vicinity graph of the query. Co-citation is in fact used in [48] as an elementary step of trust propagation. Another method [67] penalizes the biconnected component of a spam page in a subgraph obtained by backward distrust propagation.

Finally we mention the first example that gives anecdotal evidence for the usability of similarities in hyperlink structure to identify spam. Amitay et al. [2] extracted features based on the linkage patterns of web sites and trained a decision tree and a Bayesian classifier to classify each site to one of the 8 predefined functional categories. A cosine metric based clustering of the feature space produced a decent amount clusters whose members appeared to belong to the same spam ring. As it was not the original goal of their research, no results were published on classifying sites as spam or non-spam.

4.2 The stacked graphical learning framework

4.2.1 Feature generation

For a given unknown node u and edge weight function w (that may be in or out-degree, co-citation, PageRank etc.), our algorithm selects the k largest weight

neighbors of u to generate a new feature based on the known spam and honest hosts in this set. As in [9] we extract four different features from this set of size k or possibly less if u has less than k neighbors. Each neighbor v is either classified as spam with weight $p(v)$, or else labeled spam or non-spam; in these cases we let $p(v)$ be 0 and 1, respectively. Let s and h be the sum of $p(v)$ and $1 - p(v)$ in the set of neighbors; remember $s + h < k$ is possible. We define a weighted version s^* and h^* as the sum of $w(uv) \cdot p(v)$ and $w(uv) \cdot (1 - p(v))$.

In principle the value of k is a free parameter. In fact this value is fixed in most cases by the properties of the implementation. In our implementation the parameter k is not fixed beforehand, instead all nodes with non-zero similarity are taken into account.

We define our features as follows.

- Spam Ratio (SR): fraction of the number of spam within labeled spam and honest pages, $s/(s + h)$.
- Spam over non-spam (SON): number of spam divided by number of honest pages in the top list, s/h.
- Spam Value Ratio (SVR): sum of the similarity values of spam pages divided by the total similarity value of labeled spam and honest pages under the appropriate similarity function, $s^*/(s^* + h^*)$.
- Spam Value over non-spam Value (SVONV): similarity value sum for spam divided by same for honest, s^*/h^*.

In most of the experiments we use SVR that also performed best in [9]; a small comparison is made in Section 4.3.2.

We add the new feature defined by either of the above to the existing ones and repeat the classification process with the extended feature set. Since the features are unstable if $|N(u)|$, the number of nodes in the neighborhood is small, we also define versions SR', SON', SVR', SVONV' by regressing towards the undecided 1/2 or 1 value:

$$\text{SR}' = 1/2 + (\text{SR} - 1/2) \cdot (1 - 1/\sqrt{|N(u)|});$$

$$\text{SON}' = 1 + (\text{SON} - 1) \cdot (1 - 1/\sqrt{|N(u)|}).$$

4.2.2 Direction of propagation

We may use both the input directed graph, its transpose by changing the direction of each edge, or the undirected version arising as the union of the previous two graphs. We will refer to the three variants as *directed*, *reversed* and *undirected*

versions. For an edge weight function $d : V \times V \to \mathbf{R}$ we use $d^-(u, v) = d(v, u)$ for the reversed and $d^\pm = d + d^-$ for the undirected version. We extend this notion for an arbitrary similarity measure $sim(u, v)$ computed over edge weights d and compute $sim^-(u, v)$ over d^- and $sim^\pm(u, v)$ over $d^\pm$.

Performance of directed, reversed or undirected varies problem by problem: the templatic nature of a Web spam farm is best characterized by similarity of out-links (directed), honest pages have incoming links from honest ones (reversed) and finally similarity in a telephone call graph is best characterized by the undirected graph since communication is typically bidirectional regardless of the actual caller–callee direction.

4.2.3 Multi-step propagation

There are several variants of weighting neighbors at distance k. We may consider reachability and exact reachability as $d^k(u, v)_{\text{reach}} = 1$ if v is reachable from u by a walk over k edges, 0 otherwise, respectively $d^k_{\text{exact}}(u, v) = 1$ if v is reachable from u in exactly k steps and over no shorter paths, 0 otherwise. We may take the number and the weighted number of such walks: $d^k_{\text{num}}(u, v)$ is the number of walks over k edges that reach from u to v and $d^k_{\text{wnum}}(u, v)$ is the probability of reaching v when starting at u and at each step choosing a random neighbor with probability proportional to the outgoing edge weights. The main multi-step feature we use is PPR(u) (see Section 2.2.2), PageRank personalized to $p(v)$, the estimated spamicity of node v as in Section 4.2.1:

$$\text{PPR}(u) = \sum_k c(1-c)^k \sum_v p(v) \cdot d^k_{\text{wnum}}(u, v).$$

4.2.4 Co-citation, Jaccard and cosine

To compute the weight of edges in the Web graph we may use graph similarity measures such as Co-citation, Jaccard and cosine similarities (see Section 2.3.1). Co-citation turned out to be the most effective for Web spam classification in [9]. By the notation of Section 4.2.2, $coc^-(u, v)$ denotes bibliographic coupling (nodes pointed to by both u and v) and $coc^\pm(u, v)$ is the undirected co-citation. We may also use co-citation down-weighted by degree, $coc(u, v)/d(u) \cdot d(v)$.

Similarly to the above notations, for Jaccard and cosine similarity we use the notions $jac^-(u, v)$ and $cos^-(u, v)$ over the transposed graph and $jac^\pm(u, v)$ and $\cos^\pm(u, v)$ over the undirected graph.

Since filling a quadratic size matrix is infeasible, we calculate Jaccard and cosine only for existing edges. The resulting scheme down-weights unimportant edges but is unable to add "uncaught contacts" to the network. It is possible to

find all pairs with a weight above a given threshold by fingerprinting techniques, but in this thesis we omit the examination of this possibility.

4.3 Experiments

4.3.1 Data sets and methods

For Web spam classification we follow the same methodology as Castillo et al. [16]. We use the Web Spam Challenge Phase I data set WEBSPAM-UK2006 [18] that consists of 71% of the hosts classified as normal, 25% as spam and the remainder 4% as undecided as well as the Phase II data set WEBSPAM-LIP6-2006. The feature set we are using for classification is the "public feature set" (see Section 3.4.1) which is precalculated and included in the above data sets. In this preliminary experiment we consider three tasks. First we use Phase I data (the *Domain Or Two Humans* classification that introduces additional non-spam domains and gives 10% spam among the $5{,}622$ labeled sites) with the publicly available features of [16] and then classify by the cost sensitive C4.5 implementation of the machine learning toolkit Weka [101] with bagging. Then we use the Phase II data set features and use the Naive Bayes classifier of Weka. Finally we compute all graph based features of [16] for the Phase II data graph and classify by C4.5 again. We combined the text and graph classifiers by SVM.

For churn classification we use data from a small Hungarian landline telephone service provider. We form features based on aggregated call cost duration in different cost segments, including daytime and off-peak, weekday and weekend as well as local and different long-distance call volumes. Part of the users perform calls via an alternate provider by dialing a prefix; these calls are aggregated similarly for each user. We also use the pricing package information that also includes a distinction of company and family lines as well as the start date of the service usage. For a time range of 12 months, after aggregating calls between the same pairs of callers we obtained a graph with $n = 66{,}000$ nodes and $m = 1{,}360{,}000$ directed edges.

We use the cost sensitive C4.5 implementation of the machine learning toolkit Weka [101] with bagging. Since the running times on the full data set were over 10 hours we also compiled a smaller data set where a random sample of non-churned users were dropped, resulting in $7{,}151$ users but we kept the entire graph.

4.3.2 Classification results

In table 4.1 we can compare the performance of the different graph similarity measures for 1 and 2 iterations in F-measure (see Section 2.5.1) for the best selected settings, with the best results in bold. The column *none* shows the baseline results when we are not using graph stacking. Column *d* shows the results of graph stacking where edge weight is defined by the number of links between two sites. For the Web spam data we measure over the testing labels while for churn we use 10-fold crossvalidation. Since the text and link SVM-combined Web Spam II experiment is computationally very expensive, we only computed the base and the simple neighbor methods that give 0.738 and improve to 0.748 for the small and 0.338 vs. 0.449 for the large graph.

In Table 4.2 we can see that the difference in F-measure between the feature generation methods of Section 4.2.1 are minor and the length of the top list has little effect in the range of k between 100 and 1000, although for cocitation the very long and for others the very short lists deteriorate the performance.

4.4 Conclusions

We presented Web spam and landline telephone churn classification measurements over the Web Spam Challenge Phase II and a small Hungarian landline telephone provider year 2005 data set. Our experiments demonstrated that stacked graphical learning in combination with graph node similarity methods improve classification accuracy in both cases. Due to the large number of possible feature generation methods the results are by no means complete but show a very good performance of co-citation and little actual use of the neighborhood beyond two steps in the graph.

4.5 My contribution

The results demonstrated in this chapter are joint work with András Benczúr, Károly Csalogány and László Lukács and were published in [2]. My contribution is the implementation of the link feature generation methods, the definition of graph similarity measures, and the identification of the best machine learning methods available as part of the Weka machine learning toolkit. László Lukács defined the aggregation methods for graph stacking and pre-processed the churn dataset. Károly Csalogány calculated the TFIDF feature set and evaluated the results.

F-measure ×1000	none	graph stacking																
		d	coc		coc^-		$coc^\pm$		Jac		Jac^-		$Jac^\pm$		cosine		PPR	
iterations		1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Web Spam I	689	695	707	709	669	677	722	**724**	715	703	689	690	679	680	698	699	715	719
Web Spam II small, text	592	589	601	**605**	598	599	599	601	590	590	592	594	593	595	600	601	599	600
Web Spam II small, link	762	752	788	**793**	774	765	748	738	756	762	782	777	766	756	760	760	731	737
Web Spam II large, link	939	962	983	984	987	**988**	983	984	984	985	975	976	961	953	982	982	958	960
Churn	086	**102**	063	052	079	088	**102**	083	067	065	059	066	097	089	084	065	092	087
Churn, nonchurn sampled	161	155	141	142	197	200	114	121	254	265	153	147	175	158	267	**280**	277	257

Table 4.1: 1000 times the F-measure shown for different data sets and edge weights.

	d	coc	coc$^{\pm}$	Jac	Jac$^{\pm}$	cosine	PPR
SON'	602	615	602	600	599	599	599
SON	600	614	602	599	599	599	598
SR'	**603**	618	609	**611**	**606**	**610**	**608**
SR	601	**619**	**611**	610	605	**610**	603
SVONV'	602	607	602	598	596	597	599
SVONV	600	606	602	598	596	598	599
SVR'	**603**	618	609	603	**606**	604	600
SVR	601	**619**	**611**	600	604	601	600

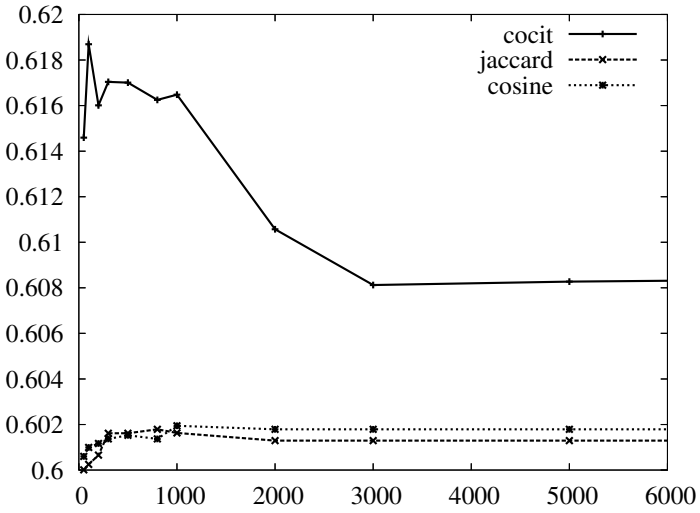


Table 4.2: **Top:** 1000 times the F-measure shown for different data weights and feature generation methods. **Bottom:** the effect of the top list size for SVR. Results are shown for the text features of the small Phase II graph and single-iteration stacked graphical classification.

Chapter 5

Sonar stacking

In this chapter we extend the stacked graphical learning results of the previous chapter in a new combination with the so-called connectivity sonar features. The connectivity sonar introduced by Amitay et al. in [2] describes the internal structure of a Web site for use in web classification.

A key step of stacked graphical learning is turning the predicted labels of the neighbors into new features of a node x. Traditionally the new features have been simple aggregates, such as the majority or average [16] of the neighbor labels. When it comes to host-level web classification, the use of such simple aggregates means that the information about the internal link structure of host x, the exact position of page-level in-links and out-links is lost.

In this chapter we introduce a refined set of features for x that combine the predictions for the neighbors of x with information about the internal structure of x, and the location of links to and from its labeled neighbors. Accordingly, for a host x and some class label c, we extend the "connectivity sonar" features proposed by Amitay et al. [2] by measuring

- the distribution of links to and from neighbor hosts with predicted label c;
- the average level (within the host internal link graph) of inlinks and out-links labeled c; and
- the fraction of links labeled c at the top and leaf levels.

To assess the prediction power of the proposed features, we run experiments on the WEBSPAM-UK2007 data set extended with topical category labels from the DMOZ directory. Our techniques are evaluated along several alternatives and yield a considerable improvement in terms of area-under-the-curve (AUC) over earlier approaches.

5.1 Related results

Our new results are formed as a combination of graph stacking and Web site structural characterization. For related results on stacked graphical learning, we refer the reader to Chapter 4.

As the starting point of our investigation into Web site structure, Amitay et al. [2] classify web hosts based on information about their internal linkage. They define the depth of a page within a host as the number of slashes in the corresponding URL and extract features such as

- the distribution of pages at different levels;
- the distribution of inlinks and outlinks at different levels;
- the fraction of links at the top and leaf levels; and
- comparisons of the distribution of external and internal up, down, and cross links.

Gyöngyi et al. [49] compute personalized PageRank vectors with the personalization vector being one of the 14 Open Directory top-level topic indicator vectors for each. This way they compute 14 features for each host and use simple majority vote to derive the topic of initially unlabeled hosts.

5.2 Machine learning by sonar stacking

Our key idea for sonar stacking classification consists of the combination of the internal and external link structure of pages on a host u with the label of other hosts that are pointing to or pointed by u's pages. The following two subsections describe the pair of core preexisting ingredients: the "sonar" features of [2], characterizing the host's page-level link structure; and label propagation [49], edge similarity measures [9], and graph stacking [16] as used in web host classification. Personalized PageRank, a special case of graph stacking, is also presented separately in Section 2.2.2, since it is used in the baseline method of [49]. Then in Section 5.2.2 we introduce our new feature set.

5.2.1 The "Connectivity Sonar" Features

The idea of the "Connectivity Sonar" of Amitay et al. [2] is that four main host types (and several more derived types) are characterized by the structure of page

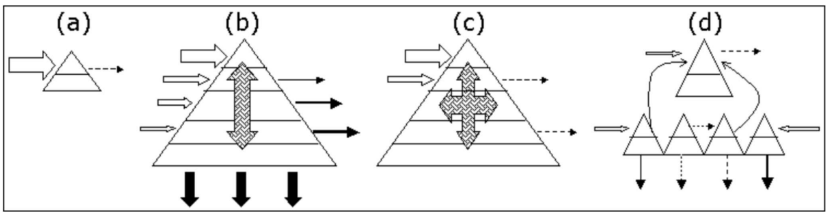


Figure 5.1: Schematic drawings of (a) search engines, (b) directories, (c) corporate sites and (d) virtual hosting services

levels within the host and their internal and external linkage. The main host types shown in Figure 5.1[1] are as follows:

1. *Search engines* have a small number of pages and few outgoing links, since most of their contents and links are generated dynamically through the search interface. In contrast, they receive huge amounts of inlinks.

2. *Directories* form deep sites with navigation mainly in the up-down direction. The number of inlinks decreases as we move downwards while the situation is quite the opposite for outlinks.

3. *Corporate sites* may be deep, but receive most of their inlinks at the top level and feature only a few outgoing links. At the same time, such sites have rich and meaningful intra-navigation structure, not just up and down but also sidewise.

4. *Virtual hosting services* consists of small virtual sites that all link to the main hosting site but otherwise act as independent sub-sites. External linkage to the sub-sites may be more prevalent than to the main site.

The types of hosts, in the above sense, are strongly related to certain types of content. For example, the Open Directory's Business category consists mainly of corporate sites, while the Science category contains university sites, which are combinations of corporate and virtual hosting sites, as noted in [2]. Finally, the Reference category contains several directories. While we will never encounter an explicit labeling of hosts by type, type information is used implicitly when host link features are incorporated in the topical classifier.

Sixteen link-based "Sonar" features are proposed [2]. In order to compute the features of a host u, we consider the graph with the vertex set formed by all pages within the host, plus a single node for each neighbor host. Within the

[1]The figure is copied from the original "Connectivity Sonar" paper [2]

set of pages of host u, the level of a page is defined as the number of slashes in the corresponding relative URL. In brief, the features are as follows (for more details, see [2]):

F1 Average level of the page;

F2 Percentage of pages in the most populated level;

F3 Top level page **expansion ratio**: the number of pages in the second level divided by the number of pages in the top-level;

F4 Inlinks per page;

F5 Outlinks per page;

F6 Outlinks per inlink;

F7 Top-level inlink portion;

F8 Outlinks per leaf page;

F9 Average level of inlinks;

F10 Average level of outlinks;

F11 The difference between F10 and F9;

F12 Percentage of inlinks to most popular level;

F13 Percentage of outlinks from most emitting level;

F14 Crosslinks per page;

F15 The ratio between internal up, down, and side-links vs. cross-links on levels 2–4;

F16 Top level internal inlinks per page.

We will use these features in part directly in our baseline classifiers and in part as building blocks in our new feature set that also takes the labels of the neighbor hosts into account.

5.2.2 Sonar Stacking: The New Features

By means of graph stacking, we may incorporate the predicted class of the neighbors into the features that characterize the link structure at different levels of the host in question.

To understand the idea, let us consider a host that is popular in general and linked from other hosts belonging to various Open Directory categories. Such a host may be a university site, an Arts page or an online store. The possible overall popularity of these hosts may deteriorate the performance of link-based classifiers such as PPR or graph stacking. However if we deeper explore the internal page link structure, we may observe that hosts on the same topic "collaborate" or "collude" by pointing deeper into one another, and implicitly use this observation to achieve better classification. As another example, Business hosts may be competitive and even if they link to each other, the links may simply point to the top-level page, not providing refined pointers inside the competitor's host. Our new features are designed so that they can distinguish linkage types and adjust the label propagation in the graph stacking classification accordingly.

Our "Sonar Stacking" features are defined in the graph stacking framework by first classifying all hosts based on features from the previous phase, and then using the prediction $p(u)$ for host u as a weight in the previously introduced "Sonar" features. We define the Sonar Stacking features as a combination of the following complementary pairs:

S Sum of $p(u)$;

A Average of $p(u)$;

I Inlinks;

O Outlinks;

T Top level;

L Leaf level.

We may choose and combine one option from each of the pairs SA, IO and TL; for example, the choice SIT stands for the sum of weights $p(u)$ for hosts u pointing to the top level page. In addition to the eight possible combined features, denoted from now on as S1–8, we use two additional ones,

S9 Average level of inlinks (F9) weighted by prediction; and

S10 Average level of outlinks (F10) weighted by prediction.

5.2.3 Graph stacking, Edge Weights, and Aggregation

In the previous chapter, we compared how graph stacking performs with several different aggregation methods and graph similarity measures. In this chapter, we use all the similarity measures and the following aggregation methods:

- The average of the weights $p(u)$, as in [16];
- The sum of the weighted positive prediction $\text{sim}(u, v) \cdot p(v)$, which replaces simple degree counts in sonar features F4–13.
- The Value Ratio (SVR, which performed best in [2]): the sum of the weighted positive prediction $\text{sim}(u, v) \cdot p(v)$, divided by the sum of all similarities $\text{sim}(u, v)$. In order to ignore the effect of low similarity on some instance u, we trim $\text{sim}(u, v)$ to zero for all except the top 100 nodes v most similar to u.

5.3 Experiments

5.3.1 Data set

We evaluate the performance of the proposed classification approach on the WEBSPAM-UK2007 data set, but instead of classifying Spam, we labeled 38,415 hosts by top-level Open Directory (ODP) categories (using links extracted from topic subtrees in the directory), out of which 32,960 received a unique label. The task is to classify Web pages based on which ODP category are they belonging to. We note that the main BBC site belongs to all 14 categories and over 20 sites belong to at least 10 categories each. We discarded the labels of all hosts with multiple labels. Also, since one of our baseline classifiers, SVM on TF-IDF required extensive CPU time, we took a random sample of 5,000 labeled hosts for the ODP classification task. The resulting distribution of labels is shown in Table 5.1. The table also contains a summary of our results, in the form of overall improvement in classification performance reached by the sonar stacking; the results are described in detail in Section 7.4.4.

Our classification procedure works as follows. We split features into related sets and for each ODP category and for each set of features we use the best fitting binary classifier, which we describe in the following subsection. These classifiers are then combined by Logit-Boost, a method that, in our cross-validation experiment, outperformed logistic regression suggested in [24]. All throughout our experiments, we used the classifier implementations of the Weka [101] machine learning toolkit.

category	total count	sample count	AUC incr.	best method
AR (arts)	5076	537	.012	CStacking 1
BU (business)	7514	880	.008	CStacking 2
CO (computer)	3115	323	.022	PStacking 1
GA (games)	817	73	.009	PStacking 1
HE (health)	1353	132	.010	CStacking 1
HO (home)	499	43		Text+PPR+Sonar
KI (kids)	422	23		Text+PPR+Sonar
NE (news)	119	8		Sonar
RC (recreation)	3599	386	.019	CStacking 2
RE (reference)	1785	125		HStacking 1
SC (science)	2422	179	.024	CStacking 2
SH (shopping)	3311	392	.038	CStacking 2
SO (society)	4507	438	.002	CStacking 1
SP (sports)	3876	449	.011	CStacking 2

Table 5.1: The distribution of hosts over the top-level ODP categories with total and evaluation sample counts, allowing for multiple labels per host. The AUC column contains the absolute increase obtained by using page-level stacking with binary classifiers for each ODP topic and the last column shows the corresponding best performing method.

5.3.2 Baseline Features and Classifiers

The most natural way to classify the topic of web sites is to use their text content. Our first and often most powerful classifier is SVM over TF-IDF, averaged over all pages of the host. We use the most frequent 20,000 terms. SVM is generally believed to be one of the best performing text classifiers; for example for larger web collections SVM performed better in the related experiment of [3].

Gyöngyi et al. [49] describe a method that relies solely on the fact that hosts within the same category tend to be connected stronger than across different categories. We implement their method that is based on personalized PageRank as described in Section 2.2.2. For the 14 ODP categories we obtain 14 PPR values as features for each host. Unlike in [49], where majority vote is used, we train a C4.5 classifier over these features.

The original 16 "Sonar" features F1–16 from [2] are formed by defining the level of a page within a host as the number of slashes in its URL. We duplicate all of 16 features by also using the alternate definition of the level as the shortest path from the root page of the host (breadth-first level). In addition, for each of features F4–13, which rely on in- or outdegree counts, we form two separate features for internal and external links, and a third aggregate feature. Altogether this results in 68 features that we feed into a C4.5 classifier.

For the combination of different classifiers, we have tested several methods, including C4.5, logistic regression (suggested for Web spam filtering by the best performing Web Spam Challenge 2007 method [24]) and random forest; the Logit-Boost classifier of the Weka machine learning toolkit performed best and is hence used in the rest of our experiments.

Graph stacking, a method first used for web classification in the spam filtering work of Castillo et al. [16], is also one of our baseline classifiers. We use an extended version of it, described in [2], which compares various uses of edge weights, as presented in Section 5.2.3. While the measured effects of various weights lead to the conclusion that cocitation is superior, we use a many-feature stacking method that in each stacking phase defines 10 new features, using all of the following similarity measures: $\mathrm{PPR}_u(v)$; plus undirected, directed, and reversed versions of degree count, cocitation, and Jaccard. As a reminder, we note that in all these methods we use Value Ratio (SVR in [2]) as defined in Section 4.2.1, and that all features are classified by Logit-Boost.

5.3.3 Results

We show our results in terms of the AUC measure for the WEBSPAM-UK2007 data set in Table 5.2. In addition, Figure 5.3 presents the distribution of some selected features that performed well in classification.

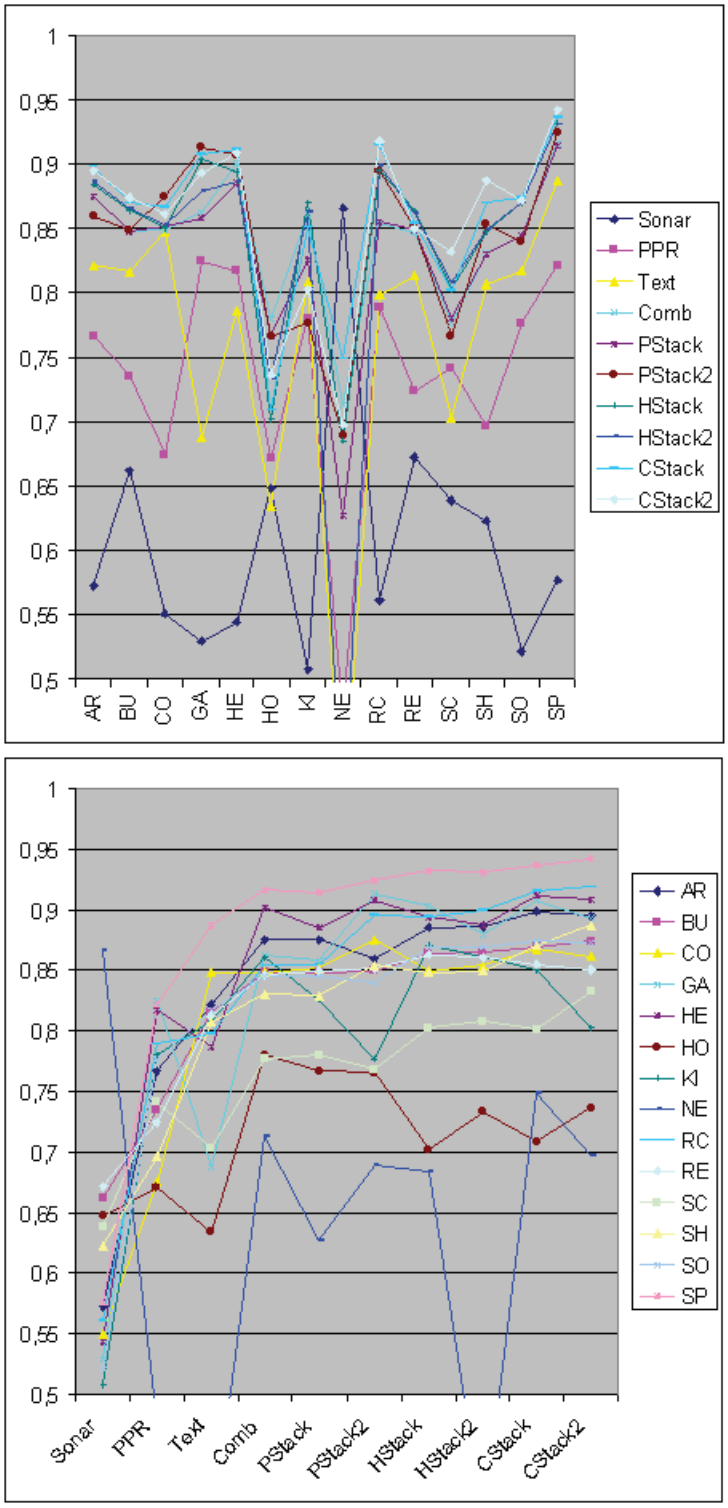
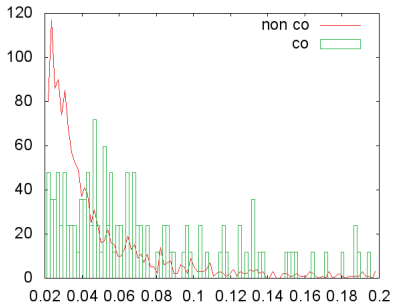
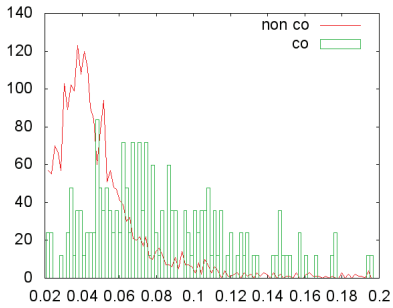


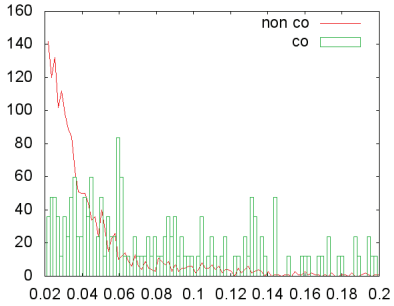
Figure 5.2: AUC measures for the WEBSPAM-UK2007 data set with different sets of features used along the baseline classifiers. Here Combination denotes all non-stacked features (text, PPR and Sonar); HStack is the host-level (non-Sonar) stacking; PStack is the page-level (Sonar) stacking over text, PPR, and Sonar; while CStack is the combination of all.



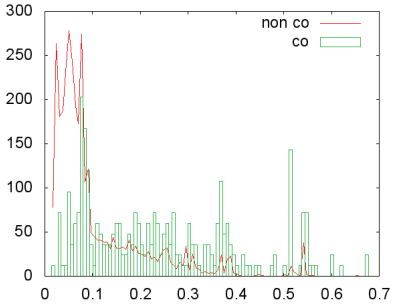
(a) Host-level stacked average prediction over links for CO.



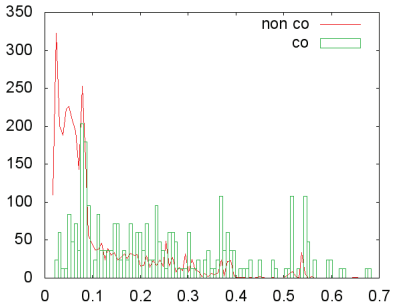
(b) Host-level stacked average prediction over Jaccard similar hosts for CO.



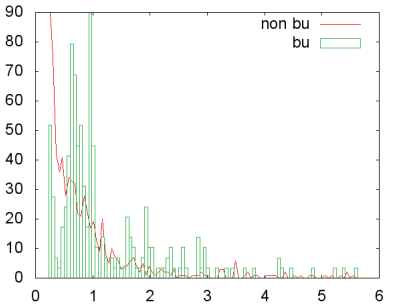
(c) Same as (b) with two iterations.



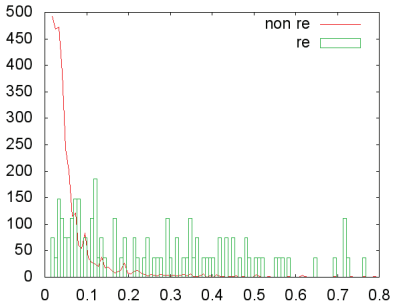
(d) Root average outlink prediction for CO.



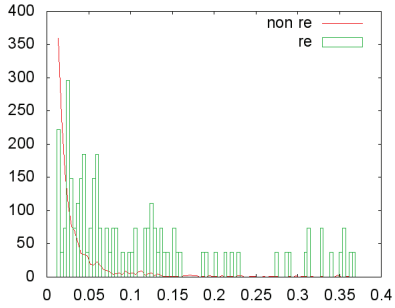
(e) Leaf average outlink prediction for CO.



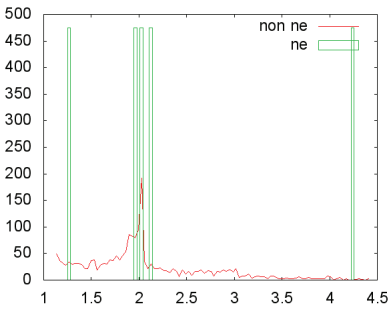
(f) Average category of leaf inlinks, BU.



(g) Leaf average outlink prediction for RE.



(h) Host-level stacked average prediction over personalized PageRank weighted links for RE.



(i) Average level of inlinks for NE (no stacking).

Figure 5.3: The distribution of stacked feature values for representative topical categories of the UK2007-WEBSPAM data set.

For most ODP categories, page-level Sonar Stacking works similar to host-level graph stacking: both top- and bottom-level, in- and out-averages yield distinctive features. The first interesting observation that we illustrate in Figure 5.3 is that links from and to same-category pages go more deeper within the hosts than between different topics.

Next we investigate the stacked average prediction of root- and leaf-level pages. We describe Computers (CO) in detail; for other hosts we only show differences. For the host-level stacking of CO there is a peak (Figures 5.3a,b) in the medium average prediction by all host similarity measures (linkage, cocitation and Jaccard). While this peak is reduced in a second iteration of stacking, we still observe it, for example, for the Jaccard measure (Figure 5.3c). In contrast, page-level linkage has better predictability, in particular for leaf- and root-level outgoing linkage (Figures 5.3d,e).

When we analyze the linkage from the same vs. different types of hosts in more detail, we observe differences between the 10 categories where page-level stacking yields improvement. While in general deeper linkage indicates the same category, Business is an exception where the lack of leaf-level linkage characterizes the hosts (Figure 5.3f).

A particularly interesting category is Reference (RE), where host-level stacking outperforms our new method. Here we can still observe characteristic page-level stacking features (Figure 5.3g), though the category is apparently so strongly interlinked that host-level stacking (Figure 5.3h) cannot be further refined.

The categories HO, KI, and in particular NE show peculiar behavior. News hosts are best (and, as seen in Figure 5.2, only) characterized by their internal structure and thus are indistinguishable from non-News by the stacked features.

These hosts can be spotted by deep pages as well as out-links (Figure 5.3i). As a word of caution, we note that the results concerning the inefficiency of graph stacking may be inconclusive due to the small number of positive instances in the sample.

Next we consider the average level of the in- and outlinks of different types of hosts in Figure 5.4. We observe three types of behavior. Two large categories HE (a) and HO, as well as the smaller KI and NE show no characteristic behavior. In this sense, categories AR (b, c), BU, CO, RC, SH, SO, and SP are typical in that there is no distinction between the level of linkage from same vs. different categories. At the same time, there are peaks at integer values for inlinks meaning that typically a given level of the host is in-linked. Finally for GA, RE, and SC both the in- and outlink depths are distinctive, while the categories also feature the peaks for integer values of inlinks (d, e, f).

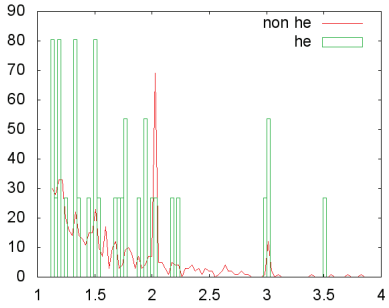
In summary we observe the following characteristic behavior of hosts:

Class 1: Strong collaboration. These hosts are linked in general deeper from the same category; different categories also tend to link to these hosts, but point mostly to the root page. There is strong collaboration within this category, while external links simply indicate the existence of particular hosts, without detailed inside pointers. We obtain the highest improvement by page-level "Sonar" stacking here. This class contains Arts, Computer, Games, Science and Shopping (AR, CO, GA, SC, and SH).

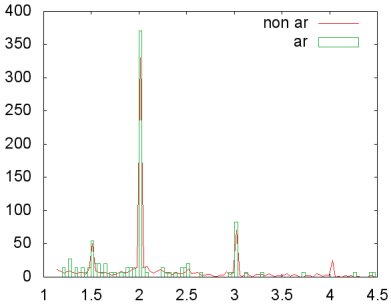
Class 2: Content defined. For these hosts, text content is so characteristic that it can only slightly be improved upon by other features; the choice of features varies within the class. Representatives are Computer and Kids (CO, KI), and to a lesser extent Business, Reference, Shopping and Sports (BU, RE, SH, SP) also belong here. While internal linkage within these categories is not as strong as for Class 1, "Sonar" stacking often improves for reasons either similar to Class 1, or, in contrast, as in the case of BU (Figure 5.3f), due to the lack of deep links.

Class 3: Closed community. These hosts are in general less linked from different categories. PPR and the stacked features are very strong in this class. Representatives are Games, Health, Kids, Recreation and Sports (GA, HE, KI, RC, SP). There is partial overlap with Class 2 and some similarity with Class 1 in that for some categories page-level "Sonar" stacking improves over host-level stacking.

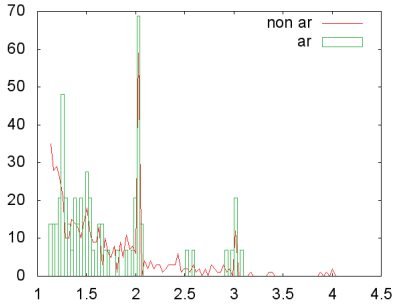
The remaining categories are considered exceptional. Home (HO) fails all attempts of stacking and its classification performance is the worst of all the categories. News (NE) performs better than HO when classified by the "Sonar"



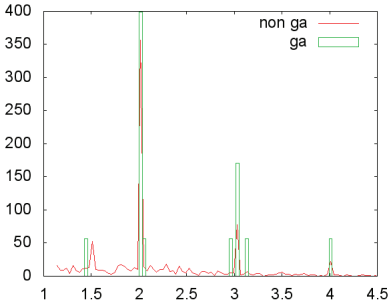
(a) Average level of outlinks, HE



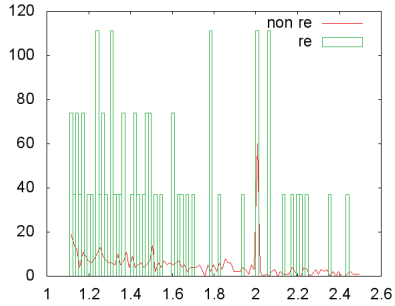
(b) Average level of inlinks, AR



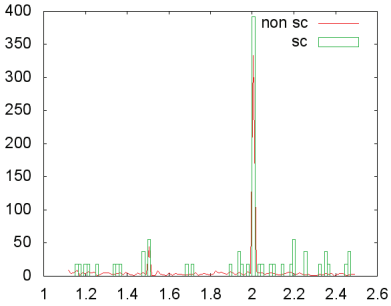
(c) Average level of outlinks, AR



(d) Average level of inlinks, GA



(e) Average level of outlinks, RE



(f) Average level of inlinks, SC

Figure 5.4: Sample distribution in- and outlink level from same and different categories.

features, but all other features perform and combine poorly. Note, however, the small positive sample size of NE, which makes the results inconclusive. In addition to HO and NE, Science (SC) is the third outlier category, with a poor performance of the TF-IDF classifier. These three categories are diverse in text content and thus the use of non-text classification features is an absolute must.

5.4 Conclusions

In this Chapter we have demonstrated the applicability of page-level linkage analysis for Web host classification in a graph stacking framework. Our features are stacked over a base classifier outcome and characterize the linkage predicted from the target category, distinguished by the level within the given host. The features work particularly well for hosts of general popularity, which receive a large number of off-topic links, but where deeper analysis reveals topically similar hosts that tend to refer to more details and link deeper within each other. Page-level stacking improves even over a very strong baseline combination of text, personalized PageRank, and stacked graphical classification for most top-level Open Directory categories. Our experiments were conducted over more than 30,000 hosts of the `.uk` domain which are listed in the Open Directory, and the absolute improvement in AUC was between 1-4% for most categories.

5.5 My contribution

The results demonstrated in this chapter form an extended and greatly improved version of the graph stacking results of Chapter 4 in joint work with András Benczúr, Zoltán Gyöngyi and Miklós Kurucz. My main contribution was creating the new sonar stacking features, running all the experiments, comparing different methods and combining the results of separate machine learning algorithms. Zoltán Gyöngyi labeled the WEBSPAM-UK dataset with the DMOZ categories. Miklós Kurucz computed the original sonar features for the baseline methods.

Chapter 6

Cross-lingual text classification

It has already been known from the early results on text classification that "obtaining classification labels is expensive" [70]. This is especially true in multilingual collections where either separate training labels have to be produced for each language in question, or techniques of cross-lingual information retrieval [31] or machine translation [73] has to be used.

While several results focus on cross-lingual classification of general text corpora [7, 88, 98, and many more], we concentrate on the special and characteristically different problem of Web classification. Our goals range from serving focused crawling of topically selected Web collections [19], prioritizing the crawl for overall quality to filtering spam, porn, or illicit content [55].

In our methods scalability considerations play central role. We envision users of limited resources or crawl-time filtering needs, e.g. prioritization or stopping spam *before* they reach the archive. Hence, we prefer local methods and features computable from a sample set of pages of a host instead of global ones, such as PageRank that is expensive to update whenever a new host appears. For the same reason we cannot rely on heavy Natural Language Processing (NLP) methods that will likely not scale to the size of the Web.

Our results combine methods from two areas, cross-lingual information retrieval and Web classification. Traditional methods in cross-lingual information retrieval use dictionaries, machine translation methods, and more recently multilingual Wikipedia editions. Web classification on the other hand relies on features of content and linkage [16], some of which are language independent (see Section 3.4.1). However, even most of the language independent features depend on the domain: PageRank and its variants may have different distribution for differing interconnectivity and the ratio of the "boundary": the pages not included but pointed to by another page in the domain, crawl, or language. TrustRank (see Section 3.4.1) and query popularity based features depend on

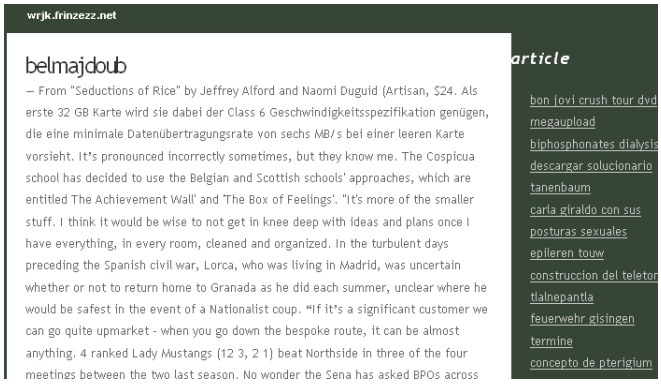


Figure 6.1: Portion of a mixed language machine generated spam page.

the availability of a trusted seed set, typically hosts listed in the Open Directory Project (see 3.2), and the coverage of search queries. Finally, the typical word length and text entropy may also vary language by language.

In this chapter we experiment with a new combination of learning methods and cross-lingual features for web classification. Our task is different from standard methods of cross-lingual text classification (see [98] and references therein) in the following aspects:

- We classify hosts not individual pages. While arguably page-level classification would exploit the available information to a similar or better degree, it is computationally much less feasible, often beyond practical use. In addition, labeling a page or an entire host is almost the same effort for a human and hence the ratio of training vs. testing size is favorable for host level classification.

- Even if we consider a national domain, the actual language used in a host can be mixed, especially for spam pages automatically generated from chunks (see Fig. 6.1 as an example).

- We may exploit multilingualism by classifying a host based on its part written in English.

In this chapter we investigate how much various classes of Web content and linkage features, some requiring very high computational effort, add to the classification accuracy. As the bag of words representation turned out to describe Web hosts best for most classification tasks of the Discovery Challenge [34], we realized that new text classification methods are needed for our tasks.

Based on recent results in Web spam filtering, we also collect and handle a large number of features and test a variety of machine learning techniques in-

cluding SVM, ensemble selection, LogitBoost and Random Forest (see Section 2.4). Our key findings are summarized next.

- Hosts that contain a mix of English and national language content, likely translations, yield a very powerful resource for cross-lingual classification. Some of our methods may even work without using dictionaries, not to mention more complex tools of natural language processing.

- Similar to our previous English-only results, the bag-of-words representation together with appropriate machine learning techniques is the strongest method for Web classification.

- The "public" spam features of Castillo et al. [16], especially the content-based ones, depend heavily on the data collection and have little generalizational power. For spam classification they require cross-corpus normalization while for topical classification, the content based features do not seem to be applicable.

To assess the prediction power of the proposed features, we run experiments over the `.pt` domain [45, 46]. We perform topical classification into one of selected 7 top-level categories of the Open Directory (`http://dmoz.org`). Our techniques are evaluated along several alternatives and yield a considerable improvement in terms of area-under-the-curve (AUC).

6.1 Related results

In this section, we review results related to cross-lingual classification. For general results in the area of Web classification, see Section 3.4.

In general, cross-lingual classification either works by translating documents [88, 63, 98], or terms only [7], or using an intermediate language-independent representation of concepts [99]. For general results on cross-lingual text classification we refer to [7] which proposes linguistic resources such as dictionaries similar to the ones used in cross-lingual information retrieval. As a broad overview, we refer to the CLEF Ad Hoc tasks overview papers, e.g. [31] in the latter area. We also note that several results exploit Wikipedia linkage and local editions [94, 69, 44].

Several cross-language classification results work over "pseudo-English" documents, similar to ours, by translating key terms into English using dictionaries [7] or using latent semantic analysis [33, 82]. The cross-lingual classification results reported are however, unlike ours, much worse than the monolingual baselines.

Semi-supervised learning finds applications in cross-lingual classification where, similar to our methods, the unlabeled part of the data is also used for building the model. Expectation maximization is used in [88, 93] for cleansing the classifier model from translation errors; others [86] exploit document similarities over the unlabeled corpus. In [98] co-training of machine translated Chinese and English test is used for sentiment analysis.

Closest to our goals is the method of [63] for classifying Chinese Web pages using English training data, however, either because of the cultural differences between Chinese and English content or the fact that they classified on the page and not host level, they achieve accuracy metrics much weaker than for the monolingual counterpart. We also note that they are aware of the existence of multilingual content but they apparently do not exploit the full power of multilingual hosts. Finally, a recent Web page classification method described in [99] uses matrix tri-factorization for learning an auxiliary language, an approach that we find computationally infeasible for classification in the scale of a top level domain.

6.2 Method

Our Web host classification method applies a classifier ensemble consisting of features based on content and linkage as well as various English, translated, and semi-supervised Portuguese bag of words models. The following subsections describe the core ingredients. We have already described the basic features in sections 3.4.1, 3.4.1 and 3.4.1. Some of these features are language independent but some of theme are not. In Sections 6.2.1 and 6.2.2 we describe how we can use these features for cross-lingual classification.

6.2.1 Features: Content

1. *Number of words in the page, in the title.*
2. *Average length of words.*
3. *Fraction of anchor text.*
4. *Fraction of visible text.*
5. *Compression rate.*
6. *Corpus precision, recall.*
7. *Query precision, recall.*

8. *Entropy.*

9. *Independent n-gram likelihoods.*

Here feature classes 1–5 and 8–9 can be normalized by using the average and standard deviation values over the two collections, although classs 3–4 are likely domain- and language-independent.

Corpus precision and recall are defined over the k most frequent words in the data set, excluding stopwords. Corpus precision is the fraction of words in a page that appear in the set of popular terms while corpus recall is the fraction of popular terms that appear in the page. This class of features is language independent but rely on different lists of most frequent terms for the two data sets.

Query precision and recall is based on frequencies from query logs that have to be either compiled separately for each language or domain (questions from Portugal likely have different distribution than from Brazil) or the English query list has to be translated. Since we had no access to a query log in Portuguese, we selected the second approach.

6.2.2 Features: Linkage

One of the strongest features among link features is TrustRank [54]. A PageRank personalized on known honest (non-Spam) hosts. TrustRank however needs a trusted seed set. Typically hosts that appear in the Open Directory Project (ODP) are used as seed. Unfortunately ODP acts as our negative sample set as well, hence in this chapter we have to omit TrustRank.

6.2.3 Features: Bag-of-Words

Bag of words features are the strongest features for Web classification [34]. The bag of words representation of a Web host consists of the top 10,000 most frequent terms after stop word removal.

In order to classify hosts in Portuguese, we translate the Portuguese terms to construct an English bag of words representation of the host. The procedure is described in Algorithm 1 with the following considerations:

- Short terms are not translated as they typically cause noise and often coincide between the languages.

- Multiple translation alternatives exist. We consider all translations, however split the term frequency value between them in order not to overweight terms with many translations. A smarter but more complex weighting method is described in [93].

- Multi-word translation such as *Monday* through *Friday* translated into *Segunda* through *Sexta feira* could only be handled by counting frequencies of expressions that we skip to simplify computations.

Algorithm 1 Algorithm for translating Portuguese term counts for evaluation by an English model

```
for all English terms en do
  count[en] = count of term en in host h
for all Portuguese terms pt of at least four letters do
  count_pt[pt] = count of term pt in host h
  variants = number of single-term English translations of pt
  for all single-term Portuguese translations en pf pt do
    count[en]+ = count_pt[pt]/variants
Classify h using term counts count[en]
```

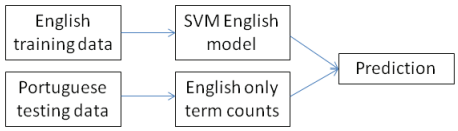
Finally we note that we use the Okapi BM25 term weighting scheme (see Section 2.4) that turned out to perform best in an earlier experiment [34]. As optimal parameters, an exceptionally low value $k = 1$ and a large $b = 0.5$ turned out to perform best in preliminary experiments. Low k means very quick saturation of the term frequency function while large b down-weights content from very large Web hosts. We do not show extensive experiments on these parameters.

6.2.4 Semi-supervised cross-lingual learning based on multilingual Web sites

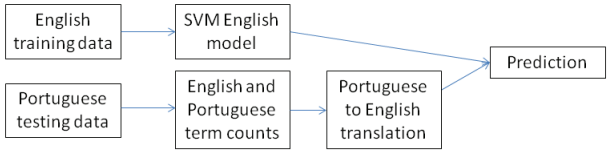
A large portion of national language Web content appears on the same host in English version as seen in Fig. 6.3. This figure shows the proportion of the total frequency of the 10,000 most frequent Portuguese terms within the sum of the Portuguese and English top 10,000 frequencies. This fact gives us rise of several options to use English, Portuguese and mixed language text classification. As summarized in Fig. 6.2, the simplest solution is to ignore non-English content and simply use term frequencies of the most frequent English terms as measured over the English part of ClueWeb09. Another solution, as described in Section 6.2.3, is to translate all content term by term into English and use the model trained over ClueWeb09 again.

We may however leverage on mixed language hosts to classify without using a dictionary in a semi-supervised procedure using these unlabeled hosts. In Algorithm 2 we give a two-step stacked classification procedure summarized in Fig. 6.2c. First we select hosts that contain an appropriate mix of English and

(a) Prediction by using the English terms only.



(b) Terms in the Portuguese testing data translated into English.



(c) After applying the method of Fig. 6.2a, strongest positive and negative predictions are used for training a model in the target language.

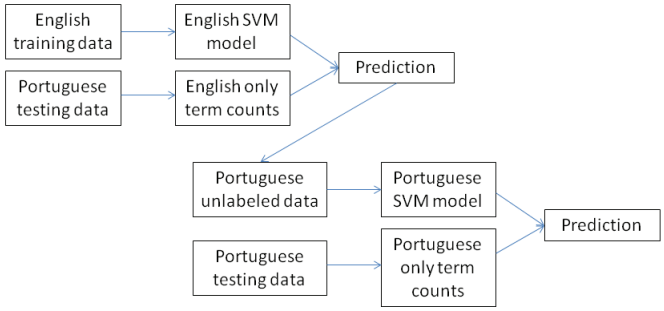


Figure 6.2: Three methods for classifying mixed language content based on a monolingual training set.

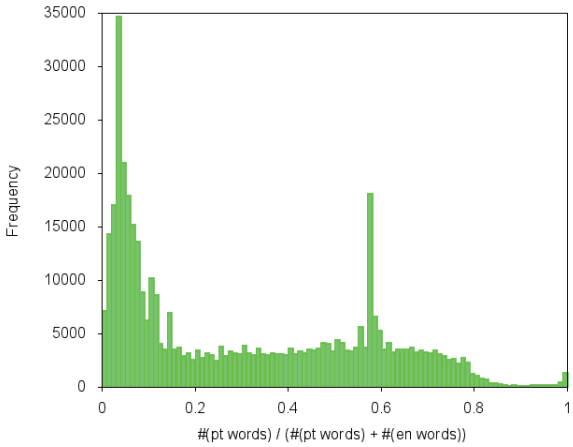


Figure 6.3: Statistics for the language distribution of most frequent terms in Web hosts over the `.pt` domain, with the 257,000 English-only hosts removed.

Portuguese terms, the middle range in Fig. 6.3 between threshold_low $= 0.4$ and threshold_high $= 0.6$. Based on the English term frequencies of these hosts, we give prediction using a model trained over the English part of ClueWeb09. Now we turn to Portuguese term count based modeling. Since we have no labeled Portuguese data, we use the outcome of the English model as training labels. More precisely if a host has prediction less than pred_low $= 0.1$, then we consider as negative and if more than pred_high $= 0.9$, then positive training instance.

Algorithm 2 Stacked classification of mixed-language hosts based on an English model

for all hosts h **do**
  ratio[h] $=$ total frequency of top 10,000 Portuguese terms divided by total frequency of top 10,000 Portuguese and English terms
  if h is not testing **and** threshold_low $<$ ratio[h] $<$ threshold_high **then**
    pred[h] $=$ prediction for h based on the English model
    if pred[h] $<$ pred_low **then**
      Add h to negative training instances
    if pred[h] $>$ pred_high **then**
      Add h to positive training instances
Train a model based on Portuguese term counts using the positive and negative instances h
Classify all testing h using the Portuguese only model

In our algorithm for selecting the the intermediate training set, the procedure is made efficient by first running a MapReduce job only to count the dictionary

term distribution, and then compute features for the selected hosts only.

We also note that the procedure summarized by the scheme in Fig. 6.2c can be used with any classifier and feature set. In addition to training using Portuguese term frequencies, we also compute the public content based features and compare models trained on ClueWeb09 vs. the semi-supervised "training" set.

6.2.5 Classification Framework

In our classifier ensemble, we split features into related sets as described in Sections 6.2.1–6.2.3 and for each we use a collection of classifiers that fit the data type and scale. These classifiers are then combined by ensemble selection. We used the classifier implementations of the machine learning toolkit Weka [101]. We use a procedure similar to [34] that we summarize here.

In the context of combining classifiers for Web classification, to our best knowledge, ensemble selection was only used by one previous result [34]. Before that, only simple methods that combine the predictions of SVM or decision tree classifiers through logistic regression or random forest have been used [24]. We believe that the ability to combine a large number of classifiers while preventing overfitting makes ensemble selection an ideal candidate for Web classification, since it allows us to use a large number of features and learn different aspects of the training data at the same time. Instead of tuning various parameters of different classifiers, we can concentrate on finding powerful features and selecting the main classifier models which we believe to be able to capture the differences between the classes to be distinguished.

We used the ensemble selection implementation of Weka [101] for performing the experiments. We allow Weka to use all available models in the library for greedy sort initialization and use 5-fold embedded cross-validation during ensemble training and building. We set AUC as the target metric to optimize for and run 100 iterations of the hillclimbing algorithm.

We use the following model types for building the model library for ensemble selection: bagged and boosted decision trees, logistic regression, LogitBoost, naive Bayes and random forests. For most classes of features we use all classifiers and allow selection to choose the best ones. The exception is static term vector based features where, due to the very large number of features, we use SVM as described in Sections 6.2.3–6.2.4.

category	.pt count	ClueWeb count
AR	362	97355
BU	308	193678
CO	366	66159
RC	205	65594
SC	234	43317
SO	385	122084
SP	42	54456
spam	124	439
non-spam	3375	8421
hosts	686443	19228332
pages	71656081	502368557

Table 6.1: The number of positive host instances in each category and the host and page count for the two data sets.

6.3 Experiments

We evaluate the performance of the proposed classification approach on a 2009 crawl of the Portuguese Web Archive of more than 600,000 domains and 70M pages. For training our English language models, we used the English part of ClueWeb09 of approximately 20M domains and 500M pages. Web spam labels were provided by the Portuguese Web Archive and the Waterloo Spam Rankings [25], respectively. While the Waterloo Spam Rankings contain negative training instances as well, for the Portuguese data we used pages from the Open Directory Project (ODP) for this purpose.

We labeled hosts in both the `.pt` crawl and ClueWeb09 by top-level ODP categories (using links extracted from topic subtrees in the directory). The following English top level categories had corresponding Portuguese categories with sufficient number of labels in both collections: arts–Artes (AR), business–Negócios (BU), computer–Informática (CO), recreation–Passatempos (RC), science– Ciência (SC), society–Sociedade (SO) and sports–Desportos (SP). Out of all labeled hosts, 1902 Portuguese and 642643 English received a unique label. We note that certain sites as e.g. `bbc.co.uk` may belong to even all 14 top level English categories. We discarded the labels of all hosts with multiple labels. The resulting distribution of labels is shown in Table 6.1.

Just like in our previous experiments, we ran binary classifications for every topic separately. We evaluate our classifiers by the area under the ROC curve (AUC) [40] as used at Web Spam Challenge 2008 [17]. We do not give results

in terms of precision, recall, F-measure or any other measure that depends on the selection of a threshold as these measures are sensitive to the threshold and do not give stable comparison of two results. These measures, to our best knowledge, were not used in Web classification evaluation campaigns since after Web Spam Challenge 2007.

6.3.1 Feature distributions

As seen by the language distribution in Fig. 6.3, our Portuguese testing data set consists of host with English to Portuguese ratio uniformly spread between mostly English to fully Portuguese, with the exception of a large number of English only hosts. These latter hosts are however underrepresented in the labeled set, hence we take no specific action to classify them.

Since most often web sites are topically classified based on the strong signals derived from terms that appear on their pages, our first and often most powerful classifier is SVM over bag of words representation, averaged over all pages of the host. After stop word removal, we use the most frequent 10,000 terms both in English and in Portuguese.

The distribution of content features differs significantly between ClueWeb09 and the Portuguese crawl. As an example, the relative behavior of spam compared to normal hosts also significantly differs between ClueWeb09 and the Portuguese data as seen in Fig. 6.4. Hence we may not expect content based features to work well across models.

6.3.2 Computational Resources

For the experiments we used a 45-node Hadoop cluster of dual core machines with 4GB RAM each as well as multi-core machines with over 100GB RAM. Over this architecture we were able to compute all features, some of which would require excessive resources either when used by a smaller archive or if the collection is larger or if fast classification is required for newly discovered sites during crawl time. Some of the most resource bound features involve the multi-step neighborhood in the page level graph that already requires approximation techniques for WEBSPAM-UK2007 [16].

We use batch processing an entire collection is analyzed at once, a procedure that is typically applied when training a model on a collection.

The first expensive step involves parsing to create terms and links. The time requirement scales linearly with the number of pages. Since apparently a few hundred page sample of each host suffices for feature generation, the running time is also linear in the number of hosts.

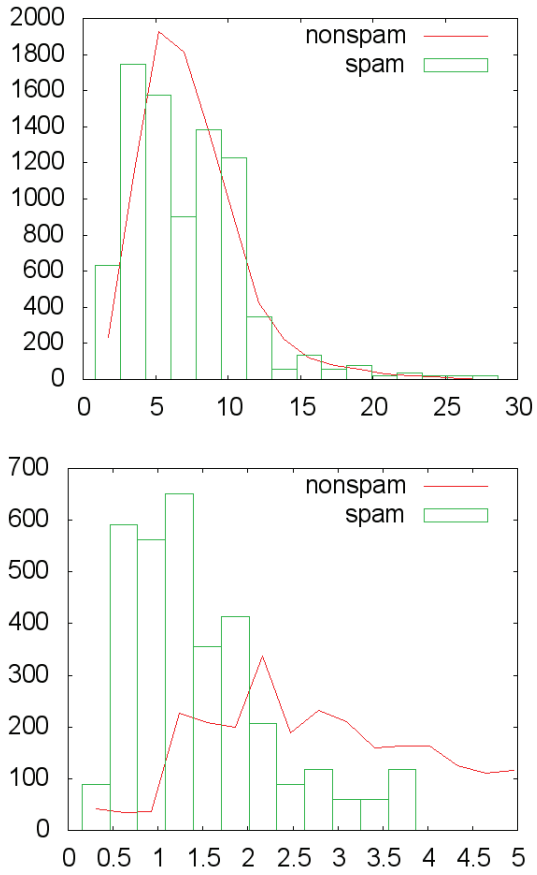


Figure 6.4: Distribution of the title length of the home page over the ClueWeb09 (top) and the Portuguese data (bottom), separate for spam and normal hosts.

category	content		link	English	translate	stacked	translate+stacked
AR	0.490	0.477	0.572	0.543	0.692	0.700	0.704
BU	0.486	0.517	0.583	0.428	0.707	0.771	0.745
CO	0.494	0.545	0.484	0.516	0.669	0.690	0.688
RC	0.492	0.515	0.525	0.457	0.731	0.727	0.740
SC	0.529	0.457	0.508	0.448	0.731	0.736	0.754
SO	0.509	0.527	0.569	0.497	0.656	0.674	0.677
SP	0.503	0.639	0.532	0.598	0.825	0.845	0.857
spam	0.719	0.751	0.921	0.752	0.861	0.894	0.900

Table 6.2: AUC of the main classification methods over the Portuguese test data. In the two variants of the content based features, we give results of the ensemble selection in the first and a single LogitBoost in the second column.

We have to be more cautious when considering the memory requirement for parsing. In order to compute term frequencies, we use Map-Reduce. Host level aggregation allows us to proceed with a much smaller size data. However for aggregation we need to store a large number of partial feature values for all hosts unless we sort the entire collection by host, again by Map-Reduce.

After aggregation, host level features are inexpensive to compute. The following features however remain expensive and require distributed or compressed graph algorithms:

- Page level PageRank. Note that this is required for all content features involving the maximum PageRank page of the host.
- Page level features involving multi-step neighborhood such as neighborhood size at distance k as well as graph similarity.

Training the classifier for a few 100,000 sites can be completed within a day on a single CPU on a commodity machine with 4-16GB RAM; here costs strongly depend on the classifier implementation. Our entire classifier ensemble took a few hours to train.

6.3.3 Results

We show our results in terms of the AUC measure for the Portuguese Web data set in Table 6.2. First we give results by using the public content and link based features [16]. These features work relative well for spam, however our ensemble selection results perform random for the ODP categories. Improved results are obtained by using LogitBoost only, as given in the second and third column of Table 6.2.

category	content	link
AR	0.677	0.561
BU	0.717	0.597
CO	0.615	0.589
RC	0.652	0.532
SC	0.711	0.614
SO	0.627	0.514
SP	0.654	0.511
spam	0.806	0.804

Table 6.3: AUC of the main classification methods crossvalidated over the ClueWeb09 data.

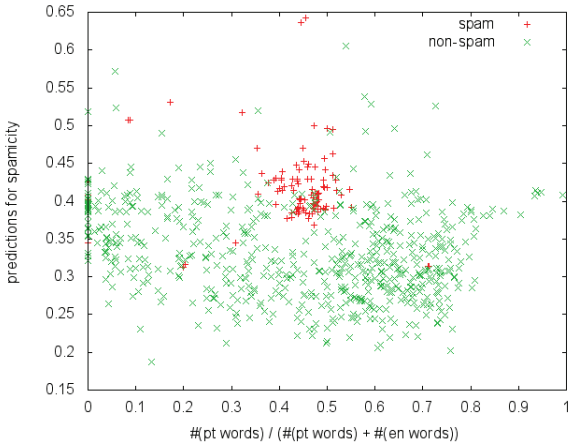


Figure 6.5: Predicted spamicity (vertical) as a function of the language distribution of most frequent terms (horizontal) in Web hosts over the `.pt` domain, separate for spam and normal hosts. For normal hosts we show a 20% random sample.

Method	AUC
English and Translated prediction average	0.952
Translated and Stacked prediction average	0.900
English, Translated and Stacked prediction average	0.952
Stacked and link based features classified with LogitBoost, both predictions normalized	0.964

Table 6.4: The results of the combined methods for Spam in AUC.

In order to validate the weakness and the relative power of content and link based features, we also give crossvalidation results over both the training and the testing set in Table 6.3.

Next we give our results based on the bag of words representation. The translation model works reliably over all categories as well as for spam. The fully English model performs near random overall. Somewhat surprisingly however if we use the English only predictions in the stacked framework of Section 6.2.4, we see significant improvement in all categories. The combination of the translated and stacked models obtained by averaging the predictions is the overall best method.

For spam classification, a single LogitBoost classifier over the link features trained over the ClueWeb09 corpus performs best over the Portuguese test data, even better than crossvalidated over the same data. This may be due to the fact that labeled spam comes from a relative small number of link farms and hence have a very characteristic link structure. Table 6.4 shows the results of the combination of various spam classifiers.

Note that unlike for ODP categories, combinations with the English only classifier improve spam classification. The best performing combination also involves the link based classifier.

As an illustration of how predictions behave as a function of the language distribution of the host, in Fig. 6.5 we show the predictions of the translated spam model separate for spam and a sample non-spam hosts. We observe that link based features work only for spam, just as for the Portuguese testing set. Content based features are however much more powerful over the same data as across data sets, as justified also by Fig. 6.4.

6.4 Conclusions

In this chapter we have demonstrated the applicability of cross-lingual Web host classification for spam and top-level Open Directory categories. Our experiments were tested over more than 600,000 hosts of the `.pt` domain by using the near 20M host English part of the ClueWeb09 data sets. Our results open the possibility for Web classification practice in national Internet archives who are mainly concerned about their resources, require fast reacting methods, and have very limited budget for human assessment.

By our experiments it has turned out that the strongest resource for cross-lingual classification consist of multilingual Web sites that discuss the same topic in both English and the local language. Note that these Web sites cannot be considered parallel corpora: we have no guarantee of exact translations, however, as our experiments also indicate, their content in different languages are topically identical. The use of dictionaries to transfer a bag of words based model also works and combine well with other methods. The normalization of the "public" Web spam content based features [16] across languages however seems to fail; also these features perform weak for topical classification. Link based features can however be used for language-independent Web spam classification, regardless of their weakness identified in [34].

6.5 My contribution

The results demonstrated in this chapter are joint work with András Garzó, Bálint Daróczy, Tamás Kiss and András Benczúr and extend the results published in [4] by giving improved models and experimenting with ODP categories in addition to Web spam classification. My contribution is running the classification methods on the link and content features, crawling and labeling the Portuguese sites with the DMOZ categories. Tamás Kiss implemented a distributed framework for calculating the link features, András Garzó implemented the hadoop jobs for calculating the content and BM25 features and he did the word by word translation of Portuguese sites. Bálint Daróczy ran the SVM over the BM25 features.

Chapter 7

Text classification via bi-clustering

Mining opinion from the Web and assessing its quality and trustworthiness became a well-studied area [29]. Known results typically mine Web data on the micro level, analyzing individual pages of blogs or even sections containing comments and reviews.

Our aim is to address a slightly different task of assessing the overall trust and neutrality of hosts, at Web scale. Instead of relying on the heavy machinery of opinion mining and sentiment analysis [77], methods that will likely not scale to the size of the Web, we analyze how a simple bag of words representation can be extracted to represent quality aspects.

In this chapter we summarize the lessons learned in the ECML/PKDD Discovery Challenge 2010 on Web Quality (DC2010) and present a first attempt for automatically classifying overall quality aspects. DC2010 introduced the tasks for assessing neutrality, bias and trustworthiness of Web hosts. Since these attributes constitute key aspects of Web quality, our goal is to improve the classification techniques for these tasks.

Participants [47, 4, 71] found the new quality tasks particularly challenging with AUC values, in all cases, below 0.6, typically even near the 0.5 value of a completely random prediction. The results we present in this paper greatly improve over those of the best participants, stable above the AUC value of 0.6, and present the first result that may be practically useful for this task.

TF-IDF, content and linkage features (see Section 3.4) were precompiled for the DC2010 participants and also in [34] but with little success for quality aspect classification.

As the bag of words representation turned out to describe Web hosts best for most classification tasks of the Discovery Challenge [34], we realized that new text classification methods are needed particularly suited to the quality related tasks in question. Such classifiers are however computationally expensive

including SVM that is generally considered to work well for text classification. For example [34] uses random forest, a suboptimal choice, since they were not able to run more expensive classifiers.

In order to both improve the quality and reduce the size of the problem, our motivation comes from image processing. Just as Web hosts consist of a collection of individual pages represented by their bag of words, images consist of regions or points of interest represented by high-dimensional image descriptors. Best performing content based image classification systems are typically based on the idea of soft clustering the set of regions and representing images by cluster histograms [79], also called bag of "visual words". The procedure, at the same time, reduces the size of the problem and removes the noise induced by individual outlier image regions.

Based on the above image classification motivation, we bi-cluster the host-term matrix in order to represent hosts by bag of concepts instead of words. As an additional advantage, the sparse bag of words data is turned to a dense continuous representation of cluster distances, a type of data best suited for SVM classifiers.

Finally for the SVM classification step, after our first discouraging experiments with the document similarity kernel over the distance matrix produced, we considered kernel selection methods [85] that also performed well for the above image classification task [28]. Since the dimensionality of the data is low and it turned out that the performance of various kernels can be measured over a small holdout set, we were able to perform a full comparison of kernel fusion methods from [85].

We compare our result over the ECML/PKDD Discovery Challenge 2010 data set, both with the best results of the participants [47, 4, 71] and with an earlier result [34] of our research group focusing primarily on spam classification. Our improvements in NDCG for bias is 20% over DC2010 best and 5% over [34]. For distrust we gain 20% and 10%, respectively, while for non-neutrality 3%. The AUC values of our various methods are convincingly above 0.6, i.e. we may say that we have reached the quality of the first results on Web spam a few years ago.

As an additional illustration of our methods, we also evaluated our classification techniques on the C3 data set of the WebQuality 2015 Data Challenge. The data set was created in the Reconcile[1] project and contains 22325 evaluations (five dimensions, among them credibility) of 5704 pages given by 2499 people. The mTurk platform were used for collecting evaluations.

While we are aware of no other results over the C3 data set, we collect reference methods from Web credibility research results. User and page-based

[1] `http://reconcile.pjwstk.edu.pl/`

collaborative filtering is suggested in [78] in combination with search engine rankings. We reuse our RecSys Challenge 2014 second place winner solution [76] to build a strong baseline method over the evaluator, site, evaluation triplets including the evaluator and site side information.

Social media and network based features appear already for Web spam [18, 51]. In a collection designed similar to C3 [74], social and general popularity and linkage were introduced and used for credibility assessment. Some of these features, in particular social media popularity, are used by the RecSys Challenge 2014 [76] as well and hence we deploy the methods we used there.

Content statistics as a concise summary that may replace the actual terms in the document were introduced first in the Web spam research [18]. The C3 data set includes content quality and appearance features described among others in [74].

Our best results reach the AUC of 0.74 for credibility, 0.81 for Presentation, 0.70 for Knowledge, 0.71 for Intentions and 0.70 for Completeness. We may hence say that all results reach the level of practical usability.

7.1 Related results

Our baseline classification procedures are collected by analyzing the results of the Web Spam Challenges and the ECML/PKDD Discovery Challenge 2010. A key ingredient of the Web Spam Challenge 2008 best result [43] was ensemble undersampling [23] while for earlier challenges, best performances were achieved by a semi-supervised version of SVM [1] and text compression [24]. Best results either used bag of words vectors or the so-called "public" feature sets of [18].

The Discovery Challenge 2010 best result [71] achieved an AUC of 0.62 for non-neutrality, 0.53 for bias and 0.506 for distrust classification while the overall winner [47] was able to classify at an average AUC of 0.80 but their results were below 0.52 for all the quality categories. As for the technologies, bag of words representation variants proved to be very strong for the English collection. For classification techniques, a wide selection including decision trees, random forest, SVM, class-feature-centroid, boosting, bagging and oversampling in addition to feature selection (Fisher, Wilcoxon, Information Gain) were used [47, 4, 71]. Note that the findings of the usability of feature selection in case of high class imbalance in [71] is similar to our present work. In [34] Erdélyi et al. improved over the best results of the Challenge participants; the best performing ingredient of their classifier ensemble was a random forest classifier over a BM25 weighted bag of words representation of the hosts.

Linkage was considered to be of high importance in results for Web spam

filtering [16]. The recent classification experiments over DC2010 [34] however indicate little use of these features, whence we omit them in the present work. As a possible reason of observing performance lower than for other collections, the DC2010 training and test set was constructed by handling of hosts from the same domain and IP. Since no IP and domain was allowed to be split between training and testing, the simple and otherwise easy connections such as same ownership do not show up in the DC2010 linkage.

If sufficiently many evaluators assess the same Web page, which is the case with the C3 data set, one may consider evaluator and page-based collaborative filtering [78] for credibility assessment. In this setting, we face a dyadic prediction task where rich meta data is associated with both the evaluator and especially with the page. The Netflix Prize competition [11] put recommender algorithms through a systematic evaluation on standard data [8]. The final best results blended a very large number of methods whose reproduction is out of the scope of this experiment. Among the basic recommender methods, we use matrix factorization [60, 95]. In our experiments we use the factorization machine [87] as a very general toolkit for expressing relations within side information. Recently, the RecSys Challenge 2014 run a similar dyadic prediction task where Gradient Boosted Trees [105] performed very well [76].

7.2 Information theoretic bi-clustering

Bi-clustering is a bidirectional clustering algorithm that is capable of clustering along multiple aspects at the same time by switching between clustering along the two axis. Bi-clustering explores a deeper connection between instances and attributes than the usual one-directional clustering methods.

Our bi-clustering method is based on Dhillon's information theoretic co-clustering algorithm [30]. The basic idea is to consider the data as a joint distribution. Now with the help of information theory we can reformulate the clustering problem as follows: The optimal bi-clustering is the one where the mutual information (see Section 2.7) of row and column clusters is the largest.

7.2.1 The algorithm

Let X and Y be discrete random variables that take values in the sets $\{\text{rows}\}$ and $\{\text{columns}\}$ respectively. Let $p(X, Y)$ denote the joint probability distribution of X and Y. Let the k clusters of X be $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$, and let the ℓ clusters of Y

be $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\ell\}$. We are interested in finding partition functions C_X and C_Y,

$$\begin{aligned} C_X \colon \{x_1, x_2, \ldots, x_n\} &\mapsto \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}, \\ C_Y \colon \{y_1, y_2, \ldots, y_m\} &\mapsto \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\ell\}. \end{aligned}$$

For brevity we write $\hat{X} = C_X(X)$ and $\hat{Y} = C_Y(Y)$, where $\hat{X}$ and $\hat{Y}$ are random variables that are a deterministic function of X and Y, respectively. Finally let $KL(p \parallel q)$ denote the ***Kullback-Leibler*** divergence (see Section 2.7.3) of probability distributions p and q.

Algorithm 3 Co-clustering algorithm

Let $t = 0$. Start with some random initial $C_X^{(0)}$ and $C_Y^{(0)}$ partition functions.
while condition **do**

for all $x \in X$ **do**
find the new cluster of x as

$$C_X^{(t+1)}(x) = \text{argmin}_{\hat{x}} \; KL\left(p(Y \mid x) || p(Y \mid \hat{x})\right)$$

Let $C_Y^{(t+1)} = C_Y^{(t)}$, let $t = t + 1$
for all $y \in Y$ **do**
find the new cluster of y as

$$C_Y^{(t+1)}(y) = \text{argmin}_{\hat{y}} \; KL\left(p(X \mid y) || p(X \mid \hat{y})\right)$$

Let $C_X^{(t+1)} = C_X^{(t)}$, let $t = t + 1$

Dhillon proves in [30] that the above algorithm never decreases the mutual information of row and column clusters, hence it converges to a local maximum.

We apply Dhillon's algorithm with one modification: we replace Kullback-Leibler divergence by Jensen-Shannon, its symmetric version. It is easy to see that the proves in Dhillon's paper are also work with JS divergence. The motivation behind this modification is that the KL divergence of two documents without any common words is zero. Meanwhile JS is capable to measure the distance of such two documents. This capacity of JS divergence is very useful when we have a sparse data. By our experience over several other data sets, this slight modification greatly improves cluster quality.

7.3 Classification Framework

The key ingredients of our method are as follows:

- We compile around 500–3000 bag of concepts from words via bi-clustering (see Section 7.2). This low dimensional representation allows computationally costly classifiers, in particular SVM, to be applied. Important to note that unlike in the original bi-clustering method [30] that uses Kullback-Leibler, we use Jensen-Shannon (see Section 2.7) divergence that greatly improves the quality of the final prediction.
- We use simple feature selection and weighting based on frequencies in the training set. Although insufficient in itself, the importance of feature selection already pointed out by DC2010 participants [71]. Unlike for all other categories of spam and genre, this method is particularly suited to the highly imbalanced classes of non-neutrality, bias and distrust.
- Given the compact representation of hosts by cluster distances, we may apply computationally expensive methods for classification. We use SVM with several kernels and compare the use of early and late kernel fusion. For SVM we use the libSVM implementation [21].

7.3.1 Bi-clustering

In our baseline term selection method, we selected the most frequent terms as vocabulary. For efficiency considerations, we selected the top 20,000-40,000 terms, even lower than in the DC2010 official data. By our experiments the vocabulary size of 30,000 performs well in the overall procedure.

Since Dhillon's [30] method is based on information theoretic distances, the raw TF values give best performance for bi-clustering. Normalized versions such as TF-IDF or the BM25 weighting scheme performs significantly worse and is omitted for further consideration.

We applied a simple supervised feature weighting over the same 30000 size vocabulary by selecting terms with increased frequency in the positive instances of the training set. This simple idea results in large gains for our three categories of interest while negligible improvement or even deterioration for spam and genre. One reason could be the rarity of positive instances in these categories, even lower than in spam. Another reason could be that non-neutrality, bias and distrust depends on special, less frequent terms since these concepts are mostly independent of genre.

For feature selection we computed the ratio of overall TF and the TF of the positive instances. For the highest weight terms, we used the category TF multiplied by a constant as the new weight and united the terms for all categories. Whenever one term is selected for more than one categories, we choose the lower weight.

We optimized the parameters of bi-clustering over the same holdout data set that was used for the Discovery Challenge to evaluate submissions before

yorkie adorable puppy teacup capuchin affectionate parrots maltese puppies lovely cute
serbia croatia bosnia albania montenegro macedonia herzegovina belarus moldova kosovo
welcome tel fax submit home please mail click contact reserved
plated earrings necklace pendants necklaces bracelets studs jewelry jewellery
google advertising real category
laptops cheap discount buy
yeah awesome folks wondering okay yes nice maybe pretty hello guys wow guess
tabs erectile erection pfizer impotence generic
shopping enlarge coupon price
cant lol reply thats btw xd alot logged offline dont pm smf

Table 7.1: Example term clusters found by our bi-clustering algorithm without term weighting.

the final evaluation. We reached the best results with 1000 word-, and 1000 document-clusters. We used 20 iterations to typically reach a level of cluster weight changes below 1%. Notice that we reduced the original 20 to 40 thousand dimensions to a mere 1000, hence the possibilities for choosing classifiers is no longer restricted by scalability as it is in the initial bag of words representation.

In our experience the term clusters carry clear meaning as summarized in Table 7.1, especially the ones selected when weighting by increase of TF in the three quality categories as in Table 7.2. We note that in addition we found quite a few high weight single-word or few word clusters including eBay, image, friend, lifestyle etc.

7.3.2 Kernel methods

While SVM is widely used for text classification, the aggregation of several kernels forms the challenging and typically computationally expensive task of Multiple Kernel Learning (MKL). Multiple kernels may arise not only by the large choice for a single feature space, but also from different transformations and feature subsets. Since we define several kernels based on the terms themselves, on the content features, and also on the distances from bi-cluster centers, our learning procedure builds upon MKL procedures by also taking efficiency into consideration.

We used a wide range of basic kernels over both the original term and the cluster distance vectors. Our kernels include linear, polynomial and radial basis function with different parameters as seen in Table 7.3 with n denoting the dimension, $d \in \{1, 2, 4\}$, and $\gamma = \frac{1}{|D|}$ where D is the training set. For kernel combination we applied various cost parameters for the linear kernel.

non-neutral	buyer rumor guru undertake unreleased donkey medications psionic catalyst imam dragon rabinic cleric blame preamble adviser lordship domestication fortify memoir ritual dogma roma conservative equivalence indemnity bombardier proletariat alien
biased	mitigation affirmative schizophrenia aboriginal verdict proprietary moose fiscal ashore monitor protagonist destiny nursery whisky inert minor fault anarchist lufthansa subnational extracellular chaos sahara bulldog bolshevik chew dispute
distrusted	compound vomit mistaken bombay rain trust messianic petrochemical skyscraper lakers admiralty sinus heroin panoramic genuine fix standardize sussex bright methodist registered fia transport resign

Table 7.2: Term clusters with emphasis on terms more frequent over quality flagged hosts. First few terms shown ordered by decreasing weight for some clusters of high correlation with the category.

	Kernel function
linear	$K(x, y) = x' * y$
polynomial	$K(x, y) = (\frac{1}{n}x' * y)^d$
radial basis function	$K(x, y) = e^{(-\gamma(x-y)^2)}$

Table 7.3: Kernel functions and parameters.

Beside the original vectors we also transformed them into a common feature space. Distance based feature transform for classification using the training set is a well-known technique. In [90] it is shown that a class of kernels can be represented as norm-based distances in Hilbert spaces. We can represent the k-th feature vector of an instance $x \in X$ as

$$L_k(x, D) = [sim_k(x_k, D_{k_1}), \ldots, sim_k(x_k, D_{k_{|D|}})] \tag{7.1}$$

where D is the set of the training instances with corresponding set of vectors D_k and sim_k denotes the selected similarity measure on the k-th basic vector.

One of the main advantages of this representation is the fact that the dimension of the transformed feature space is independent from the dimension of the original feature space. Using this property we can define a combined similarity vector of an instance $x \in X$ as

$$L_R(x, D) = \frac{1}{\mathbb{E}_D[\mathrm{e}^{-sim_R(x,D)}]}[\mathrm{e}^{-sim_R(x,D_1)}, \ldots, \mathrm{e}^{-sim_R(x,D_{|D|})}] \tag{7.2}$$

where R is the set of basic representations, the normalization constant is the expected value of the similarity values over the training set and the unnormalized similarity to the i-th training instance is

$$sim_R(x, D_i) = \sum_{k \in R} \beta_k sim_k(x_k, D_i). \tag{7.3}$$

where $\sum \beta_k = 1$.

Next, given the variety of kernels at hand, we overview the notion and the methods of Multiple Kernel Learning (MKL). The dual optimization problem of the standard Support Vector Machine (SVM) [26] with kernel $K(x_i, x_j)$ is

$$\text{Maximize } L_{Dual}(\alpha) = \sum_{i=1}^{|D|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|D|} \sum_{j=1}^{|D|} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{7.4}$$

subject to $\sum_{i=1}^{m} y_i \alpha_i = 0$ for all i with $\alpha_i \geq 0$.

With multiple kernels, the simplest method is to set aside a holdout set and select the best performing kernel over the holdout. *Late fusion* is an improved idea where we select the best linear combination weight of the SVM output over the holdout, i.e.

$$pred_{late}(x) = \sum_{n=1}^{N} \beta_n (\sum_{i=1}^{|D|} \alpha_{in} K_n(x, y_i) + b_n)$$

where $K_n(x, y)$ is the n-th kernel function and α_{in} and b_n form the optimal solution for the n-th SVM classifier as in (7.4).

Instead of late fusion, we may perform *early fusion* by combining the kernels before optimizing (7.4). We obtain the Multiple Kernel Learning (MKL) problem by linearly combining the kernels inside (7.4) with weight β_n as

$$\text{Maximize } L_{Dual}(\alpha, \beta) = \sum_{i=1}^{|D|} -\frac{1}{2} \sum_{i=1}^{|D|} \sum_{j=1}^{|D|} \alpha_i \alpha_j y_i y_j \sum_{n=1}^{N} \beta_n K_n(x_i, x_j) \quad (7.5)$$

subject to $\sum y_i \alpha_i = 0$ for all i with $\alpha_i \geq 0$, where N is the number of the basic kernels. In [85], a wrapper method is used for MKL where in each iteration they solve a standard SVM dual problem and update the weights of the basic kernels. Since they observe the high computational cost of the problem, they primarily concentrate on reducing the number of kernels, i.e. gradually setting most of the β_n to 0.

Compared to the sparse wrapper method, we only have a few but complementary kernels that all contribute to the classification quality and our task is to determine their relative weight β_n.

To avoid the high computational costs of a wrapper method we used late fusion to determine the weights of the basic kernels. In order to avoid overfitting, we use only 0-1 weight combinations after normalizing the kernels by their average value. We combine the kernels according to the ideal weight over the holdout, and let the final prediction for test instance x be

$$pred_{early}(x) = \sum_{i=1}^{|D|} \alpha_i \sum_{n=1}^{N} \beta_n K_n(x, y_i) + b \quad (7.6)$$

where $K_n(x, y)$ is the nth kernel function and b is the bias.

7.3.3 Gradient Boosted Trees and Matrix factorization

We also compare our results with Gradient Boosting Trees [105] and matrix factorization methods on the user and C3 data features. We used two different matrix factorization techniques. The first one is a traditional matrix factorization method [60], while the second one is a simplified version of Steffen Rendle's LibFM algorithm [87]. Both techniques use stochastic gradient descent to optimize for mean-square error on the training set. LibFM is particularly designed to use the side information of the evaluators and the pages.

7.4 Experiments

7.4.1 Data sets

In this chapter we use the DC2010 data set created for the ECML/PKDD Discovery Challenge 2010 and the C3 data set of the WebQuality 2015 Data Challenge.

The DC2010 data set

The data set is described well in [34]; here we explore the important aspects with respect to neutrality, trust and bias. We attempt to analyze how assessors flagged the quality issues, show examples and the distribution of flags by genre.

DC2010 is a large collection of annotated Web hosts labeled by the Hungarian Academy of Sciences (English documents), Internet Memory Foundation (French) and L3S Hannover (German). The base data is a set of 23M pages in 190K hosts in the `.eu` domain crawled by the Internet Memory Foundation early 2010.

The manually created labels included assessment for genre and quality. The motivation behind the labeling procedure was the needs of a fictional Internet archive that may or may not want to completely exclude spam but may prefer certain type of content such as News-Editorial and Educational beyond Commercial sites. Also they may give higher priority to trusted, factual and unbiased content that combine to a utility score.

The DC2010 data set includes hosts labeled by several attributes, out of which spam, trustworthiness, factuality, bias and five genre was selected to be used for classification. While no further labeling is made for a spam host, other properties and in particular the five genre News/Editorial, Commercial, Educational, Discussion and Personal are non-exclusive and hence define nine binary classification problems. We consider no multi-class tasks in this work.

As a specific task of the ECML/PKDD Discovery Challenge, a special utility was defined to measure the quality of the content as a combination of the specific labels. Spam had zero utility, News/Editorial and Educational had highest utility, followed by Discussion and then by Commercial and Personal. Hosts flagged for bias, non-neutrality or distrust received a lower but still positive utility score. In our experiments we also show improved results for ranking by the utility for quality.

Next we summarize assessor instructions concentrating on the three labels relevant for our present work. First, assessors were instructed to check some obvious reasons why the host may not be included in the sample at all, including adult, mixed, language misclassified sites, and then to assess spam. These hosts were skipped for the remaining steps and in particular spam has no bias or trust

Label	Yes	Maybe	No
spam	423		4 982
News/Editorial	191		4 791
Commercial	2 064		2 918
Educational	1 791		3 191
Discussion	259		4 724
Personal-Leisure	1 118		3 864
Non-Neutrality	19	216	3 778
Bias	62		3 880
Dis-Trustworthiness	26	201	3 786
Confidence	4 933		49

Table 7.4: Distribution of assessor labels in the DC2010 data set.

label.

Hosts were labeled by genre into five categories, news/editorial, commercial, educational, discussion and personal. Important is that discussion spaces are not assessed for bias, i.e., just as spam, skipped for both training and testing. Discussion spaces include dedicated forums, chat spaces, blogs, etc., but comment forms were excluded. We also introduced the distinct category of Personal/Leisure covering arts, music, home, family, kids, games, horoscopes etc. A personal blog for example belongs both here and to "discussion" (and hence not labeled for bias).

Finally, general properties related to trust, bias and factuality were labeled along three scales:

1. Trustworthiness: I do not trust this—there are aspects of the site that make me distrust this source. I trust this marginally—looks like an authoritative source but its ownership is unclear. I trust this fully—this is a famous authoritative source (a famous newspaper, company, organization).
2. Neutrality: Facts—I think these are mostly facts. Fact & Opinion—I think these are opinions and facts; facts are included in the site or referenced from external sources. Opinion—I think this is mostly an opinion that may or may not be supported by facts, but little or no facts are included or referenced.
3. Bias: We adapted the definition from Wikipedia[2]. We flagged flame, assaults, dishonest opinion without reference to facts.

The distribution of labels is given in Table 7.4. For Neutrality and Trust the strong negative categories have low frequency and hence we fused them with the

[2] http://en.wikipedia.org/wiki/NPOV

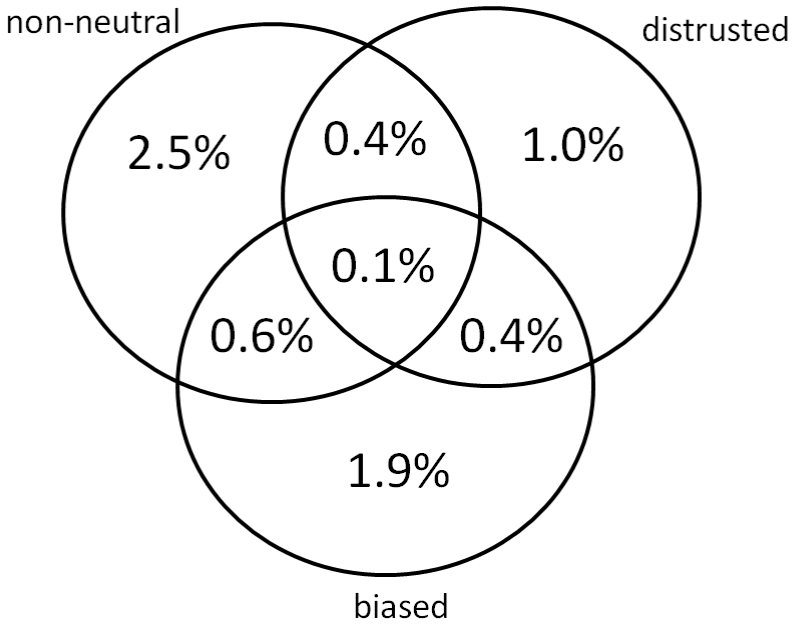


Figure 7.1: Overlap of the quality flags, shown as percentage of all hosts.

intermediate negative (maybe) category for the training and testing labels. We also remark that the assessors introduced subjectivity for judging trust: German assessors gave the intermediate maybe as default and yes, no only occasionally. For other assessors, yes was the default value as indicated in Table 7.4. We use the labels of English sites only and thus avoid this bias.

We may also notice that the three types of quality flags are rare and non-independent. In Fig. 7.1 we give the fraction of double and triple flagged hosts. Although the three labels statistically dependent, the majority for each category has a single flag, which shows weak correlation and makes the classification task hard.

When considering the ratio of quality flagged content by genre, we observe in Table 7.5 that non-neutral, biased and distrustful content is less likely among commercial and educational sites but these quality flags are orthogonal to genre. News and editorial sites often reflect one-sided opinion flagged as non-neutral and biased. Less often, but educational sites may also cover topics of controversy such as religion or environmental protection. Since multiple genre is allowed, some of the biased educational sites are also labeled news/editorial. Some commercial sites are also distrustful, such as a service for directing traffic towards adult content. Since the site itself contained financial figures such as average

Label	Non-Neutral	Biased	Distrust
News/Editorial	9%	6%	7.5%
Commercial	2.3%	2.3%	2.7%
Educational	2.2%	1%	1%
Discussion	5.3%	$*$ 5.3%	1.5%
Personal-Leisure	8%	2.2%	1.8%

Table 7.5: Distribution of quality flags by genre in the DC2010 data set. Discussion spaces are not flagged for bias; the starred cell ($*$) consists of hosts where assessors disagreed in genre and some assessors labeled Discussion while others labeled biased non-Discussion.

click revenue and no explicit content, both assessors agreed that this is a non-adult, commercial, yet distrustful site.

The C3 data set

The C3 data set consists of 22325 Web page evaluations in five dimensions (credibility, presentation, knowledge, intentions, completeness) of 5704 pages given by 2499 people. Ratings are similar to the data set built by Microsoft for assessing Web credibility [91], on a scale of four values 0-4, with 5 indicating no rating. The distribution of the scores for the five evaluation dimensions can be seen in Figure 7.2. Since multiple values may be assigned to the same page, we aggregate the human evaluations over the pages.

Since earlier results [78] suggest the usage of collaborative filtering along the page and evaluator dimensions, we measure the distribution of the number of evaluations given by the same evaluator and also for the same site in Figure 7.3.

Distribution of the variance of the ratings is shown via a heatmap of all pairs of ratings given for the same page and same dimension in Figure 7.4.

In order to perform text classification we crawled the pages listed in C3 data set. Distribution of HTTP response statuses are shown in Figure 7.5. Note that 65% of the C3 URLs returned OK HTTP status but 7% of them could no longer be crawled. Redirects reached over 20% that we followed and substituted for the original page.

7.4.2 Evaluation metrics

The standard evaluation metrics since Web Spam Challenges [17] is the area under the ROC curve (AUC) [40] described in Section 2.5.2. The ECML/PKDD

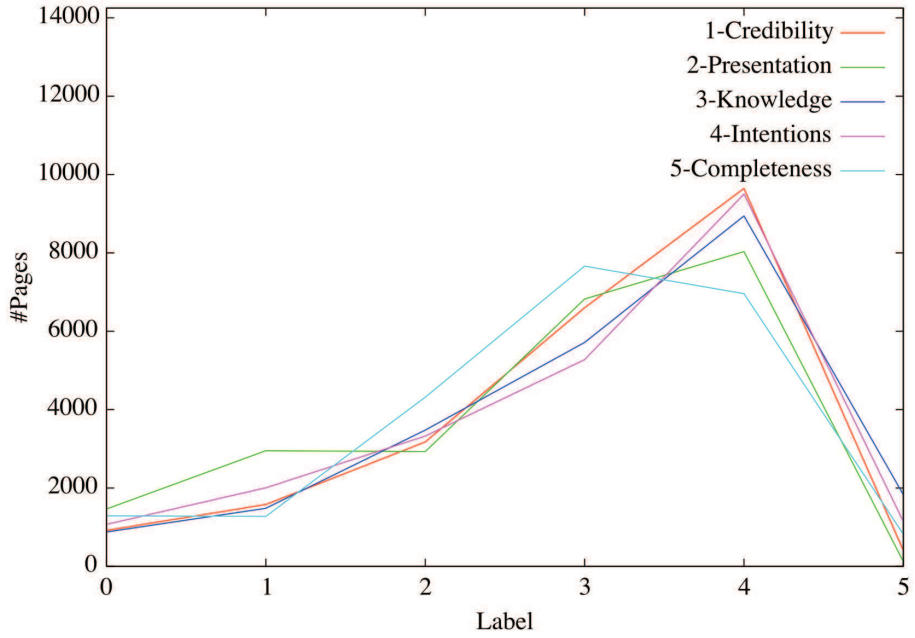


Figure 7.2: The distribution of the scores for the five evaluation dimensions.

Discovery Challenge used Normalized Discounted Cumulative Gain (NDCG, see Section 2.5.3) for evaluation since some tasks used multi-level utility based on spamicity, genre and other attributes. For the binary classification problems we use 1 for a "yes", 0 for a "no" label as utility. These measures perform very similar, even numerically [34]. We describe the version of NDCG applied for DC2010.

To emphasize performance over the entire list, the discount function is changed from the common definition to be linear

$$1 - i/N \tag{7.7}$$

where N is the size of the testing set. To justify the discount function, note that an Internet archive that may crawl 50% or even more of all the host seeds they identify and spam may constitute 10-20% of all the hosts. Our final evaluation formula is

$$\begin{aligned} \text{NDCG} &= \frac{\text{DCG}}{\text{Ideal DCG}}, \text{ where} \\ \text{DCG} &= \sum_{\text{rank}=1}^{N} \text{utility(rank)} \cdot \left(1 - \frac{\text{rank}}{N}\right), \end{aligned} \tag{7.8}$$

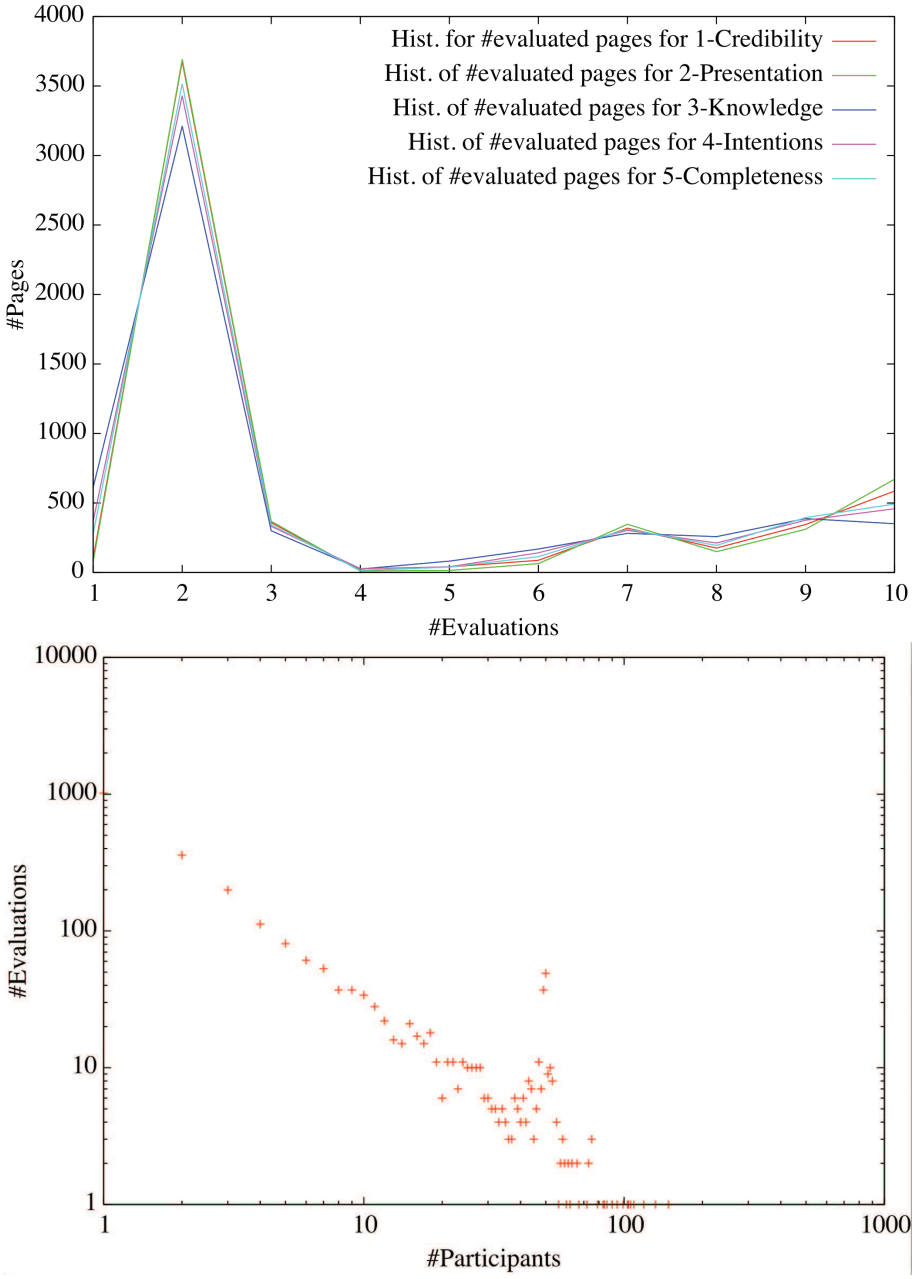


Figure 7.3: The distribution of the number of evaluations given by the same evaluator (**top**) and for the same site (**bottom**).

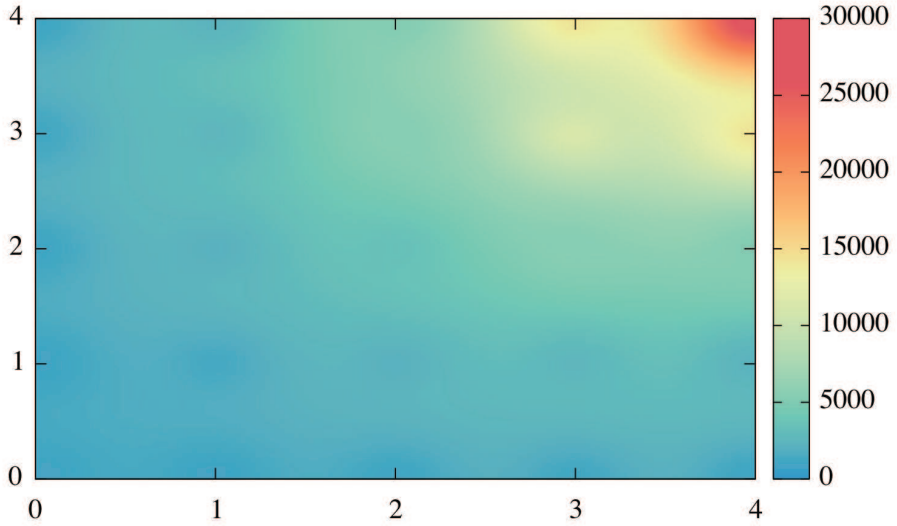


Figure 7.4: The number of pairs of ratings given by different assessors for the same aspect of the same page.

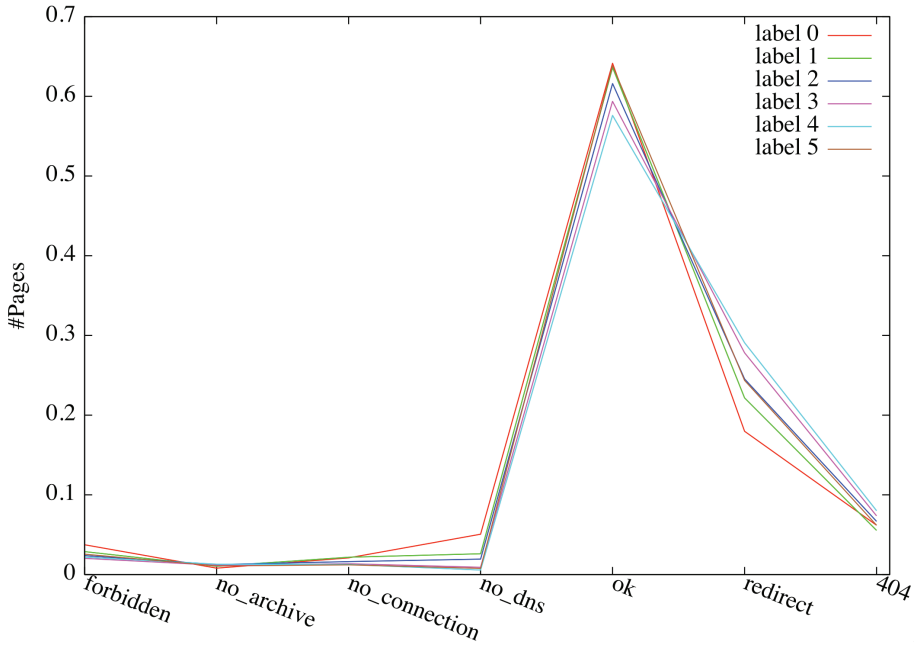


Figure 7.5: The number of pages with the given crawl status.

and Ideal DCG is obtained with utility decreasing with rank. We computed NDCG by the appropriate modification of the python script used by the Yahoo! Learning to Rank Challenge 2010 [22]. We also note here that NDCG and AUC produced numerically very close values on the Discovery Challenge binary problems. The reason may be that both measures show certain symmetry over the value 0.5, although the NDCG for an order and its reverse does not necessarily add up to one due to the normalization in NDCG.

Considering that on the C3 data set the translation of quality assessments into binary values is not so obvious, in addition to AUC we also have our regression methods evaluated by Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) defined in Section 2.5.4.

7.4.3 Results over the DC2010 data set

In this section we describe our various SVM ensemble methods over the traditional bag of words as well as the cluster distance vector representations. We measure the accuracy of various methods and their combinations. The summary of the best performing methods is found in Table 7.6. Although we concentrate on neutrality, bias, trust and overall quality, we give results for spam and genre average for the Discovery Challenge 2010 categories to give a better comparison of the techniques used.

For training and testing we use the official official DC2010 set as described in Table 7.4. As it can be seen, the quality categories show considerable class imbalance, which makes the classification problem harder. We solved the quality task by a simple weighted linear combination of the nine individual classifiers (spam, five genre, and three quality) as

$$\begin{aligned}\text{quality} = \quad & \text{news} + \text{educational} - \text{commercial} - \text{personal} - 3 \cdot \text{nonneutral} \\ & -3 \cdot \text{bias} - 3 \cdot \text{distrust} - 20 \cdot \text{spam}.\end{aligned}$$

Since the majority of the hosts have high utility, it is very easy to reach NDCG scores near 1. By using the output of the spam filter only, one may reach an NDCG 0.927, close to the Discovery Challenge winner result. The importance of our gains is an improved list of low quality sites at the low utility end that could have been better emphasized by a reverse order, blacklist utility. For comparability with the Discovery Challenge results, we score our results by the original evaluation.

We have collected the best runs from all DC2010 participants in the first row of Table 7.6. While genre and utility comes from the winners [47], the high imbalance classes including both spam and quality, those of key importance for us, were treated best by Wilcoxon feature selection [71]. From a previous result

		spam	genre average	(non)neutral	biased	(dis)trusted	**quality average**	***quality utility***
DC2010 best	AUC	0.830	0.734	0.626	0.558	0.506	0.563	—
	NDCG	0.833	0.740	0.620	0.553	0.510	0.561	0.936
best	AUC	0.891	0.808	0.618	0.653	0.582	0.612	—
[34]	NDCG	0.893	0.811	0.624	0.656	0.586	0.617	n/a
BM25	AUC	0.876	0.805	0.580	0.653	0.520	0.584	—
	NDCG	0.879	0.834	0.587	0.656	0.524	0.589	0.931
weighted bi-cluster	AUC	0.849	0.798	**0.695**	0.566	**0.640**	0.634	—
polynomial kernel	NDCG	0.852	0.827	**0.700**	0.571	**0.643**	0.638	0.927
late fusion	AUC	0.889	0.825	0.676	0.641	0.625	0.648	—
poly. and sim. kernel	NDCG	0.891	0.851	0.681	0.645	0.629	0.652	0.939
early fusion	AUC	0.880	**0.836**	0.595	0.642	0.545	0.594	—
polynomial kernel	NDCG	0.882	**0.860**	0.601	0.646	0.549	0.599	0.931
early fusion	AUC	**0.892**	0.811	0.689	**0.667**	0.626	**0.661**	—
similarity kernel	NDCG	**0.894**	0.839	0.694	**0.671**	0.629	**0.665**	**0.940**

Table 7.6: Performance summary of the best methods over the DC2010 labels in terms of AUC and NDCG as in equation (7.8).

[34], we show the best run for each category, as well as the result of the random forest classifier over the BM25 weighted bag of words representation, a stable well performing method.

Our best performing single method (weighted bi-cluster polynomial kernel in Table 7.6) itself outperforms all earlier methods for non-neutrality, distrust and average quality. For this method, first we selected the best bi-cluster parameters (500 clusters, 30,000 terms), then we used the best term weighting scheme (tf increase in the quality flagged categories), and finally the best of the three kernels (polynomial as opposed to linear and similarity). The final best methods use early or late fusion of the bi-cluster and BM25 kernels. The early fusion of similarity kernels performs best for both spam and average quality. The single kernel best method becomes overall best for non-neutrality and distrust, however the combinations perform near as good and better for quality on average.

Effect of bi-cluster parameters

We tested how the accuracy of our predictions depend on the parameters of the bi-clustering procedure. In Fig. 7.6 we show average performance for the three quality flags (non-neutrality, bias, distrust) for 500, 1000, 2000 and 3000 clusters of Web hosts by keeping the number of term clusters fixed to 1000. The number

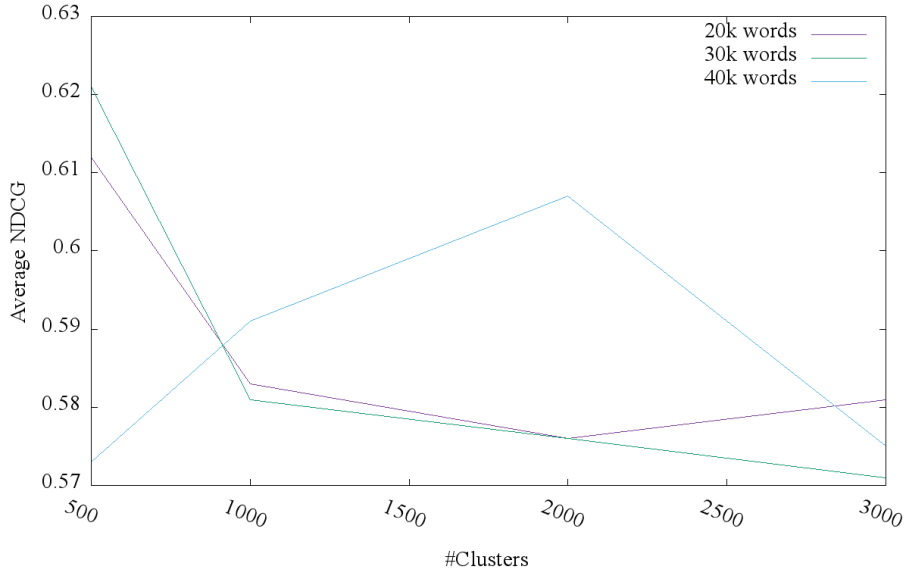


Figure 7.6: Average NDCG for the three quality labels, as the function of the number of document clusters and size of the vocabulary.

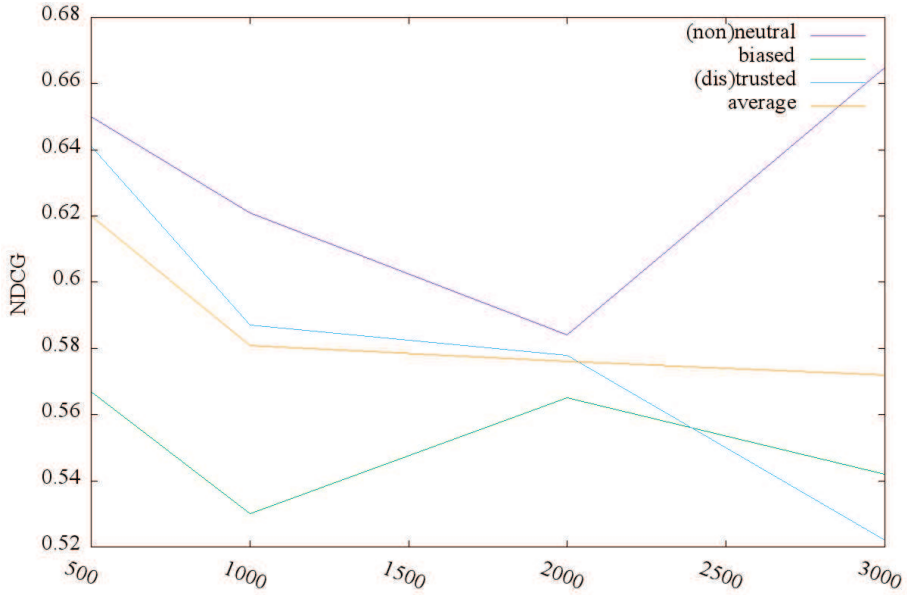


Figure 7.7: NDCG for the three quality labels, as the function of the number of document clusters.

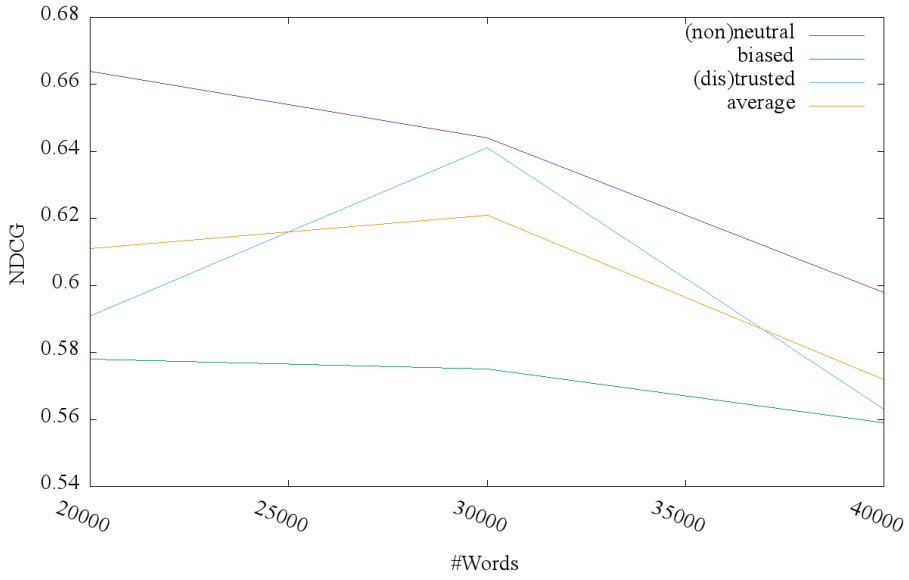


Figure 7.8: NDCG for the three quality labels, as the function of the vocabulary size.

of words used (20, 30 and 40 thousand) is also shown. As it turns out, a low number of clusters work best with a vocabulary of size 30,000.

We also note the fluctuation in the classification results for individual categories. In Fig. 7.7 we see that on average, performance decreases with increasing the number of clusters with individual variance in the categories. Similarly in Fig. 7.8 we observe that 30,000 words perform best on average because of the distrust category with a general trend of degradation with increasing vocabulary size.

	spam	genre average	(non)neutral	biased	(dis)trusted	**quality average**
unweighted	0.818	0.808	0.649	0.571	**0.644**	0.621
quality flagged	**0.852**	**0.827**	**0.700**	**0.571**	0.643	**0.638**
quality positive+negative	0.743	0.800	0.620	0.526	0.535	0.560
all labels	0.658	0.698	0.570	0.475	0.568	0.538

Table 7.7: Performance of the three different term weighting schemes and the baseline, in terms of AUC.

In Table 7.7 we also compare different methods for supervised term weighting as input to the bi-clustering algorithm. After selecting the best bi-cluster parameters (30,000 terms and 500 clusters), we weighted terms by the ratio of overall TF and the TF of selected hosts. We made three types of selection for hosts: we selected the three quality flagged hosts, these hosts and a negative sample, finally all nine labeled categories. The best performing method is to select terms that appear more frequently in the quality flagged classes. Surprisingly, this holds not just for quality but also for spam and even genre. This means that terms that appear frequently in distrusted, biased documents contribute a lot to automatically identifying the topic and quality of Web hosts.

Performance of different kernels

Regarding kernel methods, first we experimented with single kernels over the best bi-cluster output (500×1000 clusters, 30,000 terms). In Table 7.8 we show bi-cluster results both with terms unweighted and weighted with the three possible kernels. For weighting we use the best performing method using positive instances for non-neutrality, bias and distrust. We observe large variance in these results: different labels perform different. While unweighted methods occasionally outperform the weighted ones, we consider these results noise since the weighted counterparts perform near as good and better on average. Overall, we expect the combination of the polynomial and similarity kernels perform best.

Finally, we blend different bi-cluster and BM25 kernels to obtain the best results in Table 7.6. In order to avoid overfitting, we always combine two bi-cluster and one BM25 kernels with equal weight, after normalizing the kernel values between 0 and 1. Best overall methods are obtained by early fusion, justifying the applicability of the multiple kernel learning theory.

7.4.4 Results over the C3 data set

The detailed results are in Table 7.9, in four groups. The first group gives the baseline methods. Below, we show the results of the similarity kernel methods. In the third group we combine multiple similarity kernel methods by early fusion strategy. Finally, in the last group, we combine our methods by simply taking the average of the predictions after standardization.

C3 data attributes

For user and item features we experiment with GraphLab Create[3] [64] implementation of Gradient Boosted Tree and matrix factorization techniques. In case

[3] `http://graphlab.com/products/create/`

	spam	genre average	(non)neutral	biased	(dis)trusted	**quality average**	***quality utility***
bi-cluster linear kernel	0.812	**0.852**	0.595	0.511	0.619	0.575	0.922
weighted bi-cluster linear kernel	0.817	0.844	0.576	0.500	0.573	0.550	0.921
bi-cluster polynomial kernel	0.818	0.808	0.649	0.571	**0.644**	0.621	0.923
weighted bi-cluster polynomial kernel	0.852	0.827	**0.700**	0.571	0.643	**0.638**	0.927
bi-cluster similarity kernel	0.864	0.787	0.658	0.592	**0.644**	0.631	**0.935**
weighted bi-cluster similarity kernel	**0.869**	0.795	0.680	**0.600**	0.610	0.630	0.934

Table 7.8: Performance of the bi-cluster kernels in terms of NDCG.

of the gradient boosted tree algorithm (GBT) we set the maximum depth of the trees to 4, and the maximum number of iterations to 18. To determine the advantage of additional side information over the original matrix factorization technique (MF) we use factorization machine (LibFM) for user and item feature included collaborative filtering. As seen from the tables, matrix factorization (MF) fails due to the low number of ratings by a user, but LibFM can already take advantage of the website metadata with performance similar to GBT.

Linear kernel SVM

Our bag of words models use the top $30k$ stemmed terms. For TF, TF-IDF and BM25, we show results for linear kernel SVM as it outperforms the radial basis function and polynomial kernels. We use LibSVM [21] for classification, and the Weka implementation of SMOReg [80] for regression.

Similarity kernel methods

The similarity kernel described in Section 7.3.2 gives the best results both for classification and regression. For distance, we use L_2 for the C3 attributes as well as TF, TF-IDF and BM25. For the last three, we also use the Jensen–Shannon divergence (J–S) as we suggested in [5]. While the similarity kernel over the bi-cluster performs weak for classification, it is the most accurate single

Method	Credibility	Presentation	Knowledge	Intentions	Completeness	Avg
Gradient Boosted Tree (**GBT**)	0.6492	0.6558	0.6179	0.6368	0.7845	0.6688
Factorization Machine (**LibFM**)	0.6563	0.6744	0.6452	0.6481	0.7234	0.6695
Matrix Factorization (**MF**)	0.5687	0.5613	0.5966	0.5700	0.5854	0.5764
TF linear kernel	0.6484	0.6962	0.6239	0.6767	0.6205	0.6531
TF-IDF linear kernel	0.6571	0.7020	0.5935	0.6824	0.6128	0.6496
BM25 linear kernel (**Lin**)	0.7236	0.7480	0.6278	0.6987	0.6633	0.6923
Bi-cluster linear kernel	0.6402	0.7467	0.5796	0.6482	0.6382	0.6506
Bi-cluster Sim kernel	0.6744	0.7718	0.6379	0.6830	0.6560	0.6846
C3 attributes Sim kernel	0.6267	0.7706	0.6327	0.6408	0.6149	0.6571
TF J–S Sim kernel	0.6902	0.7404	0.6758	0.7047	0.6778	0.6978
TF L_2 Sim kernel	0.6335	0.6882	0.6200	0.6585	0.6300	0.6460
TF-IDF J–S Sim kernel	0.7006	0.7546	0.6552	0.7073	0.6791	0.6994
TF-IDF L_2 Sim kernel	0.6461	0.7152	0.6013	0.6902	0.6353	0.6576
BM25 J–S Sim kernel	0.6956	0.7473	0.6351	0.6529	0.6222	0.6706
BM25 L_2 Sim kernel	0.7268	0.7715	0.6741	0.7081	0.6898	0.7141
BM25 L_2 & J–S Sim kernel (**BM25**)	0.7313	0.7761	0.6926	0.7141	0.7003	0.7229
BM25 & C3 Sim kernel	0.7449	0.8029	0.7009	0.7148	0.6993	0.7326
BM25 & Bi-cluster & C3 (**All**) Sim kernel	0.7457	0.8086	0.7063	0.7158	0.7052	0.7363
Lin + GBT	0.7296	0.8056	0.6589	0.6783	0.6939	0.7133
Lin + LibFM	0.7400	0.7769	0.6622	0.6733	0.6975	0.7100
All Sim kernel + Lin + GBT	0.7549	0.8179	0.6916	0.7098	0.7123	0.7373

Table 7.9: Detailed performance over the C3 labels in terms of AUC

Method		Credibility	Presentation	Knowledge	Intentions	Completeness	Avg
Gradient Boosted Tree (GBT)	MAE	1.5146	1.3067	1.2250	1.2737	1.4438	1.3528
	RMSE	1.6483	1.4510	1.3658	1.4132	1.6021	1.4961
Factorization Machine (LibFM)	MAE	1.5313	1.3213	1.2303	1.2632	1.4984	1.3689
	RMSE	1.6725	1.4745	1.3744	1.4073	1.6759	1.5209
Matrix Factorization (MF)	MAE	1.7450	1.4093	1.3676	1.2905	1.5794	1.4784
	RMSE	1.9174	1.5912	1.5540	1.4636	1.7583	1.6569
BM25 linear kernel (Lin)	MAE	0.5562	0.7230	0.6052	0.5979	0.5896	0.6144
	RMSE	0.7085	0.9072	0.7784	0.7910	0.7724	0.7915
BM25 L_2 Sim kernel	MAE	0.5678	0.7083	0.6228	0.5946	0.6045	0.6196
	RMSE	0.7321	0.9307	0.8038	0.7878	0.7930	0.8095
Bi-cluster Sim kernel	MAE	0.5340	0.6868	0.6039	0.5883	0.5813	0.5989
	RMSE	0.6958	0.8906	0.7861	0.7778	0.7624	0.7825
BM25 & Bi-cluster & C3 All Sim kernel	MAE	0.5403	0.6324	0.5946	0.5952	0.5829	0.5891
	RMSE	0.7106	0.8357	0.7763	0.7879	0.7661	0.7753

Table 7.10: Detailed performance over the C3 labels in terms of RMSE and MAE

method for regression.

In the similarity kernel, we may combine multiple distance measures by Equation (7.6). The All Sim method fuses four representations: J–S and L_2 over BM25 and L_2 for C3 and the bi-cluster representation. By the linearity of the similarity kernel, we may use LibSVM [21] for classification and SMOReg [80] for regression.

Classifier ensembles

Without using the similarity kernel, the best method is the average of the linear kernel over BM25 (Lin) and GBT. The performance is similar to the BM25 L_2 similarity kernel. As a remarkable feature of the similarity kernel, we may combine multiple distance functions in a single kernel. The best method (All Sim) outperforms the best combination not using the similarity kernel (Lin + GBT) by 3.2%. The difference is 7.2% for classifying "knowledge". The same method performs best for regression too.

7.5 Conclusions

Over the 190,000 host DC2010 data sets, we gave methods to classify Web hosts for neutrality, bias, trust and overall quality. Over the C3 data sets, we gave a large variety of methods to predict quality aspects of Web pages, including collaborative filtering and methods that use evaluator and page meta data as well as the content of the page. By our improved text classification method we described perhaps the first attempt of a practically useful quality classification over the DC2010 data set with AUC stable above 0.6 and we also achieved very promising results on the C3 data set with AUC over 0.7 for all aspects. By bi-clustering hosts and the top 30,000 most frequent terms, we reduce the term space to groups of words and also represent hosts by their distances from cluster centers. On top of our new host representation, we use multiple kernel learning methods. Surprisingly, unlike for the more traditional tasks as spam or genre classification where our new method gives marginal improvement, if any, for the hard tasks of neutrality, bias and trust we obtain strong improvement over the baseline of existing host-level classification methods. This fact indicates that especially neutrality and trust behaves very different from the well-known tasks of spam and genre classification.

We consider our results as first step with several technologies remaining open to be explored. For example, unlike expected, the ECML/PKDD Discovery Challenge 2010 participants did not deploy natural language processing based features.

7.6 My contribution

The experiments on the DC2010 is a joint work with Bálint Daróczy and András Benczúr and were published in [5]. The latter results were recently accepted at the 5th International Workshop on Web Quality and will be published by ACM within the WWW companion volume [3]. My main contribution is the design and implementation of the bi-clustering algorithm that can be used for text classification. The code is available on github[4]. In addition, I crawled and parsed the pages of the C3 data set, constructed the textual features and evaluated all the results. Róbert Pálovics ran the matrix factorization and gradient boosted tree baseline methods on the C3 data set. Bálint Daróczy experimented with the multiple kernel methods.

[4] `https://github.com/siklosid/co-cluster.git`

Own references

[1] István Bíró, Dávid Siklósi, Jácint Szabó, and András A. Benczúr. Linked latent dirichlet allocation in web spam filtering. In *AIRWeb '09: Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM Press, 2009.

[2] Károly Csalogány, A.A. Benczúr, D. Siklósi, and L. Lukács. Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn. In *Graph Labeling Workshop in conjunction with ECML/PKDD 2007*, 2007.

[3] Balint Daroczy, David Siklois, Robert Palovics, and Andras A Benczur. Text classification kernels for quality prediction over the c3 data set. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 1441–1446. International World Wide Web Conferences Steering Committee, 2015.

[4] András Garzó, Bálint Daróczy, Tamás Kiss, Dávid Siklósi, and András A Benczúr. Cross-lingual web spam classification. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1149–1156. International World Wide Web Conferences Steering Committee, 2013.

[5] D. Siklósi, B. Daróczy, and A.A. Benczúr. Content-based trust and bias classification via biclustering. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, pages 41–47. ACM, 2012.

Bibliography

[1] Jacob Abernethy, Olivier Chapelle, and Carlos Castillo. WITCH: A New Approach to Web Spam Detection. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[2] Einat Amitay, David Carmel, Adam Darlow, Ronny Lempel, and Aya Soffer. The Connectivity Sonar: Detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HT)*, pages 38–47, Nottingham, United Kingdom, 2003.

[3] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 485–492, 2006.

[4] Ludovic Denoyer Artem Sokolov, Tanguy Urvoy and Olivier Ricard. Madspam consortium at the ecml/pkdd discovery challenge 2010. In *Proceedings of the ECML/PKDD 2010 Discovery Challenge*, 2010.

[5] Wai-Ho Au, Keith C. C. Chan, and Xin Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Trans. Evolutionary Computation*, 7(6):532–545, 2003.

[6] Luca Becchetti, Carlos Castillo, Debora Donato, Stefano Leonardi, and Ricardo Baeza-Yates. Link-based characterization and detection of web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2006.

[7] N. Bel, C. Koster, and M. Villegas. Cross-lingual text categorization. *Research and Advanced Technology for Digital Libraries*, pages 126–139, 2003.

[8] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[9] András A. Benczúr, Károly Csalogány, and Tamás Sarlós. Link-based similarity search to fight web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), held in conjunction with SIGIR2006*, 2006.

[10] András A. Benczúr, Carlos Castillo, Miklós Erdélyi, Zoltán Gyöngyi, Julien Masanès, and Michael Matthews. ECML/PKDD 2010 Discovery Challenge Data Set. Crawled by the European Archive Foundation.

[11] James Bennett and Stan Lanning. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[12] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(5):993–1022, 2003.

[13] Ilaria Bordino, Paolo Boldi, Debora Donato, Massimo Santini, and Sebastiano Vigna. Temporal evolution of the uk web. In *Workshop on Analysis of Dynamic Networks (ICDM-ADN'08)*, 2008.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. In *Proceedings of the 9th World Wide Web Conference (WWW)*, pages 309–320. North-Holland Publishing Co., 2000.

[16] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430, 2007.

[17] Carlos Castillo, Kumar Chellapilla, and Ludovic Denoyer. Web spam challenge 2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[18] Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini, and Sebastiano Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.

[19] S. Chakrabarti, M. Van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640, 1999.

[20] Soumen Chakrabarti, Byron E. Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

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

[22] O. Chapelle, Y. Chang, and T-Y Liu. The yahoo! learning to rank challenge, 2010.

[23] N.V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[24] G.V. Cormack. Content-based Web Spam Detection. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2007.

[25] G.V. Cormack, M.D. Smucker, and C.L.A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval*, 14(5):441–465, 2011.

[26] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

[27] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[28] B. Daróczy, A. Benczúr, and R. Pethes. Sztaki at imageclef 2011. *Working Notes of CLEF 2011*, 2011.

[29] K. Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

[30] I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[31] G. Di Nunzio, N. Ferro, T. Mandl, and C. Peters. Clef 2007: Ad hoc track overview. *Advances in Multilingual and Multimodal Information Retrieval*, pages 13–32, 2008.

[32] Isabel Drost and Tobias Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, volume 3720 of *Lecture Notes in Artificial Intelligence*, pages 233–243, Porto, Portugal, 2005.

[33] S.T. Dumais, T.A. Letsche, M.L. Littman, and T.K. Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI spring symposium on cross-language text and speech retrieval*, volume 15, page 21, 1997.

[34] Miklós Erdélyi, András Garzó, and András A. Benczúr. Web spam classification: a few features worth more. In *Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality 2011) In conjunction with the 20th International World Wide Web Conference in Hyderabad, India.* ACM Press, 2011.

[35] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[36] D. Fetterly and Z. Gyöngyi. Fifth international workshop on adversarial information retrieval on the web (AIRWeb 2009). 2009.

[37] Denis Fetterly, Mark Manasse, and Marc Najork. Spam, damn spam, and statistics – Using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB)*, pages 1–6, Paris, France, 2004.

[38] Dániel Fogaras. Where to start browsing the web? In *Proceedings of the 3rd International Workshop on Innovative Internet Community Systems (I2CS)*, volume 2877/2003 of *Lecture Notes in Computer Science (LNCS)*, pages 65–79, Leipzig, Germany, June 2003. Springer-Verlag.

[39] Dániel Fogaras and Balázs Rácz. Scaling link-based similarity search. In *Proceedings of the 14th World Wide Web Conference (WWW)*, pages 641–650, Chiba, Japan, 2005.

[40] James Fogarty, Ryan S. Baker, and Scott E. Hudson. Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. In *Proceedings of Graphics Interface 2005*, GI '05, pages 129–136, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[41] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[42] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of statistics*, pages 337–374, 2000.

[43] Guanggang Geng, Xiaobo Jin, and Chunheng Wang. CASIA at WSC2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[44] J. Göbölös-Szabó, N. Prytkova, M. Spaniol, and G. Weikum. Cross-lingual data quality for knowledge base acceleration across Wikipedia editions. In *Proc. QDB*, 2012.

[45] D. Gomes, A. Nogueira, J. Miranda, and M. Costa. Introducing the portuguese web archive initiative. 2009.

[46] Daniel Gomes, Jo?o Miranda, and Miguel Costa. A survey on web archiving initiatives. In Stefan Gradmann, Francesca Borri, Carlo Meghini, and Heiko Schuldt, editors, *Research and Advanced Technology for Digital Libraries*, volume 6966 of *Lecture Notes in Computer Science*, pages 408–420. Springer Berlin Heidelberg, 2011.

[47] Xin-Chang Zhang Guang-Gang Geng, Xiao-Bo Jin and Dexian Zhang. Evaluating web content quality via multi-scale features. In *Proceedings of the ECML/PKDD 2010 Discovery Challenge*, 2010.

[48] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 403–412, 2004.

[49] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Web content categorization using link information. Technical report, Technical report, Stanford University, 2006.

[50] Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005.

[51] Zoltán Gyöngyi and Hector Garcia-Molina. Spam: It's not just for inboxes anymore. *IEEE Computer Magazine*, 38(10):28–34, October 2005.

[52] Zoltán Gyöngyi and Hector Garcia-Molina. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, 2005.

[53] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 576–587, Toronto, Canada, 2004.

[54] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 576–587. Morgan Kaufmann, 2004.

[55] M. Hammami, Y. Chahir, and L. Chen. Webguard: A web filtering engine combining textual, structural, and visual content-based analysis. *Knowledge and Data Engineering, IEEE Transactions on*, 18(2):272–284, 2006.

[56] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001.

[57] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[58] Glen Jeh and Jennifer Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 538–543, 2002.

[59] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th World Wide Web Conference (WWW)*, pages 271–279. ACM Press, 2003.

[60] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[61] Zhenzhen Kou and William W Cohen. Stacked graphical models for efficient inference in markov random fields. In *SDM 07*, 2007.

[62] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.

[63] X. Ling, G.R. Xue, W. Dai, Y. Jiang, Q. Yang, and Y. Yu. Can chinese web pages be classified with english data source? In *Proceedings of the 17th international conference on World Wide Web*, pages 969–978. ACM, 2008.

[64] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[65] Wangzhong Lu, Jeannette Janssen, Evangelos Milios, and Nathalie Japkowicz. Node similarity in networked information spaces. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative research*, page 11, 2001.

[66] Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 1999.

[67] Panagiotis T. Metaxas and Joseph Destefano. Web spam, propaganda and trust. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[68] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[69] X. Ni, J.T. Sun, J. Hu, and Z. Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 375–384. ACM, 2011.

[70] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.

[71] Vladimir Nikulin. Web-mining with wilcoxon-based feature selection, ensembling and multiple binary classifiers. In *Proceedings of the ECML/PKDD 2010 Discovery Challenge*, 2010.

[72] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of*

the 15th International World Wide Web Conference (WWW), pages 83–92, Edinburgh, Scotland, 2006.

[73] J. Olive, C. Christianson, and J. McCary. *Handbook of natural language processing and machine translation: DARPA global autonomous language exploitation*. Springer, 2011.

[74] Alexandra Olteanu, Stanislav Peshterliev, Xin Liu, and Karl Aberer. Web credibility: Features exploration and credibility prediction. In *Advances in Information Retrieval*, pages 557–568. Springer, 2013.

[75] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, 1998.

[76] Róbert Pálovics, Frederick Ayala-Gómez, Balázs Csikota, Bálint Daróczy, Levente Kocsis, Dominic Spadacene, and András A Benczúr. Recsys challenge 2014: an ensemble of binary classifiers and matrix factorization. In *Proceedings of the 2014 Recommender Systems Challenge*, page 13. ACM, 2014.

[77] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[78] Thanasis G Papaioannou, Jean-Eudes Ranvier, Alexandra Olteanu, and Karl Aberer. A decentralized recommender system for effective web credibility assessment. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 704–713. ACM, 2012.

[79] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *Computer Vision–ECCV 2006*, pages 464–475, 2006.

[80] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[81] PR10.info. BadRank as the opposite of PageRank, 2004. `http://en.pr10.info/pagerank0-badrank/` (visited June 27th, 2005).

[82] P. Prettenhofer and B. Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual*

Meeting of the Association for Computational Linguistics, pages 1118–1127. Association for Computational Linguistics, 2010.

[83] Xiaoguang Qi and Brian D. Davison. Knowing a web page by the company it keeps. In *Proceedings of the 15th Conference on Information and Knowledge Management (CIKM)*, 2006.

[84] Ross J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[85] Alain Rakotomamonjy, Francis Bach, Stephane Canu, and Yves Grandvalet. simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[86] G. Ramírez-de-la Rosa, M. Montes-y Gómez, L. Villasenor-Pineda, D. Pinto-Avendano, and T. Solorio. Using information from the target language to improve crosslingual text classification. *Advances in Natural Language Processing*, pages 305–313, 2010.

[87] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[88] L. Rigutini, M. Maggini, and B. Liu. An em based training algorithm for cross-language text categorization. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 529–535. IEEE, 2005.

[89] Tamás Sarlós, András A. Benczúr, Károly Csalogány, Dániel Fogaras, and Balázs Rácz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 297–306, 2006.

[90] Bernard Schölkopf. The kernel trick for distances. *MIT Press*, pages 301–307, 2000.

[91] Julia Schwarz and Meredith Morris. Augmenting web pages and search results to support credibility assessment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1245–1254. ACM, 2011.

[92] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[93] L. Shi, R. Mihalcea, and M. Tian. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1057–1067. Association for Computational Linguistics, 2010.

[94] P. Sorg and P. Cimiano. Enriching the crosslingual link structure of wikipedia-a classification-based approach. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artifical Intelligence*, pages 49–54, 2008.

[95] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8. ACM, 2008.

[96] Y. Tang, Y. He, S. Krasser, and P. Judge. Web Spam Challenge 2007 Track II Secure Computing Corporation Research. In *Graph Labeling Workshop in conjunction with ECML/PKDD 2007*, 2007.

[97] William J Teahan and David J Harper. Using compression-based language models for text categorization. In *Language modeling for information retrieval*, pages 141–165. Springer, 2003.

[98] X. Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics, 2009.

[99] H. Wang, H. Huang, F. Nie, and C. Ding. Cross-language web page classification via dual knowledge transfer using nonnegative matrix trifactorization. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 933–942. ACM, 2011.

[100] Chih-Ping Wei and I-Tang Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Syst. Appl.*, 23(2):103–112, 2002.

[101] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[102] Baoning Wu, Vinay Goel, and Brian D. Davison. Propagating trust and distrust to demote web spam. In *Workshop on Models of Trust for the Web*, Edinburgh, Scotland, 2006.

[103] Baoning Wu, Vinay Goel, and Brian D. Davison. Topical TrustRank: Using topicality to combat web spam. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, Edinburgh, Scotland, 2006.

[104] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[105] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.

[106] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Summary

In this thesis we introduced different aspects of Web quality along three main dimensions such as spam, genre and trustworthiness. We discussed the importance of these quality categories in search engines and Web archives, and we described novel methods for assessing Web sites over these categories that scale to the size of the Web.

In chapters 4 and 5 we showed the power of graph stacking methods in combination with graph similarity measures for spam, topical and telephone user churn classification. For future work we can test more complex multi-step variants of cocitation and the Jaccard coefficient. Jeh and Widom [58] define SimRank as a multi-step generalization of downweighted cocitation, however computing the full SimRank matrix requires quadratic space, therefor we may use the algorithm of [89] instead which applies fingerprinting techniques for optimization.

In chapter 6 we discussed the difficulties of cross-lingual Web classification and we introduced a new semi-supervised technique which takes advantage of multi-lingual Web sites. We showed that our method outperforms dictionary based methods and also combines very well with them. For future work it would be interesting to validate our method on other languages, however it is very hard to acquire good quality labeled data set for these experiments.

In chapter 7 we gave a new method for textual classification. The main idea is to use bag of concepts representation instead of bag of words representation via bi-clustering. With our method we obtained strong improvement for the quality categories introduced at the Discovery Challenge 2010 over the baseline of existing host-level classification methods. For future work it would be interesting to investigate how much improvement we could achieve by adding other text classification methods to our multiple kernel framework, like compression-based methods [97] or Latent Dirichlet Allocation (LDA) [12]. In our previous work [1] we already showed that we can obtain a slight improvement with LDA for spam classification and similarly to bi-clustering LDA gives a bag of topic representation of documents.

Összefoglalás

Disszertációmban Web-oldalak minőségének a különböző aspektusaival foglalkoztam úgy mint spam, témakör és megbízhatóság. Megmutattam, hogy a webes keresőmotorok és Web archívumok szempontjából miért fontosak ezek a minőségi kategóriák és új, a Web méretéhez jól skálázódó módszereket adtam ezen kategóriák automatikus klasszifikálására.

A 4. és 5. fejezetben láthattuk, hogy a graph stacking módszerét különböző gráf-hasonlósági mértékekkel kombinálva nagyon jó eredményeket érhetünk el spam, témakör és churn felismerésében. A jövőben érdemes lehet kipróbálni komplexebb több-lépéses változatait a Jaccard és Kocitáció hasonlósági mértékeknek. Jeh és Widom az [58] munkájukban definiálják a SimRank-et, ami a lesúlyozott Kocitációnak egy több-lépéses változata. Azonban a teljes SimRank kiszámításának négyzetes a tárigénye, ezért érdemes az algoritmusnak az [89] munkában bemutatott fingerprint-ekkel optimalizált változatát használni.

A 6. fejezetben megismerkedhettünk a keresztnyelvű klasszifikáció nehézségeivel és mutattunk egy újszerű félig-felügyelt tanuló módszert a probléma megoldására, ami a többnyelvű Web hosztokban rejlő lehetőségeket használja ki. Megmutattuk, hogy módszerünkkel jobb eredményt lehet elérni, mint a szótárat használó módszerekkel, ugyanakkor jól is kombinálható ezekkel. Érdekes lenne a módszert további nyelveken is validálni, azonban nagyon nehéz jó minőségű felcímkézett adathalmazt beszerezni.

A 7. fejezetben egy új típusú szövegen alapuló klasszifikációs módszert mutattunk be. Az alapötlet az, hogy bag of words reprezentáció helyett generáljunk bag of concepts reprezentációt a dokumentumainkhoz bi-klaszterezés segítségével. Módszerünkkel lényeges javulást értünk el az alapmódszerekhez képest a Discovery Challange 2010 versenyen bemutatott 3 új minőségi kategóriára. Érdemes lehet kipróbálni, hogy további módszerekkel kombináljuk a bi-klaszteres megoldásunkat a fejezetben bemutatott multi-kerneles módszer segítségével. Ilyen módszerek lehetnek a tömörítésre alapuló szöveg klasszifikációs megoldások [97] vagy például a Latent Dirichlet Allocation (LDA) [12]. Egy előző munkánkban [1] már láttuk, hogy az LDA segítségéve lehet némi javulást elérni spam klasszifikálásban. MIvel az LDA a bi-klaszterezéshez hasonlóan bag of topics reprezentációt állít elő a dokumentumokhoz, ezért érdekes lehet, hogy milyen eredmények ad az új minőségi kategóriákra.

[1]**ADATLAP**
a doktori értekezés nyilvánosságra hozatalához

I. A doktori értekezés adatai
A szerző neve: Siklósi Dávid
MTMT-azonosító: 10017773
A doktori értekezés címe és alcíme: Assessing the quality of Web via semi-supervised methods
DOI-azonosító[2]: 10.15476/ELTE.2016.007
A doktori iskola neve: ELTE Informatika Doktori Iskola
A doktori iskolán belüli doktori program neve: Az informatika alapjai és módszertana
A témavezető neve és tudományos fokozata: Benczúr András ifj. Ph.D.
A témavezető munkahelye: MTA SZTAKI

II. Nyilatkozatok
1. A doktori értekezés szerzőjeként[3]

a) hozzájárulok, hogy a doktori fokozat megszerzését követően a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az ELTE Digitális Intézményi Tudástárban. Felhatalmazom a ELTE Informatika Doktori Iskola hivatalának ügyintézőjét Ríz-Herczeg Krisztinát, hogy az értekezést és a téziseket feltöltse az ELTE Digitális Intézményi Tudástárba, és ennek során kitöltse a feltöltéshez szükséges nyilatkozatokat.

b) kérem, hogy a mellékelt kérelemben részletezett szabadalmi, illetőleg oltalmi bejelentés közzétételéig a doktori értekezést ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[4]

c) kérem, hogy a nemzetbiztonsági okból minősített adatot tartalmazó doktori értekezést a minősítés (*dátum*)-ig tartó időtartama alatt ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[5]

d) kérem, hogy a mű kiadására vonatkozó mellékelt kiadó szerződésre tekintettel a doktori értekezést a könyv megjelenéséig ne bocsássák nyilvánosságra az Egyetemi Könyvtárban, és az ELTE Digitális Intézményi Tudástárban csak a könyv bibliográfiai adatait tegyék közzé. Ha a könyv a fokozatszerzést követőn egy évig nem jelenik meg, hozzájárulok, hogy a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban.[6]

2. A doktori értekezés szerzőjeként kijelentem, hogy

a) az ELTE Digitális Intézményi Tudástárba feltöltendő doktori értekezés és a tézisek saját eredeti, önálló szellemi munkám és legjobb tudomásom szerint nem sértem vele senki szerzői jogait;

b) a doktori értekezés és a tézisek nyomtatott változatai és az elektronikus adathordozón benyújtott tartalmak (szöveg és ábrák) mindenben megegyeznek.

3. A doktori értekezés szerzőjeként hozzájárulok a doktori értekezés és a tézisek szövegének plágiumkereső adatbázisba helyezéséhez és plágiumellenőrző vizsgálatok lefuttatásához.

Kelt: Budapest, 2016. Február 01.

a doktori értekezés szerzőjének aláírása

Siklósi Dávid

---

[1] Beiktatta az Egyetemi Doktori Szabályzat módosításáról szóló CXXXIX/2014. (VI. 30.) Szen. sz. határozat. Hatályos: 2014. VII.1. napjától.

[2] A kari hivatal ügyintézője tölti ki.

[3] A megfelelő szöveg aláhúzandó.

[4] A doktori értekezés benyújtásával egyidejűleg be kell adni a tudományági doktori tanácshoz a szabadalmi, illetőleg oltalmi bejelentést tanúsító okiratot és a nyilvánosságra hozatal elhalasztása iránti kérelmet.

[5] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a minősített adatra vonatkozó közokiratot.

[6] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a mű kiadásáról szóló kiadói szerződést.