LEARNING FROM 'TAG CLOUDS'
A Novel Approach to Build Datasets for Memory-based Reasoning Classification of Relevant Blog Articles

Ahmad Ammari and Valentina Zharkova
School of Computing, Informatics and Media, University of Bradford, U.K.

Keywords: Blogs, Memory-based reasoning, Singular value decomposition, Tag clouds.

Abstract: The advent of the Social Web has created massive online media through turning the former information consumers to present information producers. The best example is the blogosphere. Blog websites are a collection of articles written by millions of blog writers to millions of blog readers. Blogging has become a very popular means for Web 2.0 users to communicate, express, share, collaborate, and debate through their blog posts. However, as a consequence to the very massive number of blogs as well as the so diverse topics of blog posts available on the Web, most blog search engines encounter the serious challenge of finding the blog articles that are truly relevant to the certain topic that blog readers may look for. To help handling this problem, an intelligent approach to blog post search that takes advantage from the concept of 'tag clouds' and leverages many open source libraries, has been proposed. A Memory-Based Reasoning model has been built using SAS Enterprise Miner to assess the approach effectiveness. Results are very encouraging as retrieval precision has indicated a significant improvement in retrieving relevant posts to the user compared with traditional means of blog post retrieval.

1 INTRODUCTION

Blogs are one of web 2.0 outcomes. Over the past few years, blog websites, collectively known as the 'blogosphere,' have evolved from simple personal web pages to professional sources for various types of information, including politics, economics, sports, technology, engineering, statistics and every other field of human activity. Organizations have created a rich diversity of informational blogs which many see as a means to bypass the traditional media, such as radio, television and newspapers (Depken et al, 2008). Blog writers, or 'bloggers', can easily create, manage, and maintain posts in their blogs at zero to very low cost by using free software tools. Access to blogs is generally free to all web users, allowing them to read blog posts created by bloggers and write their comments and reviews, adding an extra wealth of information sharing and knowledge acquisition opportunities.

1.1 The Need for an Efficient Blog Search

There are currently many blog search engines that enable users to search for blog posts of interest either directly at the search engine site, or through Abstract Programming Interfaces (APIs) that can be easily integrated into any web application. However, recent studies suggest that blog search has not yet reached its full potential and more could be done to achieve finding blog articles to read that match a desired topic (Hearst et al, 2008). Moreover, most of the blogs are not well-connected because bloggers may write on a variety of subjects (Herring et al, 2005), reducing the efficiency of search for posts relevant to a specific subject. One of the main ways used by blog search engines to solve this problem was '*user-tagging*' or '*social-tagging*' (Millen et al, 2006), which is a feature that enables the use of a keyword or '*tag*' that is explicitly entered by the user for each blog post he reads. These *tags* allow the blog reader to organize and display their blog post collection with tags that are meaningful to them. They also allow the user to search for articles

Ammari A. and Zharkova V.
LEARNING FROM âĂŸTAG CLOUDSâĂŹ - A Novel Approach to Build Datasets for Memory-based Reasoning Classification of Relevant Blog Articles.
DOI: 10.5220/0002884003250331
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

of a certain subject using a tag relevant to this subject that has been associated to these articles by other users who have previously read them. For example, a programmer who wants to read blog posts relevant to programming languages can use the '*programming'* tag to retrieve the posts that have been associated with this tag by previous readers. User-tagging has gained wide popularity and has been even used by researchers to learn user profiles (Michlmayr et al, 2007) and improve social navigation (Millen et al, 2006). Although user-tagging may seem a good solution, a simple experiment reveals that it is still not enough to obtain an efficient collection of relevant posts. Out of 248 blog posts that have been retrieved by the Technorati Search Engine API (Technorati Blog Directory) using tags directly related to programming languages, such as '*java servlet JSP*', and '*perl python PHP Ruby*', only 92 posts (37%) could be classified as relevant to programming languages after inspection. This was even less than the relevance rate obtained (52%) when we used the query search instead of the tag search.

1.2 Contributions

This work is an extension to our work in (Zharkova et al, 2009). We aim to add '*intelligence*' to blog search, so that blog readers can find their desired articles efficiently. To achieve this, we propose a web mining '*predictive searcher*' that first retrieves a collection of blog posts using a blog search engine API, then '*scores*' each post in this collection, returning only the relevant subset of posts that will most likely satisfy the user's search needs. Contributions toward creating this intelligent searcher could be summarized as follows:

1. The development of a Java-based text analysis framework based on (Apache Lucene) to prepare the data set required to train the proposed searcher.
2. A novel approach based on the 'tag-cloud' concept (Alag, 2009) to classify the prepared training data set.
3. The use of memory-based reasoning and singular value decomposition implementations in the (SAS Enterprise Miner) software to build the predictive blog searcher.

The rest of this paper is organized as follows: The novel approach to prepare and classify the learning dataset used in building the intelligent blog searcher is described in Section 2. The algorithms used to build the predictive blog searcher are described in Section 3. Experimentations and results are discussed in Section 4. Finally, conclusions are drawn in Section 5.

2 LEARNING FROM TAG CLOUDS

2.1 Overview of 'Tag Clouds'

Tag clouds are visual representations of social tags, displayed in a 'cloud-like' layout. Each tag in the cloud appears in a relatively different size than the size of the other tags in the cloud. Figure 1 shows a sample tag cloud appeared at Flickr.com. The creator of the tags in a tag cloud can be one of three: First, a domain expert who has a good knowledge about the domain of the website where the tag cloud should appear. Tags created in this way are called '*professionally-generated*' tags. Second, a user who is allowed to tag items in the website with keywords. These tags are called '*user-generated*' tags. Finally, tags can be created by a text analysis program that analyzes the textual content of the website and generates the tags. These are called '*machine-generated*' tags (Alag, 2009). Whoever the creator of the tag cloud is, the size of each tag in the cloud always reflects how *important* this tag is in the domain that the cloud represents. Importance is usually defined by either how frequently the tag is used by users (*e.g.* as a search query), or how frequently it occurs in the overall text resources that make the website content. Tags with higher frequency are appeared in the cloud with a larger font than the other tags with less frequency (Hearst et al, 2008).

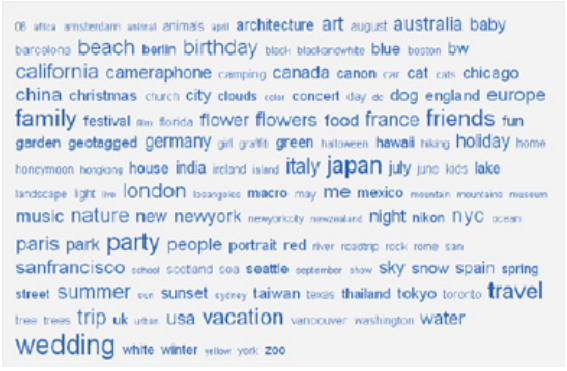


Figure 1: A Tag Cloud from (Flickr.com).

Because of this '*tag size / tag importance*' relationship in the tag cloud, we argue that if we

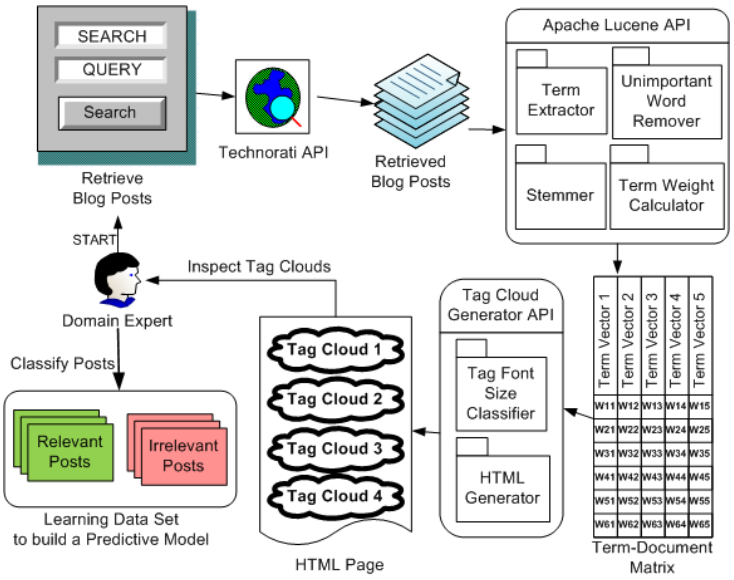


Figure 2: A Framework to create and classify a Learning Dataset.

have a tag cloud that represents the content of a specific document within a document collection, and we inspect a considerable number of relatively large tags that are strongly relevant to a specific topic of interest, we can assume that the document represented by this tag cloud is strongly relevant to that topic. For example, it is clear that the tags in the cloud shown in Figure. 1 are strongly relevant to holidays travel and tourism. Therefore, if this cloud happens to represent an article in a news or blog website, it is safe to assume that this article is strongly relevant to the topic of holidays travel and tourism.

2.2 The Dataset Creation Framework

Building a data set from a textual collection of blog articles to train a classifier normally consists of three main parts:

1. Retrieving a collection of blog posts using a blog search engine API.
2. Analyzing the unstructured content of the collection, which is a special data transformation technique in which each article is converted to a *bag-of-words* model (Cios et al, 2007) by extracting the important terms, synonyms, and phrases from the article, stemming the terms to their original linguistic roots, and computing a numerical *weight* to each stemmed term. The result is a *term weight vector* representation for each article, and the transformed data set is usually called a *term-document matrix*.
3. Classifying each article in the collection to either *relevant* or *irrelevant* to the topic of interest that triggered the search. Because classification is a supervised learning process (Hornick et al, 2007), we need to add a column, called the *target attribute*, to the term-document matrix to store a value that represents the relevance of the article to the topic of interest. We can use a binary data type {1 ≡ relevant, 0 ≡ irrelevant} for the target attribute.

For parts 1 and 2, we customize a Java-based framework that leverages a couple of open – source APIs:

- A Technorati Search API is used to retrieve a collection of blog posts containing specified keywords of interest that are input by a domain expert.
- An Apache Lucene API is used to transform the retrieved collection of blog posts into a term-document matrix. Each vector in the matrix represents a blog post and the elements of this vector are the numerical weights of the extracted terms of the whole blog collection with respect to the article that this vector represents.

For part 3, a tag cloud generator API (Alag, 2009) is used to transform each term weight vector in the term-document matrix into a tag cloud representation. Then, all the resulted tag clouds are visualized in a regular web page. In this way, the domain expert who triggered the search can easily

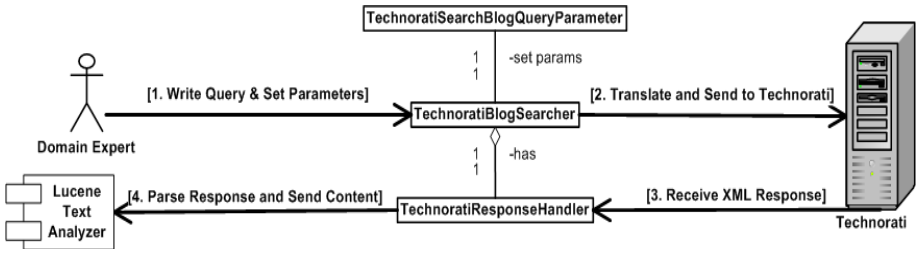


Figure 3: The process of searching the blogosphere with the Technorati API.

and quickly inspect each tag cloud and then accurately determine whether the blog post that the inspected cloud represents is relevant or irrelevant to the query that he used to trigger the search. The whole process of building the learning data set is described in Figure 2.

2.3 Searching the Blogosphere for Relevant Posts

Technorati is a popular blog search engine that provides an API that can be used by web developers to integrate many blog-related activities into their applications. Using the API provided by Technorati, it is possible to search for blog posts using three different methods: using keywords that are found in the content, using tags that are associated to posts by users, or search for posts that are linked to a particular URL. Moreover, the API allows developers to retrieve a list of the most popular N tags, get detailed information about a blog, and even view information about a Technorati member.

Searching the blogosphere using the Technorati API involves four main steps:

1. Writing a query string and setting the search parameters: The search parameters involve selecting the preferred search method, the output format, the search language, and the number of posts to be returned. Two Java classes are responsible for processing this step: *TechnoratiBlogSearcher*, and *TechnoratiSearchBlogQueryParamete*.
2. Translating the query string to Technorati-understandable format and send it to Technorati search engine: The *TechnoratiBlogSearcher* class processes this step.
3. Receiving the XML response containing the search result. The *TechnoratiResponseHandler* class processes this step.
4. Extracting the important content of the retrieved blog posts, such as the title, name, and excerpt of each retrieved blog post, merging the extracted content, and sending it to the text analyzer. The *TechnoratiResponseHandler* class processes this step.

Figure 3 illustrates the four steps of the searching process.

2.4 Analyzing the Retrieved Posts

Our text analyzer is built over Apache Lucene. Lucene is an open-source Java-based search engine. It provides a number of classes that can be used to build a customized text analysis tool. In order to transform the retrieved blog posts into a term-document matrix, our text analyzer performs the following four main activities:

- *Term extraction*: the unstructured content is parsed to generate individual terms. Each term may be a single word, a small phrase, or a synonym. Lists of considerable phrases and synonyms are fed into the analyzer so it can detect them during the parsing process.
- *Unimportant word removal*: terms irrelevant with respect to the main subject of the search are removed (Cios et al, 2007).
- *Stemming*: reducing the remaining terms to their root form by removing common prefixes and suffixes (Cios et al, 2007). For example, the words *programming*, *programmers*, and *programs* share the root *program*, and so can be treated as different occurrences of the same term.
- *Term weight calculation*: a numerical *weight* is computed for each term in each blog post, generating the term weight vector that represents the post. This term weighting takes into account the frequency of appearance of the term in the post, as well as the total frequency of appearance of this term in the whole retrieved posts.

The popular Term Frequency - Inverse Document Frequency (*tf-idf*) weighting scheme (Kobayashi et al, 2008) is used to calculate the term weight ω_{xy} as follows:

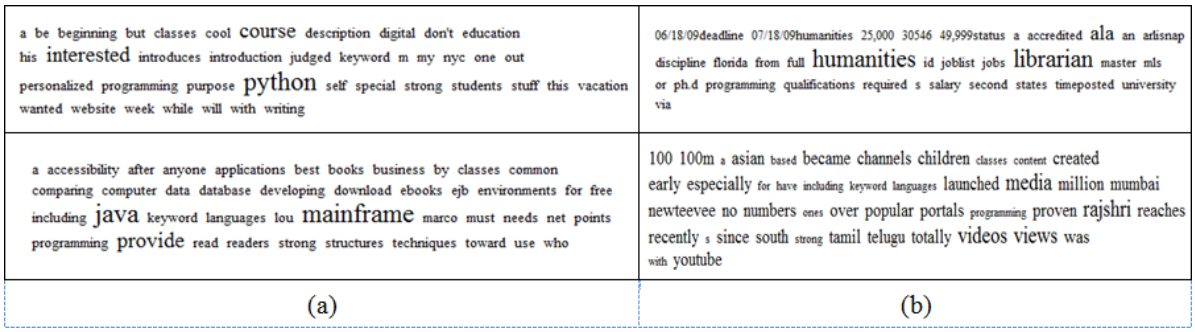


Figure 4: Tag Clouds for Blog Posts classified as (a) relevant, and (b) irrelevant to '*programming languages*'.

$$\omega_{xy} = \mathcal{F}_{xy} \times \log\frac{\mathrm{P}}{fr_x}$$

$\mathcal{F}_{xy}$ is the *normalized* frequency of term *x* in the blog post *y*, P is the total number of retrieved blog posts, and fr_x is the frequency of term *x* in the whole retrieved posts.

Attributes are often normalized to lie in a fixed range (*e.g.* zero to one). Term frequencies are normalized using the equation below:

$$\mathcal{F}_{xy} = \frac{fr_{xy}}{\sqrt{\sum(fr_y)^2}}$$

where fr_{xy} is the frequency of term *x* in the blog post *y* and fr_y is the frequency of each term in the blog post *y*.

2.5 Building the Tag Cloud Representation

As previously mentioned, to improve the accuracy and speed of the classification of the training blog data set by the domain expert, a tag cloud generator API is implemented to create a tag cloud from each term weight vector in the term-document matrix. The first component of our tag cloud generator API is a tag font-size classifier that divides the available term weights into a number of ranges of values and then allocates a specific font size to each term weight range. The larger the values in the term weight range, the larger the allocated font size is. Therefore, each term in the cloud appears in a size that reflects its relative importance in the blog post.

The second component of the API is an HTML generator that allows the domain expert to visualize all the generated tag clouds, separated by a blog post identifier, in a regular web page. After classification, each vector in the matrix will contain an additional element, the target attribute element, that stores a binary value {1, 0} indicating the relevance or irrelevance of the associated blog post to the topic of interest.

3 BUILDING THE SEARCHER

3.1 Memory-based Reasoning Classification

Memory-based reasoning (MBR) consists of variations on the nearest neighbor techniques. A description of MBR is found in (Rachlin et al, 1994). It has been used in previous research projects to classify text documents, such as news stories, with good results obtained (Masand et al, 1992). MBR solves a new task based on its computed similarity with other tasks found in previous remembered examples. In this context, MBR can predict whether a new unseen blog post is relevant or not by finding near matches from the learning dataset of posts and then choosing the best relevance value based a confidence threshold.

3.2 Performance Improvement with SVD

As discussed in section 2, analyzing the blog post collection generates the term-document matrix. For a collection of text documents such as blog posts, the generated matrix can contain hundreds of thousands of words. Such very high dimensional data requires too much computing time and space to analyze and model. To improve the searcher performance, singular value decomposition (SVD) is a popular feature extraction technique (Cios et al, 2007) that can be implemented to reduce the dimensions of the term-document matrix by transforming the matrix into a lower dimensional and more compact matrix.

4 EXPERIMENTAL RESULTS

An experiment has been done to address the evaluation of the proposed tag cloud-based approach to the learning dataset classification. Two blog post collections have been retrieved by the Technorati API and then their textual contents have been analyzed programmatically using the Apache Lucene API to create a learning dataset from each collection. The first collection has been retrieved using query strings directly relevant to *programming languages*, such as '*classes objects*', '*perl python php*', and '*polymorphism inheritance*'. Then, a tag cloud for each retrieved post has been generated and inspected by a programmer to classify its relevance to programming languages. Figure 4 shows four of the 248 generated tag clouds for the retrieved posts. The first two clouds (a) have been classified as *relevant*, whereas the other two (b) as *irrelevant*. 129 out of a total of 248 retrieved posts (52%) have been classified as being relevant to the target subject (programming languages).

The second collection contains 584 posts that have been retrieved using user-tagging. The first half of the posts has been retrieved using user-tags directly relevant to programming languages and so has been automatically classified as being relevant to this topic, whereas the other half has been retrieved using irrelevant user-tags, such as "*Football*", "*Swine flu*" and "*Pop music*" and similarly has been automatically classified as being irrelevant to programming languages.

Next, SAS Enterprise Miner has been used to build a predictive model based on each built dataset described above. First, two supervised data sources have been created based on the two created datasets. Second, each dataset has been partitioned to 60% for training, 20% for validation, and 20% for testing. Third, SVD with maximum of 50 features (attributes) has been applied to reduce the dimensionality of each generated term-document matrix before building the predictive model. Finally, The MBR classification node has then been chosen to build the predictive searcher.

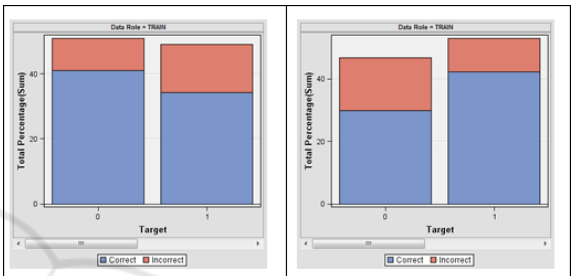
Table 1 shows a comparison of five performance measures between the training parts of the two data sets. Table 2 shows this comparison between the validation parts. Compared to the user-tagging approach, it is clear that the tag cloud inspection of a learning dataset leads to a good classification quality to this dataset, and that the argument proposed in section 2.1 is valid when applied on blog post retrieval.

Table 1: Performance of Searchers on Training Data.

Training Dataset Applied	Tag Cloud – based	User Tagging – based
Average Squared Error	0.17	0.20
Root Average Squared Error	0.41	0.44
Sum of Squared Errors	47.8	138.3
Root Mean Squared Error	0.47	0.47
Misclassification Rate	0.26	0.28

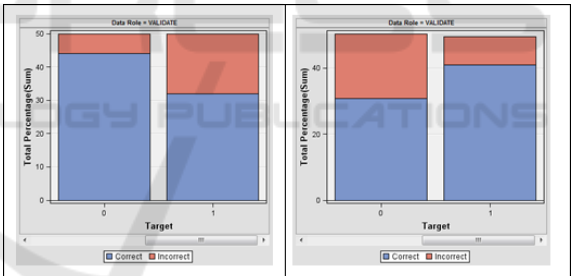
Table 2: Performance of Searchers on Validation Data.

Validation Dataset Applied	Tag Cloud – based	User Tagging – based
Average Squared Error	0.18	0.20
Root Average Squared Error	0.42	0.44
Sum of Squared Errors	17.6	45.7
Root Mean Squared Error	0.42	0.44
Misclassification Rate	0.24	0.28



(a) Tag Cloud - based (b) User Tagging - based

Figure 5: Prediction Percentage on Training Data.



(a) Tag Cloud - based (b) User Tagging - based

Figure 6: Prediction Percentage on Validation Data.

Another two popular measures to assess information retrieval systems are *precision* and *recall* (Cios et al, 2007). Precision measures the rate of *truly* relevant posts from the total posts that the model has *predicted* to be relevant.

Recall measures the ability of the model to retrieve a good rate of relevant posts from the total original relevant posts available. To compare between the precision and recall of the two searchers, we study figure 5 that shows the prediction percentages of the two searchers on the training datasets, and figure 6 that shows these percentages on the validation datasets.

Both figures (5 and 6) clearly show that the predictive searcher (a) that has been trained by the

tag cloud-based dataset is more able than the user tagging-based searcher (b) to correctly classify those posts irrelevant (Target = 0) to the topic of interest. Consequently, it is more likely (*higher precision*) for a post collection retrieved by the searcher (a) to contain more relevant posts than a post collection retrieved by the searcher (b), if the two searchers retrieve the same number of posts.

However, the searcher (b) is more able than the searcher (a) to correctly classify those posts relevant (Target = 1) to the topic of interest, which means that it is more likely (*higher recall*) to read more posts from the *totally available* relevant posts if we use the user tagging-based searcher.

5 CONCLUSIONS

In spite of the many available blog search engines on the web, a little attention has been paid on how much the blog posts that these search engines retrieve are truly relevant to what the users look for. There is a crucial need to add intelligence to the searching mechanisms of blog posts in order to improve their precision and recall results. This work has proposed an efficient framework based on open source APIs, as well as on tag cloud inspection, in order to retrieve, analyze, and classify a collection of blog posts used to train and build an intelligent predictive searcher for filtering the results of search engines, thus improving the relevance rate of the posts returned to the user.

In comparison with another popular approach to blog search improvement, user-tagging, results show that relying on tag cloud inspection to classify a collection of blog posts for training a predictive blog searcher is a good decision to take. Moreover, it is recommended by this work to apply the tag cloud-based dataset learning approach for building blog post classification models when precision in the returned results of the model is more important for the application domain than recall.

REFERENCES

Alag S., 2009. Collective Intelligence in Action. Manning Publications, Greenwich

Apache Lucene, http://lucene.apache.org/java/docs/

Cios, K. J., Pedrycz, W., Swiniarski, R. W., Kurgan, L. A., 2007. Data Mining, a Knowledge Discovery Approach. Springer, New York

Depken II, Craig A., 2008. “Benford, Zipf and the blogosphere.”, Applied Economics Letters, 15:9, 689 – 692

Hearst M., Hurst M., Dumais S., 2008. “What should blog search look like?”, In: Proceedings of the 2008 ACM workshop on Search in social media, pp. 95 – 98, California, USA

Hearst M., Rosner D., 2008. “Tag Clouds: Data Analysis Tool or Social Signaller?”, In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, p.160

Herring S., Kouper I., Paolillo J., Scheidt L., Tyworth M., Welsch P., Wright E., Yu N., 2005. “Conversations in the Blogosphere: An Analysis from the Bottom Up.”, In: Proceedings of the 38th Hawaii International Conference on System Sciences HICSS’05

Hornick, M. F., Marcadé, E., Venkayala, S., 2007. Java Data Mining: Strategy, Standard, and Practice. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, San Francisco

Kobayashi M., Aono M., 2008. “Vector Space Models for Search and Cluster Mining.”, In: Survey of Text Mining II, Springer-Verlag, London

Masand B., Linoff G., Waltz D., 1992. “Classifying news stories using memory based reasoning.”, In: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, p.59-65, Denmark

Michlmayr E., Cayzer S., 2007. “Learning user profiles from tagging data and leveraging them for personalized information access.”, In: Proceedings of the Workshop on Tagging and Metadata for Social Information Organization, 16th International World Wide Web Conference

Millen D., Feinberg J., 2006. “Using social tagging to improve social navigation.”, In: AH2006 workshop, Social navigation and community-based adaptation, Dublin, Ireland

Rachlin J., Kasif S., Aha D., 1994. “Toward a better understanding of memory-based reasoning systems.” In: Proceedings of the Eleventh International Machine Learning Conference. Morgan Kaufmann. 242–250

SAS Enterprise Miner: a Data Mining Software, http://www.sas.com/technologies/analytics/datamining/miner/

Technorati Blog Directory, http://technorati.com/blogs/directory/

Zharkova V., Ammari A., 2009. “Combining Tag Cloud Learning with SVM classification to achieve Intelligent Search for Relevant Blog Articles”, In: Proceedings of the 1st International Workshop on Mining Social Media, paper#7, Sevilla, Spain