FrequentPatternTagCloud Semantic MediaWiki Extension Documentation

Tobias Beck, Andreas Fay

University of Heidelberg

`tobias.beck@stud.uni-heidelberg.de`,
`fay@stud.uni-heidelberg.de`

August 2011

Abstract

This document introduces a new extension we have developed for Semantic MediaWiki, namely the extension, FrequentPatternTagCloud. The extension provides a tag cloud for values of any user-specified Semantic MediaWiki property and additionally displays similar values for each value - based on frequent pattern mining. The frequent pattern mining function is also integrated into the native MediaWiki search function and provides users with substantial similar search term suggestions.

Contents

1 Introduction

Semantic MediaWiki is an extension of MediaWiki. It is designed to enrich articles with annotations which provide additional structured information. These annotations describe triple relations (wiki page, property, value). Property values can be of different types. Properties with value type "page" simultaneously provide annotated links between wiki articles.
The semantic annotation of pages opens a wide range of information processing possibilities. FrequentPatternTagCloud is meant to be one of these possibilities:
It provides users with a tag cloud that not only visualizes values of a specified property but also shows relations between them based on frequent pattern rules. FrequentPatternTagCloud also provides users with an extra layer that shows these similarities while searching.

This document is organized as follows: Section 2 briefly outlines the basic architecture of FrequentPatternTagCloud. Section 3 explains the main components of FrequentPatternTagCloud. Section 4 presents some conclusions and proposals for future work.

2 Architecture

The architecture of FrequentPatternTagCloud basically follows the MediaWiki conventions for extensions, namely separation of setup instructions (*FreqPatternTagCloud.php*), execution code (*FreqPatternTagCloud.body.php*) and internationalization information (*FreqPatternTagCloud.i18n.php*).
We applied the priciples of object oriented programming and applied the following modules:

- *TagCloud*
  The tag cloud module (*TagCloud.php*) computes the tags for a tag cloud. It takes a Semantic MediaWiki property name as input and simply returns all values used for this property as a list.
  The display of the tag cloud can however be organized completely independent of this module. Thus, the appearance can be customized according to individual needs.

- *FrequentPattern*
  The frequent pattern module (*FrequentPattern.php*) is the main module of the extension. It is used to compute all rules based on the Apriori algorithm (see Subsection 3.3 and Subsection 3.4) that represent search suggestions in the context of Semantic MediaWiki properties.
  Again, the display of the gathered information is omitted and is to be done manually.

- *Search*
  The search module (*Search.php*) tests the existence of a Semantic MediaWiki property.

- *Proposal*
  The proposal module (*Proposal.php*) searches for similar Semantic MediaWiki properties. It is used to propose users certain related properties if a used Semantic MediaWiki property is not available.

- *Maintenance*
  The maintenance module (*FreqPatternTagCloudMaintenance.php*) is designed to regularly recompute the frequent pattern rules using the *FrequentPattern* module. We recommend using a cronjob for this task.

The stylesheets for display and javascripts are moved to separate files.

The first two modules (*TagCloud* & *FrequentPattern*) will be explained in more detail in Section 3.1 and Section 3.3 respectively.

3 Components

This section explains the basic components FrequentPatternTagCloud is divided into.

3.1 Tag Cloud

The tag cloud component uses the tag cloud module (see Section 2) as a basis. This means it can compute tags for a tag cloud for a given Semantic MediaWiki property. The tags represent values of the property used in the wiki.
Furthermore the component includes the display of the tag cloud using a special page. It therefore takes the output of the module, i.e. a set of tags with their respective importance, and computes actual font sizes using configurable font size constraints. An example: The minimum font size constraint is 20, the maximum is 70 pixels; a tag with an importance of 0.5 will end up with a font size of 45 pixels.
All tags are displayed in ascending order as a sequence while a tag itself is a hyperlink referring to the Semantic MediaWiki special page *SearchByProperty* that displays all pages in the wiki containing the given property-value combination. When using the right mouse button on a tag a context menu shows up that allows users to browse similar tags following the frequent pattern rules computed by the frequent pattern algorithm (see Subsection 3.3).

3.2 Search Suggestion

The search suggestion component is a ajax extension for Semantic MediaWiki that provides it with an extra layer displaying similar search values. It uses the frequent pattern rules just like the tag cloud but in contrast to the tag cloud, it can also handle multiple Semantic MediaWiki properties. This is necessary because users don't supply Semantic MediaWiki properties but only a search term. The search suggestion component uses this search term to retrieve possible values of Semantic MediaWiki properties as well as the corresponding property itself. It displays similar values for each Semantic MediaWiki property just like the tag cloud does (in its context menu) for one single property.

3.3 Frequent Pattern Algorithm

Frequent Pattern Mining [1, p. 243 ff.] is a discipline for gathering information on frequent co-occurrences of items. It can be used in a wide variety of applications, e.g. for cross marketing. In our case we use it to determine similar values of certain Semantic MediaWiki properties in a wiki. An example: If *many* wiki pages contain *PHP* and *MySQL* as well as *many* pages contain only *PHP* as a value of the Semantic MediaWiki property *Software Prerequisite*, we want to identify the implication *MySQL* $\Rightarrow$ *PHP* but not the implication *PHP* $\Rightarrow$ *MySQL*. One major difficulty in this context is to determine whether a co-occurrence can be considered frequent, i.e. it exceeds a certain threshold of needed occurrences.
Frequent Pattern Mining differentiates between two measures:

- *Support*
  The support s measures the number of occurrences of two different items (or sets of items) X and Y divided by the total number of pages:

$$s(X \Rightarrow Y) = \frac{|\{p \in pages \mid X \cup Y \in p\}|}{|pages|}$$

- *Confidence*
  The confidence c measures the number of occurrences of two different items (or sets of items) X and Y divided by the number of occurrences containing only X:

$$c(X \Rightarrow Y) = \frac{|\{p \in pages \mid X \cup Y \in p\}|}{|\{p \in pages \mid X \in p\}|}$$

Regarding the example above, the support of both implication might be the same but the confidence is not and the latter implication might therefore be discarded.

The goal of a frequent pattern algorithm is to identify all possible implications where support and confidence exceed certain thresholds. It can be divided into two steps:

1. Compute all frequent itemsets
2. Generate implications out of those frequent itemsets

For the first step we use the Apriori algorithm shown in Subsection 3.4. For the second step we simply compute each subset A of a frequent itemset X and check if a generate implication $A \Rightarrow (X - A)$ fulfills the needed minimum support.

3.4 Apriori Algorithm

The Apriori algorihm uses the monotony of frequent itemsets, i.e. each possible subset of a frequent itemset is frequent itself. With this, one can omit all itemsets that are built using a non-frequent itemset.
The method is as follows:

1. Determine all k-element frequent itemsets, starting with $k = 1$.
2. Combine those itemsets to $k + 1$-element frequent itemsets until there are no more new frequent itemsets.

3.5 Framework jQuery & jQueryUI

jQuery [2] and jQueryUI [3] are (third party) free javascript frameworks providing several very useful classes and objects making javascript development easier.
The basic features are:

- Independence through ajax and cross-browser support
- Enabling DOM element selection and traversal via CSS selection syntax
- Enabling effects through an event-system
- Extensibility
- Manipulation of stylesheet
- Availability of components which facilitate user interactions
- Supply of utilities and widgets

4 Conclusions & Future Work

With FrequentPatternTagCloud, we provided Semantic MediaWiki with a powerful extension that enables users to search more efficiently. All modules we have written strongly follow the standards of object oriented programming. The high degree of encapsulation and modularity allows developers not only to use the whole extension at once but also single modules of it. Furthermore, in case of a wiki administrator, it provides several possibilities for configuration, including visualization and rule computation. Some features even can be turned on or off.
One major drawback of FrequentPatternTagCloud is the fact that an administrator has to configure the parameters for minimum support and confidence himself. One future work therefore could focus on this issue and might try to develop a way to automatically adjust the values while ensuring high quality results for frequent pattern rules. Another problem with FrequentPatternTagCloud is the maintenance. The frequent pattern rules have to be computed regularly in order to stay up to date with the evolving wiki. A system administrator has to manually trigger recomputation or establish a cronjob to do so. But it would be a gain if the extension itself handled the recomputation using its own logic and triggers.

References

[1] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011. ISBN 978-0123814791.

[2] The jQuery Project. jQuery: The Write Less, Do More, JavaScript Library, August 2011. URL `http://jquery.com/`.

[3] The jQuery Project and jQuery UI Team. jQuery UI, August 2011. URL `http://jqueryui.com/`.