Link-Local Features for Hypertext Classification

Hervé Utard and Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering Group
Hochschulstraße 10, D-64289 Darmstadt, Germany
herve.utard@ingenieurs-supelec.org
juffi@ke.informatik.tu-darmstadt.de

Abstract. Previous work in hypertext classification has resulted in two principal approaches for incorporating information about the graph properties of the Web into the training of a classifier. The first approach uses the complete text of the neighboring pages, whereas the second approach uses only their class labels. In this paper, we argue that both approaches are unsatisfactory: the first one brings in too much irrelevant information, while the second approach is too coarse by abstracting the entire page into a single class label. We argue that one needs to focus on relevant parts of predecessor pages, namely on the region in the neighborhood of the origin of an incoming link. To this end, we will investigate different ways for extracting such features, and compare several different techniques for using them in a text classifier.

1 Introduction

Whereas the exploitation of the graph properties of the Web is already standard in search engine technology (and has resulted in a major break-through with the advent of Google's page-rank algorithm), its use for improving hypertext classification is still a hot research topic [5]. Conventional approaches to link-based text classification use either the entire text of a predecessor page or abstract this information into a class label for the entire page. Our working hypothesis is that the first approach is unsatisfactory, because not the entire page of a preceding link is relevant, and the second approach is too coarse because predecessor pages may be about entirely different topics, and may not fit into the predefined classification scheme. Nevertheless, it is clear that some *part* of a preceding page must contain information about the target page because there is a hyperlink connecting these two pages, and this link is (typically) annotated with its anchor text or text in the proximity of the hyperlink.

In this paper, we try to exploit this information by capturing different kinds of proximity (both at the structural and the textual level). We propose different types of features that can be extracted from HTML-documents and evaluate their utility for hypertext classification. Our results will demonstrate an improvement for several types of features (e.g., words in the neighborhood of a hyper-link), which could not be observed in previous work where the entire text of predecessor documents was used. Other features (e.g., headings) seem to be primarily useful in combination with others.

2 Hypertext Classification

It has been recognized early on that hyperlinks may provide important information about the content of a Web page. Anchor texts (texts on hyperlinks in an HTML document) of predecessor pages were already indexed by the WWW Worm, one of the first search engines and web crawlers [10]. Later, the importance of predecessor pages for ranking search results was established with the striking success of Google's page rank [1] and related techniques such as the HITS algorithm [8].

Not surprisingly, recent research has also looked at the potential of hyperlinks as additional information source for hypertext categorization tasks. Many authors addressed this problem in one way or another by merging (parts of) the text of the predecessor pages with the text of the page to classify, or by keeping a separate feature set for the predecessor pages. For example, Chakrabarti et al. [2] evaluate two variants: (1) appending the text of the neighboring (predecessor and successor) pages to the text of the target page, and (2) using two different sets of features, one for the target page and one resulting from a meta-document that is a concatenation of the neighboring pages. The results were negative: in two domains both approaches performed worse than the conventional technique that uses only features of the target document.

Chakrabarti et al. [2] concluded that the text from the neighbors is too unreliable to help classification. They proposed a different technique that included predictions for the class labels of the neighboring pages into the model. As the labels of the neighbors are not known, the implementation of this approach requires an iterative, EM-like technique for assigning the labels, because changing the class of a page may potentially change the class assignments for all its neighboring pages as well. The authors implemented a relaxation labeling technique, and showed that it improves performance over the standard text-based approach that ignores the hyperlink structure.

Lu and Getoor [9] generalize this work by enhancing the set of features that are considered for each neighboring page. In particular, they distinguish between in-neighbors, out-neighbors and co-linked neighbors. An in-neighbor, a *predecessor*, is a web page that contains a link pointing to the target page; an out-neighbor, a *successor*, is a web page that is linked by the target page; and the co-linked neighbors are web pages which have a common in-neighbor. Their conjecture is that cumulative statistics on the neighbors' class distribution are as informative as the identity of the neighbors, which requires much more storing space. An interesting characteristic of their model is that instead of going with the majority prediction, they learn how the category distribution of the neighbors affects the prediction. Like Chakrabarti et al. [2], they employ an iterative relaxation labeling technique, and compare two classifier types: a flat model where the local features and the non-local ones are concatenated into a common vector. The local and non-local features are thus not distinguished. The second model is obtained by combining the predictions of both a classifier based on the local features and a classifier based on the non-local features. The flat model is outperformed by the second one, which confirms the results of Chakrabarti.

3 Link-Local Features

The approaches described in the previous section basically employ two techniques for using the information on predecessor pages:

– use all words in predecessor documents as features
– abstract the predecessor documents into a single feature (the class label)

The main hypothesis of our work is that the first approach contains too much irrelevant information, whereas the second approach throws away too much relevant information: Using the entire text of the predecessor pages loses the focus on the relevant parts of the predecessor documents. For example, a page may be predecessor to several pages, each of which may belong to a different class. Thus, if the entire text is used, each example would be represented in the same way, and discrimination would be impossible. Focusing on the region around the respective hyperlinks, would result in different feature sets. On the other hand, abstracting the entire information on the page into a single class label is, again, problematic, for several reasons. First, a single page will always have the same class label, which may, again, lead to problems such as those discussed above (different pages that share the same set of predecessors are indiscriminable). Second, an important assumption that the approaches of Chakrabarti and Getoor make is that all (or a significant number) of the neighboring pages of a page can also be classified into the same set of classes. This, however, need not be the case. If you, for example, have the task of classifying web-pages about music between *official artist websites* and *fans websites*, the majority of the predecessors of an official website will belong to the *fans websites* set.

On the other hand, even if the entire page cannot be classified into the available set of classes, it can be expected that the parts of the predecessor pages around the hyperlink to the page in question contain relevant information. For example, Figure 1 shows three different feature types that could be important for classifying a target page: the anchor text of a page (left), the text in a paragraph around the hyperlink (center), and the text in a heading preceding the hyperlink (right). In all three cases, the text allows to classify the target page as a *professor*-page, whereas this cannot be inferred from the text on the target page itself. Thus, features that occur in a textual or structural proximity to the relevant out-link can be of importance. We will call such features *link-local*.

In previous work [6], we tried to utilize link-local features with so-called *hyperlink ensembles*. The idea is quite simple: instead of training a classifier that classifies *pages* based on the words that appear in their text, a classifier is trained that classifies *hyperlinks* according to the class of the pages they point to, based on the words that occur in their neighborhood of the link in the originating page. In the simplest case, we use the anchor text of the link. Consequently, each page will be assigned multiple predictions for its class membership, one for each incoming hyperlink. These individual predictions are then combined to a final prediction by some voting procedure. Thus, the technique is a member of the family of ensemble learning methods [4]. In a preliminary empirical evaluation

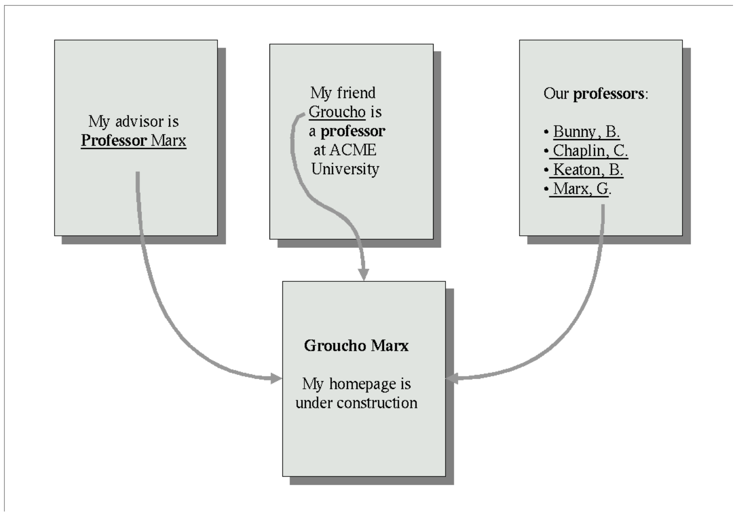


Fig. 1. Three different feature types on predecessor pages that might be helpful for classifying a page: anchor text, text in a paragraph, text in a preceding heading

in the Web→KB domain (where the task is to recognize typical entities in Computer Science departments, such as faculty, student, course, and project pages), hyperlink ensembles outperformed a conventional full-text classifier in a study that employed a variety of voting schemes for combining the individual classifiers and a variety of feature extraction techniques for representing the information around an incoming hyperlink (e.g., the anchor text on a hyperlink, the text in the sentence that contains the hyperlink, or the text of an entire paragraph). The overall classifier improved the full-text classifier from about 70% accuracy to about 85% accuracy in this domain.

The main point of this paper is to investigate this approach in more detail. Most importantly, we will demonstrate that these feature extraction techniques will result in even better performance on a conceptually simpler approach that combines all features into a single predecessor document.

4 Link-Local Features Extracted

As with classical text classification, we extract word-based features from the documents. We test different heuristic patterns to target the words which give the most relevant information about each link, namely the *anchor text*, the *paragraph around the link*, the *words neighboring the anchor*, the *headings structurally preceding the link*, *the heading of the list*, if the link is part of an HTML list and the *text of the document to classify*.

In particular, we extract the following features:

Link Description: The first spot is the *link description*, also named *anchor text*. It consists of the text that occurs between the HTML Tags `<A HREF=...>` and `</A>` of the link pointing to the page to classify.

Link Paragraph: The paragraph around the anchor text may also contain interesting words that describe the target page. We extract it in the features group *Link Paragraph*. We use the HTML tags `<P>` and `</P>` to find the borders of the paragraph.

Link Neighborhood: One difficulty with *Link Paragraph* is that the size of the paragraphs varies. The purity or the dilution of the clue features in the crowd of the words is not constant. We circumvent this problem with the features group *Link Neighborhood* where a fixed number of words neighboring the link are mined. The link description is excluded from *Link Neighborhood*. This feature location is an important source of information for the links with an irrelevant anchor text like "click here" or "next page".

Link Headings: In the *Link Headings* features, we extract the words occurring in the headings *structurally* preceding the link in the predecessor. We consider headings of type `H1`, `H2`, and `H3`.

Link List Headings: Sometimes, the link is part of an HTML list (tag `<UL>`). In this case, we store the preceding heading of each depth in the features group *Link List Headings*.

Own Text: The last but simplest feature set is the text of the target page itself. We extract it mainly in order to compare our model with traditional text-based classifiers.

These features are essentially the same that were suggested in [6]. However, in this work we extracted them in a more principled way using XPath structural patterns on the Document Object Model (DOM) representation of the documents. XPath is a language for navigating through elements and attributes on an XML document. It uses path expressions to select nodes or node-sets in an XML document. These path expressions are similar to those used for locating files in a file system. Table 1 lists the XPath expressions we use to extract the features from the predecessors of the target document.

The result of the *Link Description* request is the concatenation of the segments of the HTML file that occur between the HTML tags `<A HREF='Target_URL'>` and `</A>`. The other requests are simple extensions of the *Link Description* request. Once the anchor tag of the links is localized, *Link Paragraph* looks for its last ancestor of type Paragraph. *Link Headings* looks for the last occurrence of each heading level before the link, and *Link List Headings* looks for the last occurrence of each heading level before the beginning of the list. The implementation can certainly be improved, e.g., by trying to recognize headings that are not formatted with `<H?>` tags. For example, text immediately preceding an `<UL>` list might be interpreted as a heading for this list. However, we preferred to rely on the information that is directly provided by the HTML structure.

Table 1. XPath expressions for extracting the features.

Link Description	`//a[\@href='Target_URL']`
Link Paragraph	`//a[\@href='Target_URL']/ancestor::p[last()]`
Link Headings	`//a[\@href='Target_URL']/preceding::h1[last()]` | `//a[\@href='Target_URL']/preceding::h2[last()]` | `//a[\@href='Target_URL']/preceding::h3[last()]`
Link List Headings	`//a[\@href='Target_URL']/ancestor ::ul/preceding::h1[last()]` | `//a[\@href='Target_URL']/ancestor ::ul/preceding::h2[last()]` | `//a[\@href='Target_URL']/ancestor ::ul/preceding::h3[last()]`

As HTML is not XML compliant, we first translate the web pages from HTML format into XHTML format with the help of the Tidy program. We encountered a problem by this step because some HTML pages contain many syntax errors. Tidy cannot understand them all and can thus not output the XHTML translation of all the documents. We circumvent this difficulty by extracting the basic features (text of the target page) on the HTML page before the Tidy treatment and the complex features (link description, headings, ...) after the construction of the DOM tree by Tidy. If Tidy fails to convert HTML to proper XHTML, these features will not be extracted from this predecessor. As a page typically has several predecessors from which features are extracted, we can afford to do this.

5 Experimental Setup

5.1 Datasets

Allesklar (http://www.allesklar.de) is a German generic web directory referencing about 3 million of German web sites. Its tree organization begins with 16 main category roots, each one containing between 30,000 and 1,000,000 sites. The nodes of the tree are as specific categories as the node is deep. We chose 5 main categories, shown in Table 2.

We crawled each selected category with a breadth-first traversal in order to collect pages covering the whole category. We looked for hyperlink predecessors for each of these pages using the Altavista `link` request (for example, the request `link:europa.eu.int` retrieves all the web sites containing a link to the Web portal of the European Commission).

We looked for up to 25 predecessors per example. But we couldn't always find as many predecessors and some predecessors referenced by Altavista were not always reachable. However, only a small part of the examples have no predecessor and a large part of them has more than 10 predecessors. There is no important

Table 2. Class distribution for the Allesklar dataset

Category		3898 Examples
Arbeit & Beruf	(Employment & Jobs)	601 (15.42%)
Bildung & Wissenschaft	(Education & Science)	849 (21.78%)
Freizeit & Lifestyle	(Leisure & Life-style)	771 (19.78%)
Gesellschaft & Politik	(Society & Politics)	872 (22.37%)
Immobilien & Wohnen	(Accommodation & Living)	805 (20.65%)

difference between the categories from this point of view. Only the distribution of *Immobilien & Wohnen* is slightly biased towards fewer predecessors.

The *Web→KB dataset* [3] is a collection of web pages coming from the science departments of four main universities: *Cornell*, *Texas*, *Washington* and *Wisconsin*. One fifth group of pages named *misc* has been collected from various other universities. These pages are classified under seven categories: *course*, *department*, *faculty*, *project*, *staff*, *student* and *other*. The WebKB dataset is not equally distributed (Table 3): more than 45% of the examples are concentrated in the hold all category *other* while only 1.5% of the examples are classified as *staff* pages, which makes this dataset particularly difficult to classify.

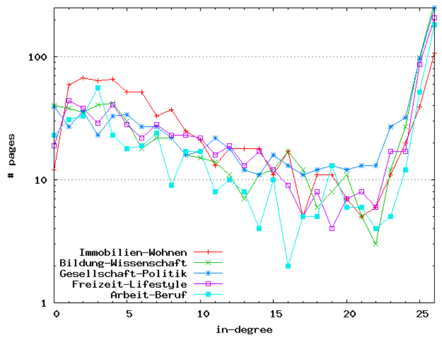
This dataset had been collected earlier [3], but we still had to discover its hyperlink graph by canonizing URLs and identifying all those that point into the dataset. Consequently, not all predecessors of a page are present in this dataset, only those that have been collected in the dataset. As a result, its graph structure is dramatically weaker connected than that of Allesklar. No predecessor could be found for 5082 pages of the dataset and only 67 pages have more than 10 predecessors. The connectivity of the two datasets is shown in Figure 2.

5.2 Classifier

As base classifier for the text classification experiments we used SVM-light in V6.01 [7]. For handling multiple classes we used two different binarization schemes: *round-robin* or *pairwise* classification, and a weighted version of *one-against-all* where a separate classifier is learned for each problem "class i against

Table 3. Class distribution for the WebKB dataset

Category	8264 Examples
student	1639 (19.83%)
faculty	1121 (13.56%)
course	926 (11.21%)
project	506 (6.12%)
department	181 (2.19%)
staff	135 (1.63%)
other	3756 (45.45%)



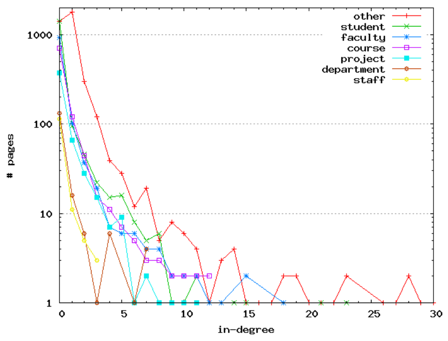


Fig. 2. Inlink-distribution of the documents of the Allesklar and Web→KB datasets

all $c - 1$ classes other than i". $c - 1$ votes are given to class i if the classifier predicts i, 1 vote is given to all other classes if the binary classifier does not predict i. Empirically, this seemed to work quite well.

Different feature types can be combined in two different ways:

Merging: Features of different sources are pooled together.
Tagging: Features of different sources are treated separately.

Thus, if the word "word" occurs in two different feature sources (e.g., in the anchor text and the preceding heading) it is treated as the same feature (and counted twice) in the merging approach, and it is treated as two different features (denoted as "Link Description:word" and "Link Headings:word") in the tagging approach.

As each page has up to 25 preceding pages, we need a technique for combining the features of the predecessors. We compared three different techniques for that:

Meta-Predecessor: The features of all predecessors are pooled together into a single document, called the *meta-predecessor*. This is basically identical to the first approach evaluated by [2], except that we do not pool all features of the predecessor documents but only those extracted by our feature extraction techniques.
Hyperlink Ensembles: Each predecessor is treated as a separate training document (labeled with class label of the page it points to). Predictions are made for each hyperlink separately and combined via voting (cf. Section 3 and [6]).
Mixed Approach: This combines both approaches: Training is like in Meta-predecessor, but for prediction, the trained classifier is used on each hyperlink separately, as in the Hyperlink Ensembles approach.

For all evaluations we used 10-fold stratified cross-validation. We measured recall, precision and the F_1 measure. We report macro-averaged results. As the classes are all of similar sizes, micro-averaged results are quite similar and can be found in [12].

6 Results

Unless mentioned otherwise, we use the following settings in the subsequent sections: the breadth of the *Link Neighborhood* feature set has been fixed to 20 words: 10 words before the anchor and 10 words after the anchor, the text of the anchor is excluded. We use *one-against-all* binarization for handling the multi-class problem, a *meta-predecessor* for combining the features of different predecessor pages, and features from multiple extraction patterns are *merged* together.

6.1 Comparing Different Feature Types

We first evaluate each feature extraction pattern in isolation (Tables 4 and 5). The two first lines represent the macro-averaged precision and the recall for the six classes, and the third line shows the F_1-value computed from these averages. The last two lines show the number of documents in the dataset that were covered by the feature pattern among 3898 documents, i.e., where at least one feature was detected with the respective method, and the number of different features extracted.

A few interesting observations can be made from this table. Although the pattern that covers the most examples is *Own Text*, it is not the pattern that derives the highest number of features. Both, *Link Neighborhood* and *Link Paragraph* retrieve a higher number of features.

For Allesklar, even though they are not applicable to all examples, they dominate a conventional text classifier in both, recall and precision. *Link Neighborhood* increases the F_1 value by 30 in comparison to *Own Text*. The other features seem to be less useful, at least on their own. In particular the features derived from the headings produce very low recall and are also not very precise.

For Web→KB, the results are not so good. Here, each of the link-based features covers only considerably less than half of the examples, and hence their performance is inferior to that of the *Own Text* classifier. It should, however, be noted that this is primarily due to the low number of predecessor pages that were included in the dataset. Also, the rather strict definition of the pattern extractors may have contributed to the bad performance. In [6], the experimental setup was

Table 4. Results for single feature patterns on the Web→KB dataset

	Link Description	Link Paragraph	Link Neighborhood	Link Headings	Link List Headings	Own Text
Precision	35.54%	29.17%	41.07%	28.35%	17.38%	45.37%
Recall	21.35%	16.71%	17.94%	17.37%	14.89%	24.71%
F1	26.68	21.25	24.97	21.54	16.04	31.99
Coverage	1,143	2,715	3,006	2,828	1,644	8,276
# Features	2,594	51,831	14,360	13,070	4,319	12,658

basically the same except that the feature extractors were defined more loosely and heuristically, which resulted in a higher coverage of individual features, and hence in a better overall performance.

6.2 Neighborhood of an Anchor

In the previous section, by far the best results were obtained with the use of features in the neighborhood of the links. However, the notion of neighboring words is vague and is computed with a parameter. In order to get an idea about the sensitivity of this parameter, we computed the macro-averaged F_1-score for each possible combination of 0 to 30 words before the anchor and 0 to 30 words after the anchor text. The anchor text itself (*Link Description*) is also included in these experiments.

The results for the Allesklar dataset show a continuous growth of the F_1-score (Figure 3). Before 20 words, the precision increases quickly. After 20 words, the precision still increases, but very slowly, while the dimensionality (the complexity of the classification problem) still grows. The best compromise for the scope of the neighborhood is around 20, which we distribute equally before and after the anchor (10 words before the anchor and 10 words after).

In the Allesklar dataset we did not observe a decrease of the F_1-score with growing size of the neighborhood in the parameter range that we had tried. This would suggest that using the full text of the predecessor pages would be the best strategy. However, the results of Chakrabarti et al. [2] suggest the opposite: in their experiments (on different datasets) this approach did not result in improved performance.

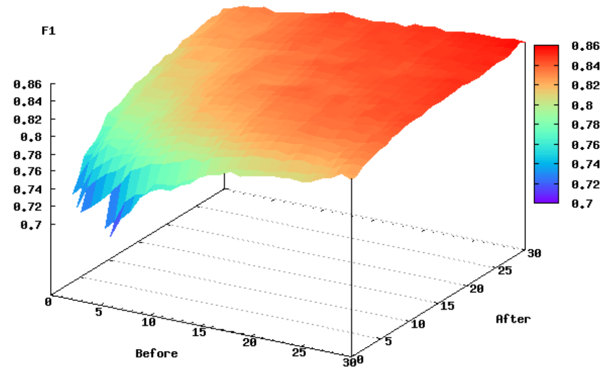
Our results on the Web→KB data are somewhat less reliable, but despite the high variance, we can observe that there is a clear ridge near the "10 words before" point, suggesting that choosing too large a neighborhood is futile. However, our experiments on this matter are obviously not yet conclusive, and need to be refined in future work.

6.3 Pairwise Combination of Features

In the previous sections we have seen that some of the features derived from predecessor pages are already sufficient to outperform a conventional text clas-

Table 5. Results for single feature patterns on the Allesklar dataset

	Link Description	Link Paragraph	Link Neighborhood	Link Headings	Link List Headings	Own Text
Precision	80.00%	79.15%	84.65%	71.80%	70.18%	71.67%
Recall	43.48%	34.30%	67.30%	29.33%	26.66%	32.17%
F1	56.34	47.86	74.98	41.65	38.64	44.41
Coverage	3,653	2,715	3,664	2,672	1,870	3,831
# Features	4,211	79,588	41,513	32,832	4,118	37,898



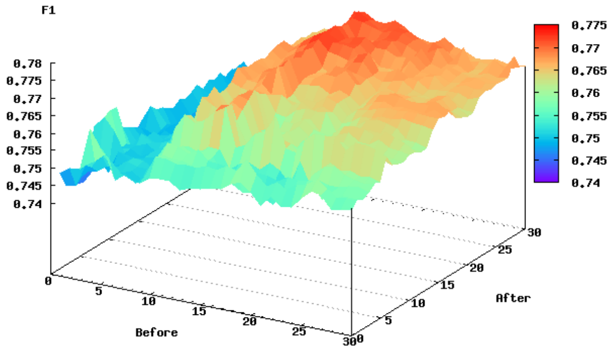


Fig. 3. F1-score for Allesklar (left) and Web→KB (right) for *Link Neighborhood* with different numbers of words before and after the anchor text

sifier. In this section we investigate whether we can further improve upon these results by combining the extracted features via *merging*.

In Table 6, we summarize these results for the Allesklar dataset. The diagonal of the table (with a dark gray background) shows the macro-averaged precision π and the F_1-score of each individual feature set (the other values can be found in Table 5). The upper triangle (light gray background) shows the recall (ρ) and precision (π) results for each combination of features in the respective lines and columns, the lower triangle (white background) shows their coverage c and the F_1-score.

Table 6. Results of pairwise combinations of features. The upper-right triangle shows precision (π) and recall (ρ), the lower-left shows coverage and the F_1-score

	Link Neighbor.	Link Description	Link List Headings	Link Headings	Link Paragraph	Own Text
Link Neighbor.	π=84.65% F_1=74.98	**π=84.89%** ρ=65.67%	**π=84.87%** ρ=67.31%	π=84.15% ρ=63.8%	π=82.72% ρ=58.88%	π=82.58% ρ=58.44%
Link Description	c=3678 F_1=74.05	π=80.00% F_1=56.34	**π=80.01%** ρ=42.15%	π=76.68% ρ=38.5%	π=76.44% ρ=36.19%	π=75.75% ρ=37.1%
Link List Headings	c=3665 **F_1=75.08**	c=3653 F_1=55.21	π=70.18% F_1=38.64	**π=71.83%** ρ=28.78%	**π=79.66%** ρ=26.77%	**π=72.36%** ρ=33.82%
Link Headings	c=3665 F_1=72.58	c=3653 F_1=51.26	c=2744 F_1=41.09	π=71.80% F_1=41.65	π=70.09% ρ=26.62%	**π=72.34%** ρ=35.11%
Link Paragraph	c=3667 F_1=68.79	c=3655 F_1=49.12	c=3013 F_1=40.07	c=3103 F_1=38.59	π=79.15% F_1=47.86	π=72.51% ρ=34.87%
Own Text	c=3898 F_1=68.44	c=3898 F_1=49.81	c=3864 **F_1=46.10**	c=3879 **F_1=47.28**	c=3882 F_1=47.09	π=71.67% F_1=44.41

In **bold** font, we show the combinations that outperform both of its constituent patterns in terms of precision (upper triangle) or F_1 (lower triangle). It can be seen that, although using two feature types instead of one does not always increase the performance, it may yield a substantial improvement in some cases. For example, the headings of a preceding list improve the precision in all combinations with other features. However, this gain has to be paid with a loss in recall. In terms of the F_1-score, not many improvements are observable. Heading-based features occassionally make a difference, but these improvements are not likely to be of significance (except when combined with the original text).

The detailed results for Web→KB can be found in [12]. They were qualitatively similar, as can also be seen from the ranking of the individual features (Tables 4 and 5). The best-performing combination was *Link Description* with *Link Paragraph*, which had a precision of 56.66% at a recall of 20.35%.

In summary, these results show that combining these feature sets may be helpful, and even feature types that, on their own, do not yield good results, may be very useful in combination with other types.

6.4 Different Classification Methods

In this section, we study the influence of the choice between the options discussed in Section 5.2, namely between *meta predecessor*, *hyperlink ensembles*, and the *mixed approach*, between the binarization algorithms *one-against-all* or *round-robin* and between the feature combiners *merging* or *tagging*. In total, there are 12 possible combinations of these options, resulting in 12 different methods.

We ran the classification process for the 6 feature sources available and for the 15 combinations of two of those feature sources. Detailed results can be found in [12]. For a summary statistic, we report the average rank (1–12, in terms of

Table 7. Ranking of the different methods for Alleksklar

Combination	**Binarization**	**Non-local**	**Avg. Rank**	**Avg. π**	**Avg. ρ**
Merging	One against all	Meta predecessor	1.14	78.37%	43.76%
Tagging	One against all	Meta predecessor	1.86	77.35%	42.25%
Merging	One against all	Hyperlink Ensembles	2.86	73.17%	33.42%
Tagging	One against all	Hyperlink Ensembles	3.38	72.43%	32.51%
Merging	One against all	Mixed Approach	5.57	68.98%	37.77%
Tagging	One against all	Mixed Approach	5.95	67.85%	36.61%
Merging	Round Robin	Meta predecessor	6.43	66.32%	59.51%
Tagging	Round Robin	Meta predecessor	7.00	64.95%	57.95%
Merging	Round Robin	Hyperlink Ensembles	8.14	61.64%	48.36%
Tagging	Round Robin	Hyperlink Ensembles	9.10	59.83%	47.50%
Merging	Round Robin	Mixed Approach	10.57	57.71%	50.44%
Tagging	Round Robin	Mixed Approach	10.86	56.00%	48.62%

precision), and the average precision and average recall for each of the methods for each of those 21 atomic experiments on the Allesklar dataset (Table 7).

In this domain, the observed pattern is very regular: our version of *one-against-all* consistently outperformed *pairwise classification*, *merging* consistently outperformed *tagging*, and the *meta predecessor* consistently outperformed the *hyperlink ensembles* and the *mixed approach*. The results in the Web→KB domain were similar but not as consistent [12].

7 Conclusions

The most important result of our study is that using features from predecessor documents may result in a largely improved classification performance, thereby shedding new light on the negative results of [2]. The key difference of our approach is that we suggest to focus on the part of the predecessor texts that is in the neighborhood of the outgoing link, whereas [2] use the complete text of neighboring pages. In our experiments, this technique worked best when used with the simplest approach, namely to merge the extracted features into a single predecessor document. However, we could not compare our approach to all competitive approaches. For example, we have not yet performed a direct comparison to approaches based on the neighbors' class distribution [2, 9], which, however, we think are not as general as our technique. We also have not yet tried the full set of extracted features, only pairwise comparisons. Our feature extraction methods could also form the basis of multi-view learning algorithms [11], and it would be interesting to find out whether the advantage of multi-view learning over single-view learning over the union of the views carries over to this scenario as well. Finally, we note that the problem of extracting useful features in the neighborhood of outgoing links is quite similar to approaches that formulate information extraction as a classification problem, for which local features are derived around the item to be extracted.

References

1. S. Brin and L. Page: The anatomy of a large-scale hypertextual Web search engine. *Computer Networks* (1998) 30(1–7):107–117. Proceedings of the 7th International World Wide Web Conference (WWW-7), Brisbane, Australia.
2. S. Chakrabarti, B. Dom, and P. Indyk: Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management on Data*. ACM Press, Seattle WA (1998), pp. 307–318.
3. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery: Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence* (2000) 118(1-2):69–114
4. T. G. Dietterich: Ensemble methods in machine learning. In J. Kittler and F. Roli (eds.) *Proccedings of the First International Workshop on Multiple Classifier Systems.* Springer-Verlag (2000), p. 1.
5. J. Fürnkranz: Web Mining. In O. Maimon and L. Rokach (eds.) *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag (2005), pp. 137–142.
6. J. Fürnkranz: Hyperlink ensembles: A case study in hypertext classification. *Information Fusion* (2002) 3(4):299–312. Special Issue on Fusion of Multiple Classifiers.
7. T. Joachims: Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol (eds.) *Proceedings of 10th European Conference on Machine Learning (ECML-98)*. Springer-Verlag, Chemnitz Germany (1998), pp. 137–142.
8. J. M. Kleinberg: Authoritative sources in a hyperlinked environment. *Journal of the ACM* (1999) 46(5):604–632. ISSN 0004-5411.
9. Q. Lu and L. Getoor: Link-based classification. In *Proceedings of the International Conference on Machine Learning (ICML-03)* (2003), pp. 496–503.
10. O. A. McBryan: GENVL and WWWW: Tools for taming the Web. In *Proceedings of the 1st World-Wide Web Conference (WWW-1)*. Elsevier, Geneva Switzerland (1994), pp. 58–67.
11. S. Rüping and T. Scheffer (eds.): *Proceedings of the ICML-05 Workshop on Learning With Multiple Views*. Bonn Germany (2005)
12. H. Utard: *Hypertext classification.* Master's thesis, TU Darmstadt, Knowledge Engineering Group (2005)