Web content annotation

Marian Mach

Technical University of Kosice
Letná 9, 042 00 Košice, Slovakia
Marian.Mach@tuke.sk

Abstract. *Bookmarking represents a typical activity of users while surfing web pages. When they come across an interesting information and want to remember where the information can be found, they can create a bookmark and store it in a bookmarking system.*
Unfortunately, currently available systems offer only limited functionality, which could be rather restricting. Two main limitations are granularity of bookmarked information and passive user support in creating new bookmarks. In order to overcome these limitations, we have developed a Firefox plug-in enabling more advanced bookmarking.

Keywords. passive and active bookmarking, text classification, text annotation, Naive Bayes

1 Introduction

Currently, the Web represents one of the most used sources of information. It has penetrated all areas of our lives. Probably every web user has already faced a need to store the location of a currently visited page. The reason for this storing can be manifold – from visiting a page (content of which could be useful for user in future) more or less randomly through spending quite a lot of time to search for the content to enabling an easy access for future frequent visits.

This problem is commonly being solved by using various methods of web content annotation and bookmarking. Various bookmarking systems enable users to store the address of a web page for subsequent easy retrieval of this page. Pages can be annotated (most commonly in the form of category assignment) to enable some form of bookkeeping of the bookmarked pages.

Some web pages provide heterogeneous content located on one page. Typical examples of such pages are web portals publishing news from very different categories. A typical user is not interested in all the content offered by such pages but only in a fragment of the content. After repeated visiting a page (utilising the address of the page stored in a bookmarking system), user is confronted with the all content offered by the retrieved page. That means to scan the content repeatedly to find the fragment user is interested in.

A remedy for this repeated search within a page is increasing the granularity of bookmarking – the possibility not to bookmark a web page as a whole but to enable to select a content fragment with subsequent bookmarking this selected fragment.

Dealing with content fragments requires from users to perform additional activities. If user wants to bookmark only a fragment of a page instead of the whole web page, then he must specify this fragment before bookmarking it. If user wants to define several fragments on the same web page and to bookmark them separately, he must also label these fragments to distinguish one fragment from other fragments.

A natural step in helping users in dealing with content fragments is the transition from a passive bookmarking to an active bookmarking. Instead of selecting fragments manually after visiting a new page, user can be offered with fragments, which could be potentially interesting for him, together with suggested labelling. The only activity required from user is to accept (or deny or modify) suggested bookmarking.

The paper introduces a Firefox plug-in for active bookmarking of content fragments located on web pages. It enables user to select fragments and define categories the fragments belong to. Based on collected data, the plug-in is able to locate fragments, which could be interesting for a particular user, along with suggested categories for them.

2 Plug-in for active bookmarking

In order to support an idea of active bookmarking, a bookmarking system has been designed and implemented. The system combines the possibility to bookmark information fragments located on web pages with the possibility to search for interesting information fragments within a web page with subsequent automatic fragment identification and annotation.

The system was implemented as a plug-in into Mozilla Firefox browser. It was designed for text in English.

2.1 Architecture

An architecture of a simple active bookmarking system is depicted in Fig. 1. The left part of the architecture represents a passive bookmarking subsystem while the right part presents plug-in enrichment enabling active fragment bookmarking.

The designed plug-in enables for user to create a personal collection of bookmarks. A new bookmark can be created using the following steps:

- selection of an interested text fragment on a displayed web page,
- assignment of one or several categories to the selected fragment,
- processing of the bookmarked fragment.

After a fragment is selected and labelled, it is subsequently processed. The processing includes not only storing the bookmarked fragment into the *bookmark database* but updating *keyword database* as well.

The keyword database serves for an embedded learning mechanism. This mechanism enables to find interesting fragments within a newly displayed web page in an automatic way. The found fragments are classified into possible categories and viable alternatives are offered as suggested labelling of the fragments.

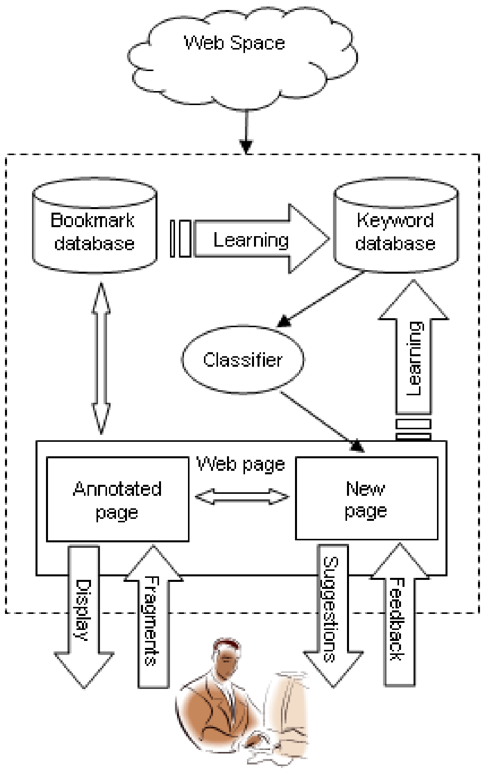

Figure 1: Architecture of a bookmarking plug-in

Besides accepting (with or without modification) or denying the system's suggestion, user can provide a feedback on the suggested fragments and/or their labelling. Based on this feedback the keyword database can be updated in order to improve system's classification abilities.

2.2 User interface

The plug-in is controlled using a context menu appearing after clicking the right mouse button anywhere within the page area. The plug-in adds a submenu in the context menu.

In order to add a new bookmark, user has to select a fragment within the displayed page using mouse and select an appropriate item from the context submenu. A new pop-up window appears and user can select/modify categories assigned to the fragment as well as fill in optional data attributes.

It is possible to select a bookmarked fragment and open the relevant page (a page containing the fragment) in a new browser tab. After the page is retrieved and displayed, defined fragments are highlighted. Positioning cursor over a highlighted fragment results in visualising all information relevant to this fragment. Such situation is depicted in Fig. 2.

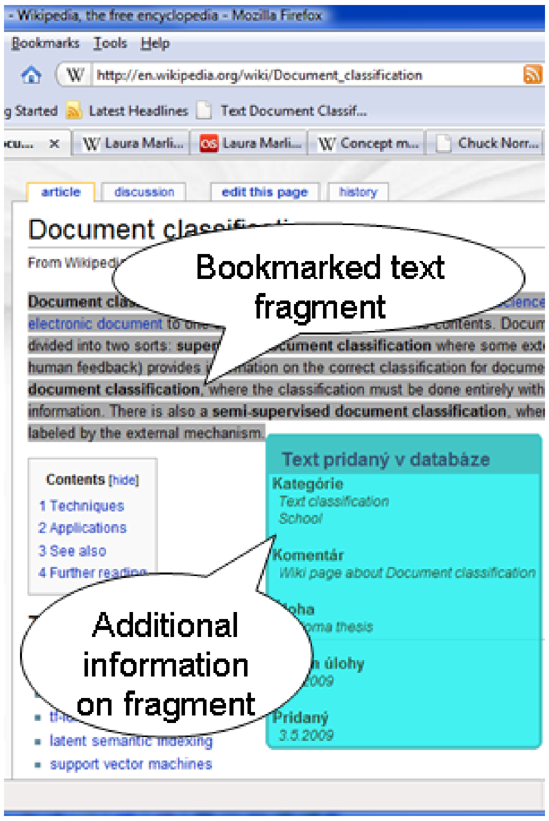

Figure 2: Highlighted bookmarked fragment and additional fragment information

Each category uses a defined colour (selected from a palette of available colours). When a fragment with an assigned category is displayed, the fragment is highlighted using the colour corresponding to the category of the fragment (if the fragment is categorised into more categories, then the colour of the first category is used). Such simple colour code enables better user orientation, especially in a situation when a page with several fragments from different categories is displayed.

When a new page is loaded into the browser, selecting a relevant item from the context submenu initialises searching for fragments which can be interesting for user. If any fragment is identified, it is highlighted and a suggested labelling is offered (the identified fragment is highlighted using the colour corresponding to the suggested category). This is illustrated in Fig. 3. Additionally, the context submenu activates two items for user positive or negative feedback. Both these items influence the employed learning mechanism.

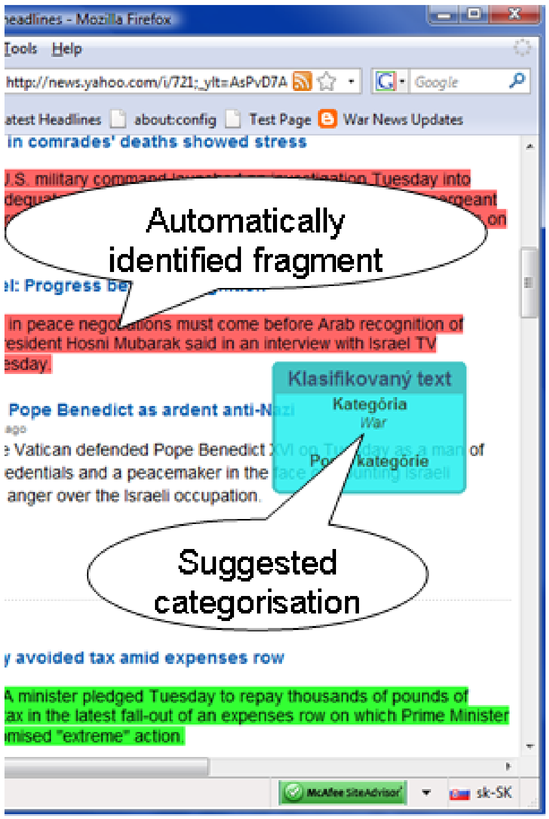

Figure 3: Automatic fragment identification and labelling suggestion

A set of windows serves for system maintenance. For example, it is possible to define, modify or delete user defined categories. Existing bookmarks can be browsed with the possibility to delete or modify fragment labelling as well as additional data fields.

2.3 Learning mechanism

Due to the implementation of the plug-in (Javascript running within browser), only a simple version of the learning mechanism was incorporated. This mechanism is represented by the keyword database, defined categories and a classifier.

Categories are defined by user. Each category can be active or not. Although user can assign any category to classify selected fragments, the system uses only active categories to suggest proper classification of automatically found information fragments. The system can classify a text fragment into one of active categories only.

The keyword database stores keywords characterising available categories. Each category is represented by a list of keywords and keyword frequencies within this category. The length of these lists is limited by a constant.

Learning takes place in two cases: adding a new bookmark and obtaining a negative feedback from user. In both cases a fragment is selected and highlighted – regardless it was defined by user or by plug-in as a result of automatic fragment identification.

Bookmarking a new fragment results in updating the keyword database with keywords contained in the fragment. But before this update takes place, the text constituting the fragment is pre-processed. This pre-processing consists of several steps:

- based on a regular expression, all characters except numbers and letters are replaced with space,
- several consecutive spaces are replaced by one space only,
- text is segmented into terms,
- all uppercase letters are converted to lowercase letters,
- terms are processed by the Porter stemmer [1].

Finally, each term is added to the categories the fragment is assigned to. Adding a term to a category can be twofold. If the category already contains the term, then only frequency of this term within the given category is incremented. If the category does not contain the term, then the term is added to the category's keyword list.

Similarly, negative feedback from user results in unlearning the highlighted fragment. The keyword list of the category which was suggested for the fragment is updated in a negative way – frequencies of fragment's terms are decremented (only if they are present in the category keyword list).

The keyword database serves for a classifier to identify relevant fragments in an automatic way. First, text nodes should be obtained from a retrieved web page. Subsequently, each text node, the length of which is higher than a constant, is processed by the classifier.

The role of a classifier is played by the Naive Bayes classifier [2]. It tries to classify a text node into active categories taking relevant keywords and their frequencies into account. A classification score is calculated for each category. After selecting a winning category, the classification score of this category is compared to a threshold. If the score is above the threshold, then the text node is considered as a newly identified information fragment and the winning category is suggested for labelling this fragment.

2.4 Other plug-in features

When bookmarking of a new fragment takes place, it is possible to add not only categories the fragment should be classified into but some additional data as well:

- fragment commentary,
- todo task,
- todo date.

These data items are displayed along with highlighting the given text fragment within a displayed web page.

Selection of an appropriate item from context submenu results in retrieving and displaying all web pages which contain bookmarked fragments having todo date set to current date. Each such page is open in a separate browser tab.

3 Experiments

The main aim of the presented experiments is to illustrate how a particular parameter setting can influence plug-in behaviour as well as how the plug-in conforms with simulated behaviour of users.

In order to perform experiments, the well known Reuters-21578 collection [3] of English documents was used. Two test document classes were formed – class C_1 representing documents related to trade and business (supposing that a simulated user is interested in this topic) and class C_0 representing "other" documents (the documents which are not interesting for the simulated user). Both classes were disjunctive – no one document used within experiments belonged to both classes.

The achieved results of experiments are evaluated using $F1$ coefficient combining precision and recall.

Since the presented plug-in is dedicated for dealing with web pages, text documents from the used classes were transformed into simple web pages using a PHP script. The generated pages contained several text documents while each document was incorporated as a separate paragraph.

3.1 Length of keyword lists

This parameter controls the amount of information used to describe defined categories which can be assigned to bookmarked fragments. A small value is expected to result in unsatisfactory description of the categories and subsequently to worse performance of the automatic fragment identification and labelling. On the other hand, a big value could be favourable from the point of the used classifier but is too demanding due to the used implementation and running environment. Therefore, a balanced value should be determined.

The test focuses on how successfully the plug-in is able to identify relevant fragments and suggest their labelling. Fifty documents from C_1 were used as a training set. The role of a test set was played by fifty other documents from C_1 and fifty documents from C_0. The experiment consisted of several cycles – each cycle used a particular value of the length of keyword lists and was performed in the following steps:

- removing all keyword information from definitions of categories,
- setting a particular list length coefficient,
- performing training phase on the training set,
- carrying out testing phase using the test set.

Achieved results are depicted in Fig. 4. It is obvious, that steady increasing the length does not imply the permanent increase of $F1$ value. After the initial increase a saturation phase occurs, when bigger values of the length do not bring any benefit.

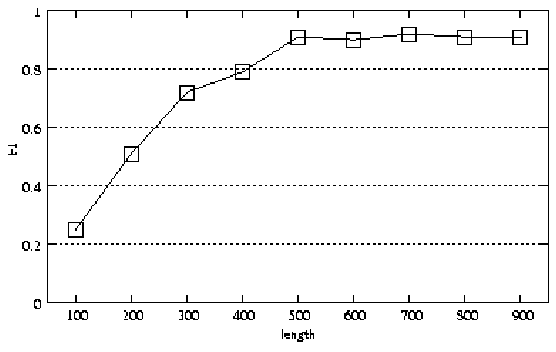

Figure 4: Length of keyword lists experiment

3.2 Number of training fragments

Before the automatic fragment identification can take place, it is necessary to provide the system with a set of bookmarked fragments for the plug-in to establish a satisfactory descriptions of defined categories to be used for labelling. Regarding the number of fragments used to initial category learning, "the more the better" seems to be a good strategy – but those fragments have to be selected and classified manually by user. That is why a balanced value should be found again – to provide as good performance of the plug-in as possible and simultaneously to minimise burden for user.

Also this test focuses on how successfully the plug-in is able to identify relevant fragments and suggest their labelling. Now, the training set consisted of several document subsets. Each subset was composed from five short (150 words maximum) documents retrieved from the C_1 class. Similarly to the previous experiment, the role of a test set was played by thirty other documents from C_1 and thirty documents from C_0.

The experiment consisted of several cycles – each cycle used bigger training set than its predecessor. The first cycle started with two subsets included in the training set. Each subsequent cycle extended the training set with another document subset incrementing the cardinality of the training set by five.

Achieved results are depicted in Fig. 5. The presented plug-in performance is quite satisfactory – the number of training fragments needed to achieve full performance is low enough for user to bookmark them manually.

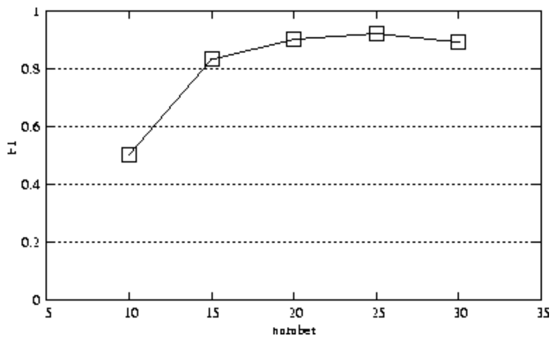

Figure 5: Number of initial training fragments experiment

3.3 User feedback

It is hardly sensible to expect that the initial training of category descriptions will be perfect. Therefore user's feedback plays a very important role – it can help to enrich the information base on which learning is based as well as it can accommodate changes of user interest.

This experiment used the training set consisting of forty documents from C_1 complemented with ten documents from C_0. The role of a test set was played by fifty other documents from C_1 and fifty documents from C_0. One more document set was used – the update set had the same structure as the test set (fifty documents from C_1 and fifty documents from C_0) but contained other documents than the test set.

First, category descriptions were trained by the training set. During the training phase all documents from the training set were bookmarked with the same labelling (bookmarking documents from C_0 as belonging to C_1 simulated a bit unreliable initial learning). Subsequently documents from the test set were used to test plug-in's ability to identify and label new fragments. This testing achieved $F1 = 0.77$.

Next, the update set was used. The system tried to identify and label relevant fragments. Its activity was complemented by user feedback implying learning or unlearning in case of successful or unsuccessful identification and labelling. After the update set was processed, the test using the test set was repeated. Now the plug-in achieved $F1 = 0.85$ – initial unreliable category definitions were updated and performance of the system increased.

4 Conclusions

The implementation and subsequent experiments have shown the feasibility of both presented ideas – finer grained bookmarking as well as active bookmarking. The implemented plug-in is quite useful especially when dealing with more information fragments of a limited length (belonging to different categories) located on the same web page. It can be successfully used as a simple personal tool enabling better usage of the Web as an information source.

5 Acknowledgments

The work presented in the paper was supported by the Slovak Grant Agency of the Ministry of Education and Academy of Science of the Slovak Republic within the 1/0042/10 Project "Methods for identification, annotation, search, access and composition of services using semantic metadata in support of selected process types".

References

[1] Porter, M.F.: An algorithm for suffix stripping, 1980, available at `http://tartarus.org/~martin/PorterStemmer/def.txt`, Accessed: 27th April 2010.

[2] A Naive Bayesian Classifier, available at `http://www.codeproject.com/KB/cs/BayesClassifier.aspx`, Accessed: 27th April 2010.

[3] Test Collections: Reuters-21578, available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, Accessed: 27th April 2010.