Techniques for the Removal of Non-informative Links

André Manuel Arsénio de Brito
Instituto Superior Técnico,
Av. Prof. Cavaco Silva,
2780-990 Porto Salvo,
Portugal
andre.brito@tagus.ist.utl.pt
September 2008

Abstract

The existence of spam in the web drastically reduces its usability and the user’s experience. Furthermore, it has a negative impact in search engines’ performances, which tend to present irrelevant results as a consequence of the usage of the spam techniques used nowadays. In order to solve this problem, it is necessary to be able to distinguish informative links and non-informative links. Unlike prior work in this area, our work targets the detection of non-informative links and not just spam. Also, the fact that we use a clustering method greatly increases our method’s usability when compared to others.

In this work we propose three methods for automatic non-informative link detection, based on the analysis of statistical features of the links between web pages: feature selection, where feature selection methods for text categorization were adapted for web page categorization; classification, using a linear combination of features or a SVM classifier; and clustering (CLUTO and \(k\)-means), where links are separated in two classes — informative and non-informative.

The experiments with different sized collections revealed that all the proposed methods, except the feature selection method, proved effective in detecting non-informative links. Among all algorithms, \(k\)-means has the greatest advantage, because, not only does it support large collections, but also it barely requires human intervention to operate.

Keywords: Web, Spam, Non-Informative Links, Detection, Features, Clustering

1 Introduction

The World Wide Web (or, in short, web) is composed by a set of pages and hyperlinks (links) that connect those pages. A link between two pages means, among other things, that the author of the link’s source page believes that the link’s destination page is relevant. In other words, a link represents a trust vote from the source page’s author on the quality of the destination page. There are several algorithms that use links as a source of information for search and classification purposes [14,16,20] and they are used mainly by search engines, whose task is to find relevant pages on a given search topic.

There are increasingly more ways of artificially boosting the probability of a web page showing up in the first results of web searches. These techniques, known as Search Engine Optimization (SEO), comprise methods such as term spamming and link spamming. The act of creating web pages with the sole purpose of misleading search engines into ranking those pages higher than they deserve, without improving their quality, is designated by web spam, or simply, spam. Current spam techniques can be divided in two groups: boosting techniques and hiding techniques [9]. Boosting techniques are aimed at increasing a page’s relevance and/or importance, by changing their content. Term spamming, for example, is the insertion of terms that are unrelated to the page’s content in order to make the page relevant to some common search queries. Another example is link spamming, where the link structure is manipulated by creating links between web pages, in order to deceive search engines into ranking those pages higher than they deserve. Unlike boosting techniques, hiding techniques don’t increase the page’s relevance — they simply hide the boosting techniques used. For example, by coloring the text created by term spamming with the page’s background color, the text becomes invisible to a human.

The motivation for this work is that the existence of spam drastically reduces the usability of the web and the user’s experience. Furthermore, it has a negative impact in search engines’ performances, which tend to present irrelevant results as a consequence of the usage of the spam techniques mentioned above. In order to solve this problem,
it is necessary to be able to distinguish informative links and non-informative links. The informativeness of a link is related to the utility of that link to the user’s necessities of information. In [5], Davison defines nepotic links as links between web pages that are present for reasons other than merit, or, in other words, links that represent an illegit trust vote by the author of the link’s source page on the quality of the link’s destination page. This definition covers not only spam links, but, for example, navigational links as well. Conversely, an informative link is a link between two web pages where it is legit to considered that the link’s destination page is more relevant due to the existence of the said link.

In this work, we propose to automatically identify informative links and non-informative links. The hypothesis we want to test is: by evaluating specific statistic features of links between web pages, is it possible to automatically determine whether the links are informative or non-informative?

The remainder of this paper is structured as follows. Section 2 describes the related work. Section 3 presents the proposed algorithms for non-informative link removal. Then, in Section 4, we evaluate the proposed algorithms and analyze their performance. Finally, Section 5 draws some conclusions and discusses future work.

2 Related Work

There are several known techniques for detecting web spam and they can be divided in two categories: detection based on the content and detection based on the link structure. This section describes the most relevant works on each of these areas.

2.1 Content-Based Spam Detection

Content-based methods analyze the web page’s content solely, disregarding any information related to the link structure. Most of these methods use classifiers, that are based on text classifiers and adapted to web page classification. Toulas et al. [19] present a set of heuristics that can be used to identify web spam by analyzing the web page’s content, e.g., the number of words in the page, the average length of words, the page’s compressibility, etc. Used alone, none of these heuristics can detect spam effectively. However, their combination shows good accuracy: 95.4% of the pages in a 100 million pages sample were correctly classified.

The work of Fetterly et al. [7] addresses content-based spam detection by using a set of statistical features to identify automatically generated web pages. The base idea is that automatically generated web pages can be distinguished from other legit pages by analyzing some features such as the URL, the page content and it’s evolution over time, for example.

Another work by Fetterly et al. [8] addresses phrase-level duplication. This method detects websites with many automatically generated spam pages and estimates how original the content of a web page or site is. The method uses a clustering algorithm to group pages that are exact or near duplicates of each other.

Finally, the Hidden Style Similarity (HSS) [23] is a similarity measure based on extra-textual features of web pages. The value of the HSS measure is calculated by removing all alphanumeric characters, keeping only the “noisy” parts of the HTML document. This method uses a clustering algorithm to group pages based on their template and on their writing style. It is then possible to identify pages of a certain type, e.g., forum pages or automatically generated pages.

2.2 Link Structure-Based Spam Detection

Some forms of spam can only be detected by analyzing the link structure that connects web pages. For instance, link farms are complex structures of links between web pages created for the sole purpose of boosting the rank of one of those pages. Becchetti et al. [1] define a set of features for spam detection based on link structure analysis including the page’s in-links, the page’s out-links, and link reciprocity, for example. The combination of all the defined features is effective at detecting spam.

Davison [5] defined a set of seventy five features to recognize nepotic links, which are not only spam links, but also any kind of undesired and irrelevant links. For example, a website with persistent navigational links on an all its n pages is adding n irrelevant in-links to each of those pages, resulting in an undesired increase of their ranks. These features defined by Davison analyze the source and destination pages of a link and compare their domains, host-names, URL’s, and IP addresses, among other things. This method shows good results at detecting nepotistic links on a small sample used for testing.

A supporter is a page that contributes directly (through a link) to another page’s rank. SpamRank [2] is an algorithm that uses the definition of supporters to detect web spam. SpamRank measures the amount of undeserved PageRank [20] score received by a web page. The base idea of this method is that supporters of an honest page should not be overly dependent on one another, i.e. they should be spread across sources of different quality.
For example, a suspicious page would be one that receives its PageRank score from many low ranked pages.

All the methods described above analyze the web solely at the page level, i.e. they analyze the links between web pages as individual and independent units. However, there are certain link structures that can only be detected by a site level analysis and not page by page. There are three different approaches for this problem presented by da Costa et al. [4]: identifying mutual reinforcement relations between websites (Mutual Site Reinforcement); identifying relationships between sites where one site has PageRank scores accumulated mostly from only one different site (Site Level Abnormal Support); and penalizing alliances of sites that artificially promote some target site (Site Level Link Alliances). These methods do not replace page-level analysis and should be used together for accurate results.

3 Non-Informative Link Detection

In this section, we describe the three methods we propose for the detection of non-informative links: features selection, classification and clustering.

3.1 Feature Selection

The feature selection method detects non-informative links by adapting existing feature selection methods for text categorization. Feature selection methods focus on aggressive dimensionality reduction by determining which terms allow us to better distinguish text documents. On the web, we want to determine which links allow us to better distinguish web pages and those links are considered informative links, while the others are considered non-informative. To this end, we use three feature selection measures: Information Gain (IG), Mutual Information (MI) and $\chi^2$ [25]. As these measures are for term selection in text categorization, we adapted them to link selection in web page categorization by assuming that web pages are text documents and that the links between those pages are text terms.

IG is widely used in document classification as a measure of term goodness and is calculated as:

$$ig(t) = -\sum_{c \in C} P(c) \log P(c) + \frac{P(t) \sum_{c \in C} P(c|t) \log P(c|t)}{P(t)} + \frac{P(t) \sum_{c \in C} P(c|t) \log P(c|t)}{(P(t))^2}$$ (1)

where $t$ is a term, $c$ is a category and $C$ is the set of categories.

MI measures the co-occurrence relation between a term and a category and is calculated as:

$$mi(t,c) = \log \frac{P(t \land c)}{P(t)P(c)}$$ (2)

where $t$ is a term and $c$ is a category. In order to make this measure category independent, we combine the category specific scores of a term by calculating the average score ($mi_{avg}$) or the maximum score ($mi_{max}$).

Finally, $\chi^2$ statistically measures the lack of independence between a term and a category, and is calculated as:

$$\chi^2(t,c) = \frac{N[P(t|c)P(|c) - P(t|c)P(t|c)]^2}{P(t)P(t|c)P(c)P(c)}$$ (3)

where $t$ is a term, $c$ is a category and $N$ is the total number of documents. In order to make this measure category independent, we combine the category specific scores of a term by calculating the average score ($\chi^2_{avg}$) or the maximum score ($\chi^2_{max}$).

In order to simplify the calculation of these measures, we use contingency tables [21]. A contingency table stores information about the co-occurrence of two variables (a term and a category) and allows us to analyze how they relate to each other. The usage of 2x2 contingency tables would require one instance of this table for each term-category pair. This translates not only on high computational costs, but also on the creation of redundant information. To solve this problem, we use a generalized contingency table [6], which stores information about the relation between all terms and all categories. As the generalized contingency table refers to documents and terms, we need to adapt it to web pages and links. We consider that web pages are text documents and the links between those pages are text terms. Each cell of the generalized contingency table $f_{i,j}$ will account the number of links whose source web page is categorized as $c_i$ and whose destination web page is $p_j$. For example, if there is a link $l$ originating at a web page $p_1$ categorized as $c_{10}$ and linking it to a web page $p_2$ categorized as $c_{11}$, the existence of the link $l$ will be accounted in the cell $f_{2,10}$. Table 1 shows an example of a generalized contingency table adapted to web pages where each $c_i$ is a category and each $p_j$ is a web page. $f_{j,i}$ is the number of links whose source web page is categorized as $c_i$ and whose destination web page is $p_j$. $F_j$ is the total number of links whose destination web page is $p_j$, i.e., the number of $p_j$’s in-links, and is calculated as: $\sum_{i=1}^{n} f_{j,i}$. $S_i$ is the total number of links whose source page is categorized as $c_i$ and is calculated as: $\sum_{j=1}^{m} f_{i,j}$. $N$ is the total number of links between all web pages in all categories.

$$\text{Table 1: Example of a generalized contingency table}$$

<table>
<thead>
<tr>
<th>Category</th>
<th>Web Page 1</th>
<th>Web Page 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{10}$</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>$c_{11}$</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

3
### 3.2 Link Features

In order to detect non-informative links, we can analyze specific features of web pages and the links between them. To that end, we define a set of features that allows us to characterize web pages and their links, by analyzing information like the amount of in-links of a web page, the amount of words in a web page’s text, or the link’s position in the source page, among others. The objective of these features is to allow classification and clustering methods to detect non-informative links. The definition of the presented features was based on the works of Ntoulas et al. [19], Becchetti et al. [1] and Davison [5].

The defined features were divided in two major groups, based on the information they analyze: **page-only features** and **link-only features**. Notice that this division differs from the division between content-based spam detection and link structure-based spam detection, presented in the previous section. The features hereby presented as page-only features and link-only features analyze both the web page’s content and the link structure. The division is made based on the focus of that analysis: the web page (as an independent unit) or the link (as a relation between two web pages).

#### 3.2.1 Page-only Features

Page-only features characterize web pages as independent units and are described next:

- **Word Count**: counts the number of words in a web page’s text. The objective is to identify spam pages by analyzing the number of words in the page’s text, as it is expectable that spam pages have either too many (term spamming) or too few (link structure manipulation) words. The separator used to delimit words is the character space. Therefore, we define word as any sequence of characters without a space.
- **Title Word Count**: like the previous one, but for the page title.
- **Average Word Size**: calculates the average word size of a web page’s text. The objective is to identify pages with a high average word size, which are probably spam pages that use keyword composition in order to have the page shown in the results of misspelled queries.
- **Path Keywords**: assesses the presence of keywords in the web page’s URL that suggest the presence of spam, e.g., advertise, links, etc.
- **Slash Count**: counts the number of slashes (“/” character) in the web page’s URL. The objective is to measure the web page’s depth in the host website. The more slash characters a web page’s URL has, the deeper the web page is.
- **InLinks**: counts the amount of in-links of a web page, i.e., the number of links whose destination page is the web page being analyzed. The objective is to detect web pages with many in-links, which are probably spam pages that manipulate the link structure in order to obtain higher ranks.
- **OutLinks**: like the previous one, but for out-links (links whose source page is the web page being analyzed).

<table>
<thead>
<tr>
<th></th>
<th>c₁</th>
<th>c₂</th>
<th>...</th>
<th>cₙ</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>f₁,₁</td>
<td>f₁,₂</td>
<td>...</td>
<td>f₁,ₙ</td>
<td>F₁</td>
</tr>
<tr>
<td>P₂</td>
<td>f₂,₁</td>
<td>f₂,₂</td>
<td>...</td>
<td>f₂,ₙ</td>
<td>F₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Pₙ</td>
<td>fₙ,₁</td>
<td>fₙ,₂</td>
<td>...</td>
<td>fₙ,ₙ</td>
<td>Fₙ</td>
</tr>
<tr>
<td>S</td>
<td>S₁</td>
<td>S₂</td>
<td>...</td>
<td>Sₙ</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 1: Generalized contingency table for web pages.

This table allows us to calculate the values A, B, C and D of a 2x2 contingency table for a web page \(p\) and a category \(c\) as: \(A = f_{p,c}; B = F_p - A; C = S_p - A;\) and \(D = N - (A + B + C)\). Using these values, the IG, MI and \(\chi^2\) formulas can be simplified, reducing the computational cost of their calculation [6]. Thus, IG becomes:

\[
\sum_{d=1}^{i} \frac{S_d}{N} \log \left( \frac{S_d}{N} \right) + \frac{F_p}{N} \sum_{d=1}^{i} \frac{A}{A + B} \log \left( \frac{A}{A + B} \right) + (1 - F_p) \sum_{d=1}^{i} \frac{C}{C + D} \log \left( \frac{C}{C + D} \right)
\]

MI is computed as:

\[
\log \left( \frac{A \times N}{(A + C) \times (A + B)} \right)
\]

Finally, \(\chi^2\) is computed as:

\[
\frac{N \times [(A \times D) - (C \times B)]^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)}
\]
3.2.2 Link-only Features

Link-only features characterize the link between two specific web pages and are described next:

- **Host Name**: compares the host names of the link's source and destination web pages. *Host name* is defined as the URL part that follows the prefix `http://` and ends at the first `"/"` character found next. The objective is to determine whether the link is *internal* (connects two pages from the same website) or *external* (connects pages from different websites). An internal link is considered non-informative.

- **Top Domain**: checks if the top domains of the link's source and destination web pages are the same. We define *top domain* as the last two components of the URL's host name. Each component is separated by a dot (`"."`). For example, given the host name `www.ist.utl.pt`, the top domain is `utl.pt`. The objective is to determine whether the link is internal or external.

- **Domain Name**: checks if the domains of the link's source and destination web pages are the same. We define *domain* as the first component of the URL’s top domain (each component is separated by a dot). For example, given the top domain `utl.pt`, the domain is `utl`. Like the previous features, the objective is to determine whether the link is internal or external.

- **IP**: checks if the IP addresses of the link’s source and destination web pages are the same. The objective here is also to determine whether the link is internal or external, by detecting web pages hosted on the same device.

- **Link Count**: counts the number of times the same link appears on the web graph, i.e., how many times a link between the same two web pages and in the same direction exists. The objective is to detect link spamming situations where the same link appears multiple times with the intent of boosting the rank of the link's destination page.

- **Link Position**: determines the link's position on the text of the link’s source page. The objective is to detect pages whose links are mostly located on the top and bottom of the page. Those positions make the links more visible and accessible to humans, which means they are probably navigational or publicity links and, therefore, are non-informative links.

3.3 Classification

Using the features defined above, we can use classification methods to detect informative and non-informative links automatically. We propose two approaches: linear combination of features and SVM classification.

3.3.1 Linear Combination of Features

This method classifies the links between web pages as informative or non-informative according to the feature values, by linearly combining them. The first step of the algorithm is to define feature weights, followed by the calculation of the feature values according to the available information about the web pages and their links. Next, the calculated feature values are linearly combined with the pre-defined feature weights. The last step is to classify each link as informative or non-informative, based on the linear combination results and on a threshold value.

The linear combination (*LC*) of the feature values is calculated as follows:

\[
LC (l) = \sum_{i=1}^{d} a_i v_{i,l}
\]

where \( l \) is a link between two web pages, \( d \) is the total number of features, \( a_i \) is the individual weight of feature \( i \) and \( v_{i,l} \) is the value of feature \( i \) for link \( l \).

The predefined feature weights should guarantee that the larger the feature value is, the higher the probability will be that the link is non-informative and, the smaller the feature value is, the higher the probability will be that the link is informative. The idea behind is that, on the one hand, a link whose combined feature value is high is probably a non-informative link, while on the other hand, a link whose combined feature value is low is probably an informative link.

Concluding, a link is considered informative if the result of the linear combination of the features values is less than or equal to the threshold value. Conversely, a link is considered non-informative if the result of the linear combination of the features values is higher than the threshold value.

3.3.2 SVM Classifier

*Support Vector Machines* (SVMs) are a classification method proposed by Vapnik in 1995 [24]. Recently, they have been successfully used in document classification [11, 13]. The documents are represented on the vector space and the idea is to find the hyperplane in that space that better divides two categories. SVMs deal effectively with the obstacles of the vector space representation, namely high dimensionality and sparse vectors.

This method was adapted to link classification, by assuming that web pages are documents and that the links between those pages are terms. The features defined above are used to characterize links between web pages. Each link is represented by a vector with \( d \) dimensions (\( d \) is the total number of features) and the value of each of those dimensions
is the value of a feature. The links are represented in a vector space with \( d \) dimensions and the algorithm's objective is to find the hyperplane in that space that better separates two categories (informative links and non-informative links), with minimal error.

The implementation of this method uses SVM-Light [12], a Vapnik’s SVM implementation for the problem of pattern recognition, for the problem of regression, and for the problem of learning a ranking function. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently.

3.4 Clustering

Another way of detecting non-informative links is to use clustering algorithms to group links between web pages according to their characteristics (described by the feature values). This will, ideally, allow us to separate informative links from non-informative links. Each link is represented by a vector with \( d \) dimensions (\( d \) is the total number of features) and the value of each of those dimensions is the value of a feature. The links are represented in a vector space with \( d \) dimensions and the clustering algorithm groups them according to their features.

The objective of a clustering algorithm is to group physical or abstract objects in classes of similar objects. In our case, we want to group links between web pages into two classes: informative links and non-informative links. Clustering is an unsupervised learning method [17], as it does not require human intervention to function. However, our clustering solution requires the presence of a human, but only at the last phase, to decide which cluster contains the informative links and which cluster contains the non-informative links.

We used two implementations of clustering algorithms: CLUTO and \( k \)-means.

3.5 Repeated Bisections

CLUTO [26, 27] is a clustering package containing several clustering algorithms that supports multi-dimensional data sets, offers several choices for criterion functions and similarity functions, and produces cluster quality statistics. In our work, we used the \texttt{vacluster} program that treats each object as a vector in a high-dimensional space. The clustering algorithm used was repeated bisections, an algorithm that computes the desired \( k \)-way clustering solution by performing a sequence of \( k \)-1 repeated bisections, where \( k \) is the desired number of clusters. Cosine similarity [22] was used as the similarity function and the chosen criterion function was the \( L_2 \).

CLUTO reads all input data (vector matrix) to memory before applying the clustering algorithm. Although this method is suitable for small size collections, it becomes unbearable for large collections, as the memory requirements become too high.

3.6 \( k \)-Means

\( k \)-Means is a partitional clustering method that, given a parameter \( k \) (the desired number of clusters), partitions a data set with \( n \) objects into \( k \) clusters. The algorithm’s objective is to group the objects in clusters, while maximizing the similarity between the objects of the same cluster (intra-cluster similarity) and, at the same time, minimizing the similarity between the objects of different clusters (extra-cluster similarity). Similarity is measured in relation to the cluster’s center of gravity — the centroid [10].

As mentioned above, CLUTO has high memory requirements. This \( k \)-means implementation, developed specifically for this work, intends to solve that problem by reading all input data on every iteration, instead of storing it in memory. This solution increases the algorithm’s execution time, but it is the only way of clustering large collections.

This \( k \)-means implementation works as follows. First, the algorithm randomly selects \( k \) objects that will be the initial cluster centroids. The remaining objects are then assigned to their nearest cluster — the one whose centroid is closer to the object. The similarity measure used is the euclidean distance [10], calculated as:

\[
dist (x, c_j) = \sqrt{\sum_{i=1}^{d} (x_i - c_{j,i})^2}
\]

where \( x \) is an object, \( c_j \) is the centroid of cluster \( j \), \( d \) is the number of dimensions (features) of the vector space, \( x_i \) is the value of dimension (feature) \( i \) of object \( x \) and \( c_{j,i} \) is the value of dimension \( i \) of the centroid of cluster \( j \). This measure becomes 0 when the objects are identical and, the lesser the distance between two objects, the more similar they are.

After all objects are assigned to a cluster, the new centroid of each cluster must be calculated. This is accomplished by calculating the arithmetic mean of each dimension separately over all the objects in that cluster. The new coordinates of the a cluster centroid are calculated as:

\[
c_j = \frac{1}{|C_j|} \sum_{x \in C_j} x
\]

where \( C_j \) is the cluster \( j \), \( c_j \) is the centroid of cluster \( j \) and \( x \) is an object that was assigned to cluster \( j \).

This process continues until the variation of the criterion function of two consecutive iterations is less than a set limit or when a predefined maximum number of iterations is completed. The criterion
function used is the quadratic error \( E \) [10] and is calculated as:

\[
E = \sum_{j=1}^{k} \sum_{x \in C_{j}} \text{dist}(x, c_{j})^{2}
\]

(10)

where \( x \) is an object, \( C_{j} \) is the cluster \( j \), \( c_{j} \) is the centroid of cluster \( j \), \( \text{dist} \) is the euclidean distance and \( k \) is the total number of clusters.

When the algorithm converges, the result is the assignment of each object in the vector space to one of the clusters, maximizing intra-cluster similarity while minimizing extra-cluster similarity. Also, the algorithm produces cluster quality statistics for each cluster: internal similarity (average similarity between the objects of the cluster), external similarity (average similarity of the objects of the cluster and the objects of all other clusters), descriptive features (the features that contribute the most to define the cluster) and discriminating features (the features that contribute the most to distinguish the cluster from the other clusters).

4 Experiments

4.1 Experimental Setup

To test the proposed algorithms, we used three datasets: SS, WBR-99 and CADE. SS is a small size collection that contains 52 HTML documents in their original format and 5162 links originating at those documents. This collection was created specifically for this work during January of 2008. WBR-99 [3] is a large dataset containing about 6 million HTML documents in an already indexed format and about 40 million links between those documents, classified as internal links or external links. This collection was created from the database of TodoBR (a search engine for the Brazilian web) during November 1999. Lastly, CADE is based on the WBR-99 collection and contains about 40,000 of the documents in WBR-99 classified under 12 different categories.

In order to evaluate the proposed methods, we use several measures, such as: precision and recall [10], two basic measures widely used in Information Retrieval to evaluate an algorithm’s quality; accuracy, defined in SVMLight as the proportion of the total number of predictions that are correct [12]; cluster quality measures, namely, internal similarity (ISim), external similarity (ESim), descriptive features and discriminating features [26].

4.1.1 Tests

In order to test the feature selection method, we used the CADE collection, because it already has it’s documents categorized. Furthermore, we extended CADE with the objective of increasing it’s link density. The extended versions of CADE categorize the \( p \)-predecessors and the \( p \)-successors of each document in the original CADE under the same category as that document, where \( p \) is the desired depth level. We created two extended versions of CADE: the first with 2 depth levels (CADE2), containing a total of 142,909 documents; and the second with 6 depth levels (CADE6), containing a total of 143,130 documents. The procedure for the tests execution was: calculate IG, MI and \( \chi^{2} \) values for each document on each instance of the collection (original CADE, CADE2 and CADE6); combine the category independent scores of MI and \( \chi^{2} \) (maximun value); analyze the calculated scores for the feature selection measures.

To test the classification methods, we used the SS collection. As the SVM classifier requires a previously classified sample, all links in the collection were manually classified as informative or non-informative. The features used to test both classification methods were (feature weights for the linear combination method are described in parenthesis): Host Name (30); Domain Name (27); Top Domain (40); IP (40); Parent OutLinks (15), the out-links of the link’s source page; Path Keywords (45); Link Position (15); Link Count (20) and Slash Count (20). For the method of linear combination of features, the procedure for the tests execution was: calculate the feature values for each link in the SS collection; combine the feature values with the linear combination method; classify each link as informative or non-informative, according to the threshold value of 0.25. For the SVM classification method, we used a 10-fold cross-validation procedure [15] to run the tests. For this, the SS collection (already classified) was divided in ten subsets and we executed the program ten times. Each subset was used as testing set exactly once and was used as training set on the remaining executions. To combine the ten different values obtained, we calculated their average value.

The tests to the clustering methods used the SS and WBR-99 collections. Regarding the features used in the tests, we will refer to the link’s source web page as parent and the link’s destination web page as page. For example, the feature Parent OutLinks refers to the out-link count of the link’s source web page, while the feature Page OutLinks refers to the out-link count of the link’s destination web page. For the tests with the SS collection, we used the following features: Host Name, Domain Name, Top Domain, IP, Parent OutLinks, Path Keywords, Link Position, Link Count, and Slash Count. The tests with the WBR-99 collection used the following features: Host Name, Domain Name, Top Domain, Path Keywords, Slash Count, Parent Average Word Size, Page Average Word Size, Parent OutLinks, Page OutLinks, Par-
### 4.2 Results

The tests to the feature selection method with the original CADE collection revealed a problem with the feature selection formulas that make it impossible to obtain conclusive results. Given a web page $p_j$ that has no in-links, there will be several divisions by zero in the formulas, making it impossible to calculate the feature selection measures for that page. This situation happens multiple times in the original CADE collection and is the reason for the creation of the extended versions of the collection that attempt to increase link density. In Table 2, we can see the percentage of web pages in the original CADE, CADE2 and CADE6 for which we could calculate the IG, MI and $\chi^2$ values. As evidenced in the table, the IG measure could only be calculated for a small number of documents and, therefore, should be discarded. By looking at the pages with highest and lowest scores, we can conclude that these feature selection measures are not able to properly distinguish informative links from non-informative links, as both top and internal web pages of a site have either high or low values.

To determine the correctness of the results produced by the tests to the linear combination of features method, we compared them with the manual classification of the SS collection. The results allowed us to calculate the precision and recall values for informative links — 86% and 88%, respectively — and for for non-informative links — 98% and 98%, respectively. The accuracy was 97%. Table 3 describes the results of the tests to the SVM classifier method. The table presents the average values of accuracy, precision and recall over the 10 executions of the program for informative (I) and non-informative (N-I) links.

The results of the tests to the clustering methods are summarized on Table 4. We can see the values of ISim and ESim as well as the most descriptive (Desc. F.) and discriminating (Disc. F.) features for the cluster containing the informative links (IC) and for the cluster containing the non-informative links (N-IC). Also, we calculated the values of precision, recall and accuracy for both CLUTO and k-Means algorithms on the tests with the SS and WBR-99 collections.

### 4.3 Discussion

The results of the tests to the feature selection method lead us to conclude that the method is not suitable for non-informative link detection. For example, we see internal links of websites getting both high and low scores.

About the classification methods, both the method of linear combination of features and the SVM classifier showed good results. We can see high values of precision, recall and accuracy for both methods, proving they are suitable for non-informative link detection. However, the tests also revealed that the method of linear combination of features requires a huge human effort to function. The tuning of its parameters — feature weights and threshold value — is not a straightforward task and it needs to be done for every new collection. Regarding the SVM classifier, the fact that it requires a previously classified collection to train its knowledge base is also an holdback.

The tests made to the clustering algorithms reveal good results, except for k-Means with the SS collection. We can see high values of precision, recall and accuracy for the first two tests, while the last one shows very low values, especially for the cluster with the informative links. The cluster quality measures reveal good sample partitioning for all tests — as proved by the high ISim values and the low ESim values — including the k-Means with the
SS collection test, which means that the bad results are not caused by the clustering algorithm itself, but by the input data — the feature values. Taking a closer look at the discriminating features (Disc. F.), we see that, unlike the first two tests, these features are not URL-related on the last test. The fact that the most discriminating feature is the Parent OutLinks feature, biases the clustering algorithm into partitioning the sample in one cluster containing pages with many links and another containing pages with few links. We believe that this happens, because the SS collection is too small and, as it contains only the source pages of its links, some features related to the link structure could not be used, e.g., Page InLinks, Page OutLinks, etc.

5 Conclusions and Further Work

All the proposed methods, except the feature selection method, proved effective in detecting non-informative links. k-Means tests showed poor performance with a small collection (SS), contrary to the good results obtained with a large collection (WBR-99).

Among all algorithms, k-means has the greatest advantage. Besides supporting large collections, the only human interaction needed for its operation is at its later phase, to decide which cluster contains the informative links and which cluster contains the non-informative links, unlike classification methods that require much more demanding tasks — previously classified collections and parameter tuning.

In the future, we intend to find a way of automatically detecting which cluster contains the informative links and which cluster contains the non-informative links, making the k-Means method able to detect non-informative links in a fully automated way.

6 References


