ARKTiS - A Fast Tag Recommender System Based On Heuristics

Thomas Kleinbauer and Sebastian Germesin

German Research Center for Artificial Intelligence (DFKI)
66123 Saarbrücken
Germany
firstname.lastname@dfki.de

Abstract. This paper describes the tag recommender system ARKTiS, our contribution to the 2009 ECML PKDD tag discovery challenge. ARKTiS consists of two separate modules for BibTeX entries and for bookmarked web pages. For generating tags, we distinguish between so-called *internal* and *external* methods, depending on whether a tag was extracted from the given information about a resource or whether additional resources were employed.

1 Introduction

The role of the end-user in the world wide web (WWW) has undergone a substantial change in recent years from a passive consumer of relatively static web pages to a central content producer. The addition of backchannels from WWW clients to internet servers empowers non-expert users to actively participate in the generation of web content. This has led to a new paradigm of usage, colloquially coined "Web 2.0" [1].

These novel kinds of interactions can be divided into two categories: producing or making accessible of new information (e.g., web logs, forums, wiki wikis, etc.) and enriching already existing contents (e.g., consumer reviews, recommendations, tagging, etc.). One interpretation of the second type of interaction is that it provides means to cope with one of the problems generated by the first type of interaction, namely the massive growth of available content and the increasing difficulty for traditional information retrieval approaches to support efficient access to the contained information. In this sense, the *meta-content* produced in the second kind of interaction can be construed as mainly serving as a navigation aid in an increasingly complex, but weakly structured online-world.

In particular, the possibility for users to attach keywords to web resources in order to describe or classify their contents bares an enormous potential for structuring information which facilitates subsequent access by both the original user and other users. This task, commonly referred to as *tagging* is simple enough not to scare users away, yet the benefit of web resources annotated in such a way is obvious enough to keep the motivation to supply tags high. The process of attaching tags to web resource must therefore show a fine balance between simplicity and quality. Both of these properties could be greatly improved if an automatic system could support a user by recommending tags for a given resource.

In this paper, we describe the `ARKTiS` system, developed to recommend tags to a user for two specific types of web resources. The system was developed as a contribution to an international challenge as part of the 2009 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2009).

2 Task and Data

The goal of this challenge is the implementation of a system that can automatically generate tag recommendations[1] for a given resource. Here, a resource is either a bookmark of a web page or a BibTeX entry for different kinds of documents. Tags typically are short English words although they may also be artificially created terms, generated e.g. by concatenating words ("indexforum") or by using abbreviations ("langfr", for "language french"). The number of tags a system may generate is restricted to a maximum of five.

To prepare for the challenge, a training set of manually tagged resources was provided. The data set consists of web page bookmarks and BibTeX entries taken from the BibSonomy project[2]. Each entry has a unique id and was tagged by at least two users. Thus a point in the data set can be viewed as a triple `<resource-id, user-id, tags>`.

For each resource, the corpus contains meta-data describing the resource with a number of different fields. These fields are different for BibTeX entries and for bookmark entries. For instance, the meta-data for BibTeX entries contain fields describing the title of an article, the authors, the year of publication, or the number of pages. For bookmarks, one field gives a short description of the resource while another one contains the web pages' URL. A full list of all available fields can be found on the homepage of the challenge.

In total, the training corpus contains 41,268 entries for bookmarks and 22,852 BibTeX entries, annotated with 1,3276 unique tags by 1,185 different users.

The data of the actual challenge (the *eval set*), was provided 48 hours before the submission deadline. This set consists of unseen data that each system had to tag and all results that are presented and analyzed in section 5 were achieved on this set. The evaluation data set contains 43,002 entries in total, with 26,104 BibTeX- and 16,898 bookmark-entries - circa two thirds of the amount of the training data.

[1] For the remainder of this paper, we will refer to this process as "(automatic) tagging"

[2] http://www.bibsonomy.org

3 Approach

One key observation for our participation in the ECML PKDD challenge was that time played a central role in two different senses. First, the task required each system to suggest tags for quite a large number of resources in a rather short period of time: 43,002 entries in only 48 hours, including retrieval and parsing of the test data as well as formatting and uploading of the final result data. Second, since the challenge had a fixed deadline, the time to develop a running system faced a naturally limit with the release of the 48 hour evaluation period.

Both points had a direct influence on the conceptualization and realization of our system ARKTiS, in that a number of more sophisticated ideas had to be sacrificed. As a result, ARKTiS can be seen as an exercise in software engineering rather than thorough science. The final system implements straight-forward strategies with a focus on robustness and processing speed.

3.1 Motivation

As outlined above, the training corpus contains 13276 different tags for over 41268 data points, a proportion that indicates a potential data sparseness problem for classical machine learning approaches. We therefore opted for an algorithmic approach instead, based on heuristic considerations.

Since the desired system output is (English) words, we can distinguish two potential sources for output: from within the resource itself (internal words) or from outside material (external words). This distinction is in so far blurred, as the system input actually consists not of the resources themselves, but rather of meta-data. In so far, even tags taken from the resources themselves could be argued to be external. We take the view that words stemming from meta-data or the resources referred to by the meta-data are considered internal.

Since we could not hope to implement a competitive system, we were mainly interested in how useful such a distinction would be in terms of recommending tags. Although we concentrated on internal methods as described below, we explored using document similarity measures in the BibTeX module to re-use tags that were manually assigned to documents similar to the current system input. Also, some of our implemented techniques, such as translating German words from the original resource to English, can be considered borderline between internal and external.

For the internal approaches, we looked at the task of tagging a resource as an analogy to automatic text summarization, somewhat taken to an extreme where a “summary” consists only of five words. In *extractive summarization*, summaries of documents are generated by identifying those sentences in a document which when concatenated serve as surrogate for the original document. In that spirit, tagging becomes the task of identifying those words from a resource that together describe the “aboutness” of the resource.

3.2 Related Work

A number of researchers in recent years have engaged in the task of developing an automatic tagging system. [2] use a natural language resources to suggest tags for BibTEX entries and web pages. They include conceptual information external resources, such as WordNet [3], to create the notion of a "concept space". On top of this notion, they exploit the textual content that can be associated with bookmarks, documents and users and generate models within the concept space or the tag space derived from the training data.

[4] model the problem of automated tag suggestion as a multi-label text classification problem with tags as categories.

In [5], the TagAssist system focuses on the task of the generation of tags for web-log entries. They access tags of similar documents in a similar spirit to our own method described in section 4.1.

In addition to these concrete systems, we find automatic tagging to bare some similarities with research in automatic extractive summarization. In both task, the identification of salient portions of a resource's text is a central consideration. For tagging which reduces extraction single words, we call such a method an *internal* approach. (s. section 3.1).

For instance, in [6], the author conducted a wide range of tests to find predictive features for relevant sentences. Despite relying on manual experiments, the general results from this early research were later confirmed by machine learning approaches, e.g., [7]. For the bookmark modules, both the *title* and the *first sentence* heuristic (see 4.2) were inspired by these findings.

4 Taggers

As hinted by the data set, the task can be viewed as two different sub-tasks, the tagging of BibTEX entries and the tagging of bookmarked web pages. Consequently, the `ARKTiS` system consists of two independent modules which are both instances of a common framework architecture, depicted in Figure 1. The modules can be run in two distinct processes.

Efficient processing of the input data is an important requirement for this challenge. In a sequential architecture, processing 43002 data points in 48 hours would leave a tagging system about 4 seconds per data point on average. Given that the data points contain only metadata and that the actual documents, if needed, have to be retrieved through the internet and parsed, a time span of 4 seconds poses quite a strong limitation on the complexity of the performed computations. Running multiple instances of taggers concurrently relaxes this limitation. In our current setup, both modules for BibTEX and bookmark tagging internally run ten tagging threads in parallel which increases the maximum average processing time to 80 seconds per data point.

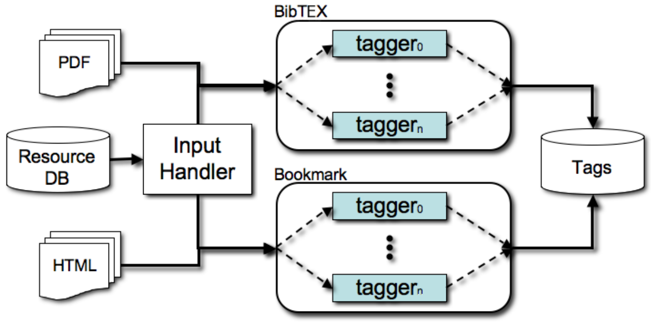

Fig. 1. The parallel architecture of `ARKTiS`.

4.1 The BibTEXTagger

The tagging system responsible for the BibTEX entries uses a combination of internal and external techniques. A thorough investigation of the provided training material showed that most of the entries (95.4%) do not contain a valid link to the actual PDF document. This is unfortunate, as it limits *internal* approaches which draw tags from the contents of the document in question.

Internal approach To compensate the cases in which the PDF document is unavailable, we use the remaining information from the meta-data, namely the *title* of the document, its *description* and its *abstract*. The employed approach analyzes these fields and extracts tags directly out of their textual information. Before that, we lowercase all words in the text of each field and remove all punctuation and symbols. After that, we apply the POS tagging system described in [8] to extract content-words - nouns, adjectives and verbs - out of the text. The sequential processing of the text is shown below:

	Pre-processing
Original title:	`The PageRank Citation Ranking: Bringing Order to the Web`
Lowercased:	`the pagerank citation ranking: bringing order to the web`
No symbols:	`the pagerank citation ranking bringing order to the web`
	Extraction of Tags
POS-tagged:	`the/DT pagerank/NN citation/NN ranking/NN bringing/VBG order/NN to/TO the/DT web/NN`
Content words:	`pagerank citation ranking order web`

Fig. 2. Sequential processing of textual contents

After removing stop-words, the remaining words are directly used as tags, giving preference to tags stemming from the *title* field over those from *description* field over those from the *abstract* field. Only the first five tags are returned after filtering out duplicates.

External approach For the remaining entries - where the source documents were available - we use a corpus-based approach inspired by standard information retrieval techniques. The idea here is that if a new document is similar to a document from the training corpus, we may re-use the tags that have been added manually to the training document.

Hence, this part of the tagger first ranks all documents from the training data by similarity to the current document. A second step then takes tags from the documents in rank order and returns the first five of them, discarding duplicates.

To do so, all documents in the corpus are first transformed from PDF format to plain text by the PDFBox toolkit[3]. After that, the whole text is segmented into sentences using punctuation information (`.!?;:\n`) and then pre-processed in the same way as described in the internal approach. After removing non-content words, we calculate tf.idf values for each word in the document, resulting in the following mapping:

`<list of tags>` → `<vector of TF/IDF-values>`

A tf.idf value is a value that calculates the relative importance of the word w_i for the current document j, in relation to a set of documents D (see equation 1, where $n_{i,j}$ is the number of occurrence of the word i in document j).

$$tfidf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} * \log \frac{|D|}{|\{d \in D : w_i \in d\}|} \tag{1}$$

This procedure is, of course, carried out only once and the resulting mapping is stored offline. In the actual tagging process, we generate the vector of tf.idf values in the same way for the document to tag and compare the resulting vector to with all document vectors in the corpus. We have experimented with two different similarity measures.

The first variant compares of two documents by the normalized distance between their tf.idf vectors, as shown in equation 2.

$$sim(\boldsymbol{t_0}, \boldsymbol{t_1}) = \frac{\sum_{i=0}^{N-1} |t_0[i] - t_1[i]|}{\sum_{i=0}^{N-1} t_0[i] + t_1[i]} \tag{2}$$

In addition, we implemented *cosine* similarity that measures similarity by the cosine of the angle Θ between the two vectors that the two documents describe (equation 3).

$$sim(\boldsymbol{t_0}, \boldsymbol{t_1}) = \cos\Theta = \frac{\sum_{i=0}^{N-1} t_0[i] t_1[i]}{\sqrt{\sum_{i=0}^{N-1} t_0[i]} \sqrt{\sum_{i=0}^{N-1} t_1[i]}} \tag{3}$$

In our experiments, the normalized distance measure yielded better performance than cosine similarity and consequently we used only the former in the final system.

[3] http://incubator.apache.org/pdfbox/

4.2 The Bookmark Tagger

As in the case for the BibTeX tagger, the bookmark tagger relies on relatively simple heuristics to determine the keywords to recommend. The input data provides two kinds of information, the URL of the web page to tag and a short description which in some cases is identical to the web page's title string.

Processing the URL field In our system, the URL is used to fetch the contents of the actual web page, but since the domain name and path may already contain candidate terms, the URL string is also processed itself, in three sequential steps: *tokenizing*, *filtering*, and *dict/split*.

For the tokenization, the URL is split up at every non-letter non-digit character, such as a forward slash. By matching against a manually crafted blacklist of terms generally considered uninformative, typical artifacts such as "www" or "html" that result from the tokenization process are filtered out. The following examples illustrate these two steps:

Original URL:	`http://www.example.com/new-example/de/bibtex.htm`
Tokenizing:	`http www example com new example de bibtex htm`
Filtering:	`example new example de bibtex`

Original URL:	`http://www.coloradoboomerangs.com`
Tokenizing:	`http www coloradoboomerangs com`
Filtering:	`coloradoboomerangs`

A dictionary of American English together with a list of the names of all articles in the English Wikipedia[4] of 2007 are used to check if the resulting tokens are actual words. The rationale for incorporating Wikipedia is that it gives additional terms from article titles which often are not found in a dictionary, such as, e.g. technical terms ("bibtex"). If a token cannot be found in either list, we try to split the token up into two sub-tokens which, in case they are both contained in the dictionary, are then used instead of the original tokens. This idea is based on the observation that domain names in particular are sometimes a concatenation of two terms.

Applied to the above example, this step generates the following keyword lists:

Keywords:	`example new example de bibtex`
Dict/Split:	`example new example bibtex`

Keywords:	`coloradoboomerangs`
Dict/Split:	`colorado boomerangs`

[4] http://en.wikipedia.org

Processing the description field The description that is part of the input data is tokenized in the same way. However, no further attempts are made to filter out tokenization artifacts or to split the resulting tokens into sub-parts in case they are not contained in the dictionary. In other words, of the above three steps, only *tokenizing* is performed on the description of the bookmark.

Processing the bookmarked web page With the provided URL, the content of the given web page is retrieved at run-time. We do not attempt to detect whether the server returns an actual content page or a specialized message, such as a `HTTP 404 Not Found` error page. After the HTML content of a web page has been downloaded, three different extraction methods are applied: *HTML-meta*, *title*, and *first sentence*.

The first method operates on the head section of the document where it locates and parses the `<meta>` elements "keywords" and "description". The contents of these elements are provided by the author of the HTML document and may contain valuable hints on what the document actually is about. The contents are extracted and then undergo the same tokenizing procedure as described above.

```
            <html>
              <head>
                <meta name=keywords content="example, sample">
                <meta name="description" content="A made-up
HTML:                 example webpage">
                ...
              </head>
              ...
            </html>

Extracting: example, sample
            a made-up example webpage
Tokenizing: example sample a made up example webpage
```

Also in the head section is the declaration of the title of the document. This is not only intuitively a good source for relevant keywords, but research in the field of automatic text summarization has also shown in the past that headings contain informative content [9].

```
HTML:       <title>Hello, world - again, an example</title>
Extracting: hello, world - again, an example
Tokenizing: hello world again an example
```

Another finding from summarization research is that locational cues work well for determining relevant content words. To apply this insight to the task at hand, the third document-based method tokenizes the first sentence of each HTML document in the same manner described above.

The result of these steps is a set of basic terms. For the final recommendation, two more processing steps are performed, a ranking step and a normalization step.

Ranking keywords For the ranking, each of the previously extracted keywords is described according to four predefined dimensions: Source, InDict, POS, and Navigational.

The values for these dimensions are floating point numbers that represent how valuable a keyword is with respect to being among the recommended tags. For instance, analog to the first step described above, the Source dimension may receive one of the following values:

- URL (= 0.4)
- Description (= 1)
- HTML-Meta (= 1)
- Title (= 0.8)
- First sentence (= 0.9)

The other dimensions describe whether a keyword is found in the English dictionary (and/or list of Wikipedia articles), its part of speech (NN = 1, NNS = 0.9, VBG = 0.8, VERB = 0.5, OTHER = −4) and whether it is found on a blacklist of navigational terms, such as “impress”, “home”, etc. which was created manually by the authors. As with other heuristics, the idea of using such *stigma* word lists can also be tracked back to early summarization research, see e.g. [6].

To rank the keywords, a weighted sum of the four values is computed for each keyword. Since a training corpus was available, good practical weights could have been determined with a machine learning approach. Unfortunately, since time was scarce, we had to estimate sensible weights by hand–inspecting the performance of the tagger on selected samples from the training corpus helped in this part of the development.

Normalization In a final normalization step, a German-English dictionary is used to translate German keywords to English ones and to re-weight those keywords that contain other keywords as sub-strings. In such a case it was speculated that the keyword contained as a sub-string would likely be the more general term and thus its final score was slightly increased.

5 Results

In the evaluation, the results of our tagging system `ARKTiS` had to be compared against the tags that were annotated by a human. The test data were provided 48 hours before the submission deadline. Considering at most five tags per entry, the evaluation uses *precision, recall* and *f-score* values as measurements. In the following, we will present our results, that were achieved on this data and compare them against a baseline system. The baseline system predicts the five most common tags from the training data (Figure 3) to each input entry.

BibTeX: `ccp jrr programming genetic algorithms`
Bookmarks: `software indexforum video zzztosort bookmarks`

Fig. 3. Most common tags in the data

The results of the baseline system are presented in Table 1 where we can see a maximum f-score of 0.55%. Comparing this to the results of `ARKTiS` (Table 2), we can see that our system clearly outperforms the baseline with an f-score of almost 11%.

#(tags)	recall	precision	f-score
1	0.0025	0.0114	0.0041
2	0.0025	0.0057	0.0035
3	0.0039	0.0057	0.0046
4	0.0041	0.0046	0.0043
5	0.0053	0.0058	0.0055

Table 1. Baseline

#(tags)	recall	precision	f-score
1	0.0305	0.1072	0.0475
2	0.0595	0.1082	0.0768
3	0.0839	0.1064	0.0938
4	0.1032	0.1032	0.1032
5	0.1179	0.0995	0.1079

Table 2. Results of the `ARKTiS` system

6 Conclusion and Future Work

Our work shows that is is possible to design and implement a basic tag recommender system even with a very limited development time. The two tracks, BibTeX and bookmark tagging, were designed and realized independently but on top of a common, concurrent framework.

The overall task can be considered challenging, especially if results are evaluated on the basis of recall and precision: our final system scored a rather low 11% f-score. The large number of different gold-standard tags makes this number difficult to interpret; however, it is clear that it leaves room for improvement. The winning entry of the 2009 challenged reached an f-score of 19 percent.

In a more detailed analysis, we found that the bookmark module outperformed the BibTeX module to some degree. As described above, the two modules employ rather different approaches, thus a next logical step will be to combine the best ideas from both modules.

The biggest drawback for `ARKTiS` as described in this paper was the fact that we entered the ECML PKDD challenge at a late point. As a consequence, a number of interesting and more sophisticated ideas had to be left out of the system purely due to the lack of implementation time. For instance, the use of tf.idfscores in the BibTEX module is very limited, as is the use of content terms beyond the first sentence in the bookmark module.

At the same time, the ARKTiS system has proven its robustness and will be a good starting point for further research in the area of automatic tagging.

References

1. Oreilly, T.: What is web 2.0: Design patterns and business models for the next generation of software. Social Science Research Network Working Paper Series (2005)
2. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc'08: Tag recommendations using bookmark content. In: ECML PKDD Discovery Challenge 2008. (2008)
3. Fellbaum, C., ed.: WordNet–An Electronic Lexical Database. MIT Press (1998)
4. Vlahavas, I.K.G.T.I.: Multilabel text classification for automated tag suggestion. In: ECML PKDD Discovery Challenge 2008. (2008)
5. Mishne, G.: Autotag: A collaborative approach to automated tag assignment to weblog posts. In: Proceedings of WWW'06, Edinburgh, Scotland (2006)
6. Edmundson, H.P.: New methods in automatic extracting. Journal of the ACM **16**(2) (1969) 264–285
7. Kupiec, J., Pedersen, J., Chen, F.: A trainable document summarizer. In: SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (1995) 68–73
8. Phan, X.: CRFTagger: CRF English POS Tagger. (2006)
9. Mani, I., Maybury, M.T., eds.: Advances in Automatic Text Summarization. The MIT Press, Cambridge, MA (1998)