Tag Suggesting for Digg

Bruno Oliveira
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa, Portugal
bruno.santos.oliveira@ist.utl.pt

ABSTRACT

Social tagging systems are becoming increasingly popular, mainly because of their ease of use and low entry barriers. The subject is fairly new, and although there is not much literature on the topic, there are quite a few systems available. Despite its popularity, tagging is usually a manual process. When publishing on a social tagging system, the user is asked for the tags he wishes to assign to the resource being made available. In this paper we present a text-based tag suggester. Such a system makes the task of tagging resources easier by recommending tags for the user to choose from. The system was evaluated by a group of users and some statistical measures were applied to infer its performance. The preliminary results looked promising. Nonetheless, there is still room for improvement. Hence, some ideas on how to further develop the system are suggested.

General Terms

Algorithms, Experimentation

Keywords

Tagging, Tag Suggesting Systems, *Digg*, Folksonomies, Information Management Systems

1. INTRODUCTION

The shape of the World Wide Web has been changing, as websites tend to be more interactive and user-centric than in the past. Some of the most successful systems on this "Web 2.0" generation are the social tagging ones [15]. These systems have their content managed by the users, who publish resources along with sets of tags. Tags are simply keywords to help organize and describe each resource. Since there is no predefined vocabulary or semantics, the user can choose whichever string he prefers to use as a tag. Typically there is no way for the user to establish relations between tags or resources, which means they will not form hierarchies. Resources vary from system to system. For instance, *flickr* [1] is a website for photos sharing, *YouTube.com* [2] for videos, *del.icio.us* [3] for bookmarks and *WordPress.com* [4] for *blog* entries.

Social tagging systems share a common set of features: resource publishing, browsing and searching. To take full advantage of the system, a user must create an account, providing a username and a password. The registered user can publish any resource he wishes, as long as the system requirements are fulfilled. Typically, the user uploads the resource and associates a set of tags to be attached to it, as well as any additional information the system may require. Some systems suggest a set of tags for the user to choose from, making his task easier. *BibSonomy* [5] and *del.icio.us*, for example, present the most popular tags associated to a resource when it was already tagged by other users and is being resubmitted.

Typically there are two ways of browsing: through tags or through users. Browsing through tags enables the user to find the resources that were assigned a certain tag. Since every resource has its owner, there is also the ability to browse through users, that is, to find all resources and tags published by a certain user. Generally users combine both browsing modes. Moreover, users can find resources using the social tagging system's search engine.

Along with the features presented so far, some websites display other interesting features as well. Tag cloud is one of such popular features. It consists of a visual representation of the most frequently used tags on a website. Usually the tags are presented in alphabetic order and are given different emphasis according to their popularity. Another interesting feature found in some social tagging systems is the creation of groups. Generally, groups are organized in categories and the resources shared among the group are all on the same thematics. A somewhat less common characteristic is the distinction between users. Some websites allow users to choose among different account types, each one with their own set of features.

One may ask what made tagging so popular and interesting. First of all, it is very easy. The concept is simple and requires no special skills. Also, it gives users a sense of freedom and power, as there are no restrictions to the resources organization and description. Another appealing feature is the ability to share resources. Furthermore, these systems are constantly evolving and reflect the trends going on in society. When dealing with non-textual resources, such as photographs or videos, tagging is a rather straightforward technique to help improve search and information retrieval.

Tagging has its drawbacks as well, mostly due to the lack of structure and unrestricted vocabulary. Fig. 1, a real example taken from *del.icio.us*, will be be used to illustrate some of these drawbacks. One problem is polysemy, which consists on one word having multiple meanings. For example, the word 'ruby', might refer to a computer programming

[1] http://www.flickr.com/
[2] http://www.youtube.com/
[3] http://del.icio.us/
[4] http://wordpress.com/
[5] http://www.bibsonomy.org/

language or to a red gemstone. Another is synonymy, which occurs when different words have the same meaning. For instance the words 'fun' and 'humor', may have the same meaning. A third problem arises when users employ some tags to help organize their agendas, instead of describing the resource. The tag 'toread' is probably assigned when the user does not have the opportunity to read the resource, acting more as a reminder. The fourth problem is due to the fact that these systems are likely to be used by people from all over the world, so, probably, more than one idiom will

Figure 1: *del.icio.us* tag cloud with the most popular tags on July 20, 2007

be used simultaneously. All these issues make it difficult to infer the tags semantics in a precise way. Since there aren't relations between tags, it is also difficult to find out which ones have broader or narrower meanings or which ones are related [9].

The use of tags as metadata – data about data – whose semantics emerges from its community is called "folksonomy". Thomas Vander Wal was the first to use this word in a discussion on an information architecture mailing list [10]. It is a combination of the words "folk" and "taxonomy". Folksonomy enthusiasts claim that the semantics emerging from a community is more accurate than when it is defined by a committee or a single person. They also say that, since content creation is decentralized, its description be decentralized as well. Furthermore, they argue that hierarchical classification schemes are too rigid to classify web data [2].

Digg [6] is one of the most popular social networking systems. It was launched in November 2004 as a news sharing website, democratically managed by its user community. Initially, it was mainly focused on technology news [16]. Today, *Digg* has a wider scope, with several distinct categories, such as science and sports and two new sections dedicated to videos and podcasts.

The content of the website is chosen by its community, who acts as the editor board of a newspaper. A registered user can submit a story by providing a title, its link, a brief description of the story and the most appropriate topic, of all the ones available on the website. *Digg* does not include a tagging mechanism. When a new story is submitted, it is placed in the "Upcoming Stories" queue, awaiting for votes. If a story receives enough positive votes – or, using the website terminology, 'diggs' – it gets promoted to the front page. There is also the possibility to vote down – or, 'bury'. Thus, if a story does not get enough 'diggs' or is 'buried' by the users, it disappears from the website. The process of promotion and decay of user submitted stories is run by a proprietary algorithm, kept undisclosed in order to avoid abuse and exploit. It is an automatic technique based on several factors, such as the number of 'diggs', time of day and category [4].

Despite its growing popularity, tagging is still a manual process. Hence, the user is the only responsible for assigning keywords to resources. Some users may may not be willing to engage in such a chore. In order to overcome this shortfall, some websites have a built-in tag suggesting mechanism, which suggests a set of words for the user to choose from and assign to the resources. *del.icio.us* is one of those websites. If the resource being published has already been submitted and tagged by other users, the system gathers and sorts the most popular tags and suggests them to the user. Despite useful in several situations, it is unable to perform when the resource is being published for the first time. Moreover, it does not work in websites where each resource is unique.

In this paper we present a text-based tag suggesting system, based on information retrieval and clustering techniques. The tags suggested by the system are extracted from documents only. Since there is no dependency on tags that were previously employed by users, the system is likely to keep up with the changes and evolution of the communities, their interests and resources. Furthermore, it is able to deal with resources that have never been tagged before, as long as they are text based, which makes it suitable to a wide range of systems. Although *Digg* was used as subject and data source on this research, the tag suggesting system should be adaptable to other websites. *Digg* was used as the data source due to its popularity, variety of subjects and large amount of resources.

The tag suggesting system we present was built from scratch. Several techniques were experimented and combined as an attempt to improve results. To infer the system's performance, a user-based evaluation was carried out. The preliminary results were promising. On average, over 60% of the suggested tags can be considered relevant.

2. RELATED WORK

Most social tagging systems are rather simple and, thus, do not exploit folksonomies to its full potential. In the following subsections we present some of the proposed techniques and group them into categories according to their most relevant features. In general, we find that these techniques can fit into three main categories: resource organization, search engine improvement and tag suggesting.

2.1 Resource Organization

Folksonomies lack of structure and organization makes some user tasks very difficult. Although its uncontrolled nature induces serendipitous discoveries, it becomes hard to find specific resources [10]. To overcome this issue, some authors tried to cluster related resources and place them in broader categories. Additionally, some tried to automatically infer a topical hierarchy, as a means of helping users navigate through its content.

[6] http://www.digg.com/

2.1.1 Grouping Similar Resources

Wu *et al.* present a probabilistic method, based on co-occurrences, to derive emergent semantics on folksonomies [18]. They claim that tags are usually semantically related when used on the same or related resources. The same idea applies to users and resources, that is, users may share common interests if they use many semantically related tags and resources tagged by users with similar interests. This can be represented using frequencies of co-occurrences between users, resources and tags. In the algorithm they propose, each entity (user, tag or resource) is represented in a conceptual space where each dimension represents a knowledge category. The number of knowledge dimensions is defined using a Separable Mixture Model, which is a statistical framework for modeling and predicting co-occurrences of events [6]. According to the authors, the algorithm successfully grouped tags according to their semantics. Moreover, it identified tags ambiguity and synonymy, the users interests and the resources semantics. Despite these favorable results, there are some drawbacks. A lot of experimentation is needed in order to find the appropriate parameters – number of dimensions D and number of iteration steps on the EM method. Also, the whole conceptual space has to be re-calculated whenever the annotation set changes.

A different approach, to serve a somewhat different purpose, is presented by Wu, Zubair and Maly [17]. In their opinion, folksonomies lack some fundamental features to be used as the supporting infrastructure of knowledge management activities, such as the identification of topical communities of users and/or resources and ways to find high quality sources. In order to solve these issues, two algorithms are proposed. The first one aims at identifying communities in social tagging systems, by representing the folksonomy as a graph using an adjacency matrix. The main topics of interest and the most important users on a topic are identified by applying singular value decomposition to the matrix. The second algorithm's goal is to identify high quality documents and experts. For that purpose, they adapted the HITS (Hypertext Induced Topic Selection) [1] algorithm. At the time of writing, the authors claimed they were still tuning the algorithm. Nonetheless, the preliminary results looked promising.

Brooks and Montanez present a study on the effectiveness of tagging as a means to categorize documents [3]. The idea is to determine how useful are tags in clustering similar articles. Intuitively, a group of documents sharing the same tag is more similar than a group of randomly selected articles. In order to find out whether this hypothesis is correct or not, they clustered a set of documents from a social tagging system and then compared them with two bounds. The lower bound was obtained by randomly clustering articles, whereas the upper one was determined by clustering articles reported as being similar by *Google News* [7]. They concluded that the process of tagging, groups articles into broad categories, making it difficult for a user to find documents when the topics are more specific.

2.1.2 Inducing Hierarchies

Wu, Zubair and Maly state that hierarchies are very useful for navigation and as means to assist in keyword-based search [17]. They further claim that it is possible to use tags to induce such structures, using a hierarchical clustering algorithm. In their article they describe an agglomerative clustering approach that uses a document-tag matrix and a tag-tag matrix. The former matrix represents the associations between documents and tags, whereas the latter one is built using a thesaurus and is intended to capture the semantic similarities between tags. The method described can produce different results, depending on the clustering and labeling methods used.

Likewise, Brooks and Montanez present an agglomerative clustering technique as a means to generate hierarchies [3]. Their algorithm starts by collecting tags from a website and then, for each tag, it gathers 20 articles and forms a cluster. Next, the most similar clusters are successively merged and assigned a new tag. According to the authors, the generated hierarchy resembles hand-built taxonomies. Although the cluster merging was based on document similarity, that is, the tags were only used to build the original clusters, tags fit into a topical hierarchy. This remark supports the idea of being possible to automatically generate a category hierarchy.

2.2 Search Engine Improvement

Searching is one of the main features in social tagging systems. It is the only mechanism, other than browsing, that enables users to find specific resources. The information about users, tags, resources and the relations among them, can be used to improve the searching mechanism.

The work proposed by Wu *et al.*, besides presenting a method to group similar resources can also be used to improve search in folksonomies [18]. In this case, the conceptual space is used to develop an intelligent semantic search engine. They followed an incremental approach, starting with a basic search model and improving it gradually, by extending its features. The search models are independent and can be combined in different ways. Thus, distinct search engines can be obtained. The authors claim that the ability to discover semantically related resources is the most important feature of their models.

A ranking mechanism to improve search is also described by Hotho *et al.* [7]. The authors claim that social tagging systems have limited search support, since searching can only be done over the tags and the resource descriptions. Besides, no ranking is done over the retrieved resources. To overcome this issue, they propose two algorithms, both based on a graph-like representation of the folksonomies. The first one is adapted from PageRank [1], whereas the second algorithm consists of an improvement of the first one. The experiments described in the article confirm that the second algorithm obtains the best results. Moreover, when a topical community is small, a single user can have a very strong influence on the results and create bridges to other topics. Furthermore, the algorithm can be used to recommend tags or resources of interest to a given user.

2.3 Tag Suggesting

To the best of our knowledge, only one tag recommending system has been proposed in the literature so far: the *AutoTag* system [11]. Using the most common recommender systems as an analogy, which suggest products to their users, in the *AutoTag* system, blog posts take the role of users and tags are seen as the products the user might be interested in. Similar tags are assumed to be suitable to similar blog posts. For that reason, the system suggests tags for a blog

[7] http://news.google.com/

entry by examining the tags assigned to similar entries. The proposed method consists of five steps. First, the user publishes a new article. Second, the system finds and collects the most similar entries by generating a query from the new article and submitting it to a search engine. Third, the system aggregates all tags assigned to the retrieved entries and ranks them according to their frequency in the top posts. Fourth, the system filters and re-ranks the set of tags by boosting the ones that were previously used by the blogger. Fifth, the system presents the top tags to the user. The experiments described in the article show good results, in spite of the relatively small corpus used. However, there is still room for improvement. Other query generation methods might be used, tag aggregation could be enhanced and different filtering and re-ranking techniques might provide better results.

Although the authors of [3] expressed their willingness to develop a tag suggester as well, they have not released any article about it yet. There was also a webservice called *Tagyu* which, when given an article, suggested a set of tags, but was shut down in August 2006 [8]. Since no publications regarding the *Tagyu* system were found, it was impossible to analyze it.

If one looks at tags as being categories, the automated tagging problem may fit a well-known research area in computer science: text categorization (TC). Therefore, although not focused specifically on folksonomies, we find this area relevant to our problems and worth to be mentioned in this section. A comprehensive survey about TC has been published by Fabrizio Sebastiani [14]. TC consists in determining the categories of natural language texts using a set of predefined categories. A classifier is a program responsible for this labeling activity. Several methodologies can be used to build a classifier. Nowadays, machine learning (ML) techniques are the most popular ones. Basically, they are used to automatically induce a text classifier by learning through a set of preclassified documents. Regarding the classifier algorithms, several methods can be used. Although it is hard to predict which ones would work well in the tagging process, support vector machines, example-based methods, regression methods and boosting-based classifier committees seem to be the most accurate in TC. However, there seem to be some limitations when using this kind of systems as tag suggesters. First of all, they depend on a finite, predefined set of categories (or tags, in this case), which means they will not be able to automatically keep up with the evolution of the users vocabulary. Furthermore, these systems cannot be used with non-textual resources, as they need to analyze their content, which must be text.

3. TAG SELECTION ALGORITHM

In this section we present our tag suggesting system, entitled Tess. It is based on information retrieval and clustering techniques. The data needed by the system was gathered from *Digg*'s website, given its popularity and large amount of resources. Hence, Tess can use the same set of topics and their hierarchical relations.

3.1 System's Architecture

There are three major components in Tess architecture. In the first stage, *Digg*'s top stories are collected, as well as their additional information, such as the number of votes, title, publisher's name, and so on. Next, the gathered documents and information are processed and indexed in order to be used efficiently by the third component. Lastly, the main component does the most noticeable job on this system. Using the data collected and prepared in the previous stages and also the user query – document and its additional information – it recommends several sets of tags, according to Tess configuration. Optionally, the system's categorizer may be used to assign a topic to the document.

3.2 Data Gathering

Before starting to collect data, *Digg* was analyzed to find out which stories should be retrieved. Besides the Videos and Podcasts sections, which were excluded from the start due to the non-textual nature of its resources, the 'Offbeat News' topic was ignored as well, since there is not a definite subject among such stories. Another important requirement, defined in advance, was to retrieve top stories only. To get to the front page, a story needs to be voted by many users. For that reason, the stories are more likely to be of interest and were probably filed under the appropriate topics. That is, selecting top stories only, ensures a higher degree of quality.

Besides the article itself, each *Digg* entry has additional information, provided by the user when publishing the resource. The following data was found to be relevant to our system: title, description, URL, topic, subtopic, user name and number of votes. Thus, it was stored along with the story document.

Digg has a hierarchy of topics, where each topic has its own front page, that is, a place where popular stories are placed. These sections of the website work as stacks, that is, each time a story is promoted, it is placed on top of its topic's front page. As newer entries are added to the top of the page, older ones go down and, when they reach the bottom, they move to the next page.

When our crawler was written, there was no API available to retrieve data from *Digg*. For this reason, the program was designed to fit *Digg*'s structure and getting the information needed by using regular expressions to parse the downloaded pages. However, any change in the website's HTML templates or URL hierarchy may jeopardize the program's functioning. The crawler is server-friendly, as it keeps the number of requests to a minimum and waits 60 seconds between requests. To deal with network issues, GNU Wget is used by the crawler as an auxiliary tool.

The crawler was scheduled to work around 11 o'clock everyday, using cron, a scheduling service included in most Linux distributions. The data was gathered between November 1, 2006 and March 29, 2007. After those five months, the corpus was frozen in order to get stable results and start analyzing the tag extraction methods that were being developed.

The crawling information was also used to better understand the community's habits. Regarding story submission, each day about 100 stories are promoted to their section's front page. The main topics of interest are, undoubtedly, technology and gaming, whereas the most unpoular one is sports. In short, *Digg*'s audience may be characterized mainly as tech-savvy, though it seems to be broadening its subjects of

[8]See the author's blog entry at http://kalsey.com/2006/08/and_thanks_for_all_the_fish/

interest, as topics like Politics or World News are becoming popular. As a consequence of this asymmetry on the user's interest, the amount of stories on each topic varies greatly, making story distribution in the corpus rather irregular.

3.3 Processing and Indexing Data

The information gathered using the previously described component should be processed to a meaningful representation. Otherwise it would not be very useful. Furthermore, the large amount of data calls for efficient structures, in order to speed up and reduce the memory print of the computer processes using such information. A standard way to represent these helpful structures is through indexes, which are appropriate when using large and semi-static collections, as in this case [1]. The one used in this system's data is the inverted index mechanism. Inverted indexes are data structures composed of words and occurrences. In short, each word points to every file that contains it.

In what comes to document representation, it is common procedure, in Information Retrieval, to apply the vector model [1]. Using this classic model, documents are represented as vectors in a t-dimensional space, that is, a space with as many dimensions as the number of distinct terms in the document collection. Each position of a document vector contains the weight of its corresponding term. A term's weight can be computed using different term-weighting schemes. The *tf-idf* is the best known scheme and uses the following formula:

$$w_{ij} = tf_{ij} \times idf_{ij} \tag{1}$$

Where, tf_{ij} is the frequency of the term i in document j, that is, the number of times the term i is found in document j; idf_i is the inverse document frequency and is given by:

$$idf_{ij} = \log \frac{N}{n_i} \tag{2}$$

Where N is the total number of documents in the collection and n_i is the number of documents containing the term i.

Considering the need for efficient structures, just as described, as well as an appropriate data representation, Apache's Lucene was chosen to be used in this project. Lucene is a well-known framework for information retrieval. It was chosen for being efficient, frequently upgraded, having a good documentation and support for most of the required features.

Lucene's frameworks generates its own index files. Hence, this module, imports the data collected in the previous stage, processes the story documents, that is, strips the HTML tags, discards stopwords, removes dates, number and long words, and indexes the documents. Additional data, required by the suggesting module, that can be precomputed to speed up the algorithm, was also computed and stored at this stage.

3.4 Tag Suggesting

The module described in this section does the most noticeable job in the system. It is responsible for taking a document and its additional information as input and presenting tags as recommendations. The algorithm for term selection consists of two distinct phases, which will be described next. In sum, the query document is first processed by the vector displacement module. Then, its words are ranked and presented to the user. Optionally, if no topic was assigned to the document, the built-in categorizer can be used for such matter.

3.4.1 Vector Displacement

The first stage on the process of selecting words is based on vector displacement. The process starts with the query document being converted to a vector, just like the other documents in the collection, to be placed in the vector space. The next step is to find the N most similar documents, that is, the closest ones, where N is a user-defined parameter. To quantify the degree of similarity between two documents, the following formula was used [1]:

$$\begin{aligned} sim(s,p) &= \frac{\vec{s} \cdot \vec{p}}{|\vec{s}| \times |\vec{p}|} \\ &= \frac{\sum_{i=1}^{t} w_{i,s} \times w_{i,p}}{\sqrt{\sum_{i=1}^{t} w_{i,s}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,p}^2}} \end{aligned} \tag{3}$$

Where $\vec{s}$ and $\vec{p}$ are the two vectors to be compared, $|\vec{s}|$ and $|\vec{p}|$ are their norms, $w_{i,s}$ and $w_{i,p}$ are the weights of term i in documents s and p, respectively. This formula computes the cosine of the angle formed by the two vectors and, thus, is usually called cosine similarity. The closer the cosine value is to 1, the more similar are the two documents. When the cosine value is 0, the vectors are orthogonal, that is, there are no common terms.

The vectors of the most similar documents are then used to alter the query vector's position, that is, to change its terms weights. Six different formulas were developed for that purpose, and are described next.

Method 1
Use terms from similar documents only, that is, discard the query's terms:

$$\vec{f} = \sum_{s \in S} \frac{1}{|S|}(w_{1,s}, w_{2,s}, ..., w_{n,s}) \tag{4}$$

Where $\vec{f}$ is the final vector, S is the set of similar documents and $w_{i,s}$ is the weight of term i in document s.

Method 2
Use all terms:

$$\vec{f} = \sum_{s \in S} \frac{1}{|S|+1}\big((w_{1,s}, ..., w_{n,s}) + (w_{1,q}, ..., w_{n,q})\big) \tag{5}$$

Where q is the query document.

Method 3
Use query terms only

$$\forall_{i \in Q} : f_i = w_{i,q} + \sum_{s \in S} \frac{1}{|S|} w_{i,s} \tag{6}$$

Where Q is the set of query terms and f_i is the weight of the term i in the final vector.

Method 4
Use all terms but take vector distance into account

$$\begin{aligned}\vec{f} &= \sum_{s\in S} sim(q,s)\times(w_{1,s},...,w_{n,s}) + \\ & sim(q,q)\times(w_{1,q},...,w_{n,q}) \\ &= \sum_{s\in S} sim(q,s)\times(w_{1,s},...,w_{n,s}) + (w_{1,q},...,w_{n,q}) \quad (7)\end{aligned}$$

Where *sim* is the cosine similarity, defined in 3

Method 5
Similar to Method 2, but changes the vectors importance.

$$\begin{aligned}\vec{f} &= 0.05\times\left(\sum_{s\in S}\frac{1}{|S|+1}(w_{1,s},...,w_{n,s})\right) + \\ & 0.95\times\left(\frac{1}{|S|+1}(w_{1,q},...,w_{n,q})\right) \quad (8)\end{aligned}$$

Method 6
Based on Method 4, but changes the vectors and the distance importance.

$$\begin{aligned}\vec{f} &= 0.05\times\left(\sum_{s\in S}\frac{sim(q,s)}{2}\times(w_{1,s},...,w_{n,s})\right) + \\ & 0.95\times\left(\frac{1}{2}\times(w_{1,q},...,w_{n,q})\right) \quad (9)\end{aligned}$$

Two different approaches to this vector displacement procedure were implemented. The first one was inspired by the *k-means* algorithm [5]. It starts by placing the query vector in space. Then, the vector is successively displaced by applying one of the methods described above until it stabilizes, that is, its values do not change between iterations, or until a user-defined limit of iterations is reached. Several experiments indicated that the vectors never stabilize. Because of that, a different approach had to be elaborated.

The second approach makes use of the document subtopics. Like in the previous approach, it starts by placing the query document in space. Then, when searching for the most similar documents, only the ones from the same subtopic are considered. In other words, a narrower version of such space is considered, one that contains only the documents which were assigned the same subtopic as the query document's. The vector is then displaced a single time, using one of the displacement methods described above.

3.4.2 Tag Extraction

The words to be suggested as tags are extracted from the displaced vector, using several formulas to rank them. The first measures to be experimented were weight (1) and the inverse document frequency (2). Ranking words by weight produced some satisfactory results. The inverse document frequency, on the other hand, turned out to be less useful, as most of the words were too rare to be considered as tags. Since many documents in the collection are blog entries or have comments written by their readers, this measure was bringing out misspelled words, neologisms, several words concatenated and even onomatopoeias. Words like 'simluator', 'mousejiggler' or 'duhhhhhhhhhh', for instance, were prone to appear as top terms.

Based on these two measures, two new ones were implemented. One of such measures consisted on computing the average of the inverse document frequency of all the terms in the vector and then sort the words by their closeness to that value:

$$idfavg = \sum_{t\in T}\frac{idf(t)}{|T|} \quad (10)$$

$$\begin{aligned}\forall t_1,t_2: \quad & rank(t_1) > rank(t_2) \Rightarrow \\ & |idf(t_1) - idfavg| < |idf(t_2) - idfavg| \quad (11)\end{aligned}$$

Where T is the set of terms. Put in words, the closest the term's inverse document frequency is to the average inverse document frequency, the higher its rank should be. The other measure consists of ranking the words in two phases. First, the words are ranked according to their weight. From that sorted set, only the top 50 words are kept. This new set is then sorted by inverse document frequency and the top terms are extracted. The idea is that, sometimes, rarer words are useful. However, because of the problems described above, they cannot be too rare. So, this measure, $w50idf$, selects the rare words from a set of not-so-rare ones.

The measures presented this far did not take the document topic into account. When considering the categories, other measures can be used. Inverse document frequency, for one, can be adapted to be used in a space partitioned by its documents categories using the following formula:

$$idfcat_{t,c} = \log\frac{N_c}{n_{i,c}} \quad (12)$$

Where i is a term, c is a category, N_c is the number of documents in category c and $n_{i,c}$ is the number of documents that contain the term i and are categorized as c. In practice it is like using only the vector space formed by the documents of the given category, instead of considering the whole vector space.

Yang and Pedersen suggest information gain, mutual information and χ^2 statistic as useful term selection methods to be used in text categorization [19]. Since the documents in the collection are categorized with *Digg*'s topics, these measures can be employed. Information gain is commonly used in Machine Learning as a means to measure how good a term is. It is computed using the following formula:

$$\begin{aligned}ig(t) = \quad & -\textstyle\sum_{c\in C} P(c)\log P(c) + \\ & P(t)\textstyle\sum_{c\in C} P(c|t)\log P(c|t) + \\ & P(\bar{t})\textstyle\sum_{c\in C} P(c|\bar{t})\log P(c|\bar{t}) \quad (13)\end{aligned}$$

Where C is the set of categories and t is a term. Mutual information is another well-known method for term selection and is computed as follows:

$$mi(t,c) = \log\frac{P(t,c)}{P(t)P(c)} \quad (14)$$

This measure depends on both the term to be evaluated and on its document's category. As a measure to be used in TC problems, the category of the submitted document is unknown. Thus, two alternative formulas are proposed:

$$mi_{avg}(t) = \sum_{c\in C} P(c)mi(t,c) \quad (15)$$

$$mi_{max}(t) = max_{c\in C}\{mi(t,c)\} \quad (16)$$

Finally, the third method, χ^2 statistic measures the degree of dependency between a term and a category:

$$\chi^2(t,c) = \frac{N[P(t,c)P(\bar{t}|\bar{c}) - P(t|\bar{c}P(\bar{t}|c)]^2}{P(t)P(\bar{t})P(c)P(\bar{c})} \quad (17)$$

Where N is the number of documents in the collection. As in mutual information, this formula depends both on the term and the category. So, two formulas are presented:

$$\chi^2_{avg}(t) = \sum_{c \in C} P(c)\chi^2(t,c) \quad (18)$$

$$\chi^2_{max}(t) = max_{c \in C}\{\chi^2(t,c)\} \quad (19)$$

In order to improve the efficiency on the computation of the feature selection methods just described, contingency tables described by Prabowo and Thelwal [13] were adapted and integrated in the system. They were precomputed at the data processing stage.

Since information gain was performing well, a new version of this measure was tried: $ig2$. *Digg* has a 2-level hierarchy of topics, that is, each subtopic has a parent topic and several subtopics as siblings. In the information gain measure presented before, only subtopics were considered. Our new version, instead of considering each subtopic as a category in the global space, considers only the documents of the parent topic. For example, when submitting a document labeled with 'Apple' as its subtopic, this measure considers only documents in subtopics descending from 'Technology', the parent topic. The formula used is the same as (13), but with C being the set of sibling subtopics of the query document's subtopic.

Some new methods, consisting of combinations of the measures described so far, were tried as well. Basically, they consist of multiplying the term frequency or the term weight by one of the other measures.

Finally, since *Digg* was used as data source, some additional information on the documents can be used as well. In this manner, the title and description were used as a means to boost the score of the selected words. After applying the term extraction measures, the system searches for words in the final vector that are also in the title or description, and raises their scores. The new score is computed by multiplying the old one by a factor of 3, if the word is found in the title, or by 2, if found in the description, or by 6, if found in both fields. This feature was named 'boost'. This feature can easily be turned off. This ensures Tess adaptability to websites that follow a structure different than *Digg*'s.

Additionally, a filtering process was experimented, which consisted on defining thresholds on information gain or inverse document frequency and automatically remove words with scores above or below such limits, depending on the rule. However, these filters turned out to be useless, as words that are apparently good appear mixed up with uninteresting ones, making it very difficult, if not impossible, to find good thresholds.

If every combination of measure was tried, we would have a total of 70 measures. However, some were abandoned along the research for their bad performance. Hence, a total of 30 measures are were tried, which are indicated in Tab 1.

The last step of this process is the removal of similar words.

w	$w50idf$	$w \times ig$	$tf \times ig2$	$boost(tf \times ig2)$
idf	idf_{avg}	$tf \times idf_{cat}$	$w \times ig2$	$boost(w \times ig2)$
ig	mi_{max}	$boost(w)$	$tf \times \chi^2$	$boost(tf \times mi)$
mi	mi_{avg}	$boost(tf \times ig)$	$w \times \chi^2$	$boost(w \times mi)$
χ^2	χ^2_{max}	$boost(w \times ig)$	$tf \times mi$	$boost(tf \times chi^2)$
$ig2$	χ^2_{avg}	$tf \times ig$	$w \times mi$	$boost(w \times \chi^2)$

Table 1: Term extraction measures available in Tess

To avoid having words like 'game', 'gaming' and 'gamer' on the same set of tags, a filtering procedure is applied. It consists of removing words with the same stem, using an implementation of the Porter stemming algorithm included in Lucene's framework. When two or more equal stems are found, only the word with higher score remains. This procedure improves the diversity of tags.

3.4.3 Categorizer

As stated before, the use of categorized documents improves the system's performance. However, not every website uses a predefined hierarchy of topics. Hence, in order to keep Tess as generic as possible, a categorizer was included. Since all the documents in the collection were retrieved from *Digg* and, thus, were already categorized – using the topic assigned by their publishers – the *k-nearest neighbor* (k-NN) method could be employed [12]. When given a document to classify, the categorizer finds the k most similar documents, using cosine similarity. The categorizer returns the topic associated with the majority of documents in the set.

To find a good value for k, we used a set of 50 already classified documents, and submitted them several times to the categorizer using different values of k. Our experiment suggested values between 10 and 13 as the most appropriate ones.

4. EVALUATION

In order to accurately assess the system's performance, a rigorous user-based evaluation had to be carried out. Such procedure helped perceiving which methods perform better, whether the system needs further improvement and, if so,what sort of changes may help enhancing it.

4.1 Experimental Setup

Given the large number of methods and, thus, of suggested sets of tags, the first step was to choose which methods to use. Since none of the vector displacement methods was clearly underachieving, they were all included in the evaluation procedure. Regarding term extraction, 16 of the 30 measures were discarded. Only the 14 most auspicious methods were kept: w, $boost(w)$, idf_{avg}, $w50idf$, $tf \times ig$, $w \times ig$, $tf \times mi$, $w \times mi$, $boost(tf \times ig)$, $boost(w \times ig)$, $boost(tf \times mi)$, $boost(w \times mi)$, $w \times ig2$, $boost(w \times ig2)$.

Another goal of this evaluation was to compare the system's performance when using its categorizer, with that obtained when using the topic selected by the story's submitter. Hence, the combinations of methods described above were used to generate sets of tags with and without the use of the system's categorizer. The k parameter was set to 12.

Two different tasks were requested to the users. Besides reading the document, the users were asked to write down 2 to 5 tags, like they would do on a regular social tagging

system. Next, they were prompted to validate each of the suggested tags as either good or bad. Each user was asked to evaluate 10 documents and each document was evaluated by 3 users. Therefore, the set of user-tags would have up to 15 distinct words. Tagging is a rather subjective process. In fact, tags may be validated differently depending on the users, as some are more demanding about the quality of tags than others. Also, the expertise on the document's subject may influence the user's choices. If a tag is meaningful on the document's context but the user is unaware of their relatedness, the tag will probably be rejected. Contrarily, a user with deeper knowledge on the subject would find the tag most useful. So, by forcing a document to be evaluated by 3 different users, the set of user-tags becomes larger and the diversity on the validation of the suggested tags increases.

According to the conditions just described, a total of 168 sets of tags were subject to evaluation. The size of each set of tags was fixed at 15. The experiment was taken by 4 groups of 3 users, which resulted in 12 users and 40 different documents, 10 for each group.

4.2 User-based Evaluation

To proceed with the evaluation of the tag suggesting system, a program to interact with users and manage the whole process had to be built. Three different stages are involved in this procedure. First the data has to be gathered and prepared. Next, the users perform the tasks they were assigned. Lastly, the results are treated using appropriate statistical measures.

The first stage of this procedure consisted of setting up the data to be used by the evaluation system. In this stage we used a tool we developed to collect 40 documents, 2 for each topic, submit them to Tess and gather the suggested tags. The cluster size, one of Tess input parameters, was set to 10. Then, the sets of tags on each document were joined, so that, when displaying the suggested tags to the user, no repeated words appear. Furthermore, the setup tool is responsible for creating the user groups and randomly assigning documents to each group, and consequently, to each user as well as the passwords for user authentication.

The second and most important stage consists on the user evaluation of the suggester's performance. To interact with the users, an online application was crafted. The web application is responsible for selecting the appropriate information, that is, the documents and tags under evaluation, and display them to the user. Moreover, it stores the results in their proper place, along with the user and document identification. The user interaction goes as follows:

1. Log in by entering the username and password
2. Pick a document to evaluate
   (a) Read the document
   (b) Write down 3 to 5 tags
   (c) Classify each and every suggested tag as either good or bad

The process terminates when all ten documents are evaluated. There is no time limit for this procedure and the user needs not evaluate every document in one session. Another factor worth noticing is that the user is asked to write down his own tags before seeing the system's suggestions, so that they will not influence his reasoning.

The evaluation process terminates when all the users have finished evaluating the documents they were assigned to. Having all the required data, we developed a tool to join the results and apply the appropriate statistical measures.

4.3 Evaluation Results

System performance can be assessed over different dimensions. Therefore, one must carefully choose the proper measures. Space and time consumption are usually worth measuring. The smaller the space and the shorter the response time, the better. However, in this case, we are more interested in assessing the system's output. Nevertheless, currently Tess uses a corpus of 844MB, which, through the indexing process, reduces its size to 197MB, approximately. When running, the suggester needs around 150MB of RAM and takes about 95 seconds to generate 180 sets of tags, with the cluster size of 10. Hence, on average, each set takes roughly 0.5 seconds to be produced.

Baeza-Yates and Ribeiro-Neto present some measures found to be very relevant, as they focus on the quality of the system's output and the users expectations [1]. In our case, we are going to discuss results for precision, coverage/recall and novelty. Such measures were adapted to better suit the tag suggesting problem.

Precision is among the most commonly used measures. In Tess' case, it indicates the amount of good tags in all of the suggested ones. Let T be the set of tags suggested by one of Tess methods and R_T a subset of T, containing only the tags found to be relevant by the users:

$$precision = \frac{|R_T|}{|T|} \tag{20}$$

Since there were three users evaluating the tags suggested for each document, three precision measures were derived from eq. (20):

- $precision1^+$ – a tag is considered relevant if at least 1 user finds it so
- $precision2^+$ – a tag is considered relevant if at least 2 users find it so
- $precision3$ – for a tag to be considered relevant, all 3 users must find it so

The average of each precision formula was computed for every combination of vector displacement methods with tag extraction measures. For instance, to find out the *precision1+* of the combination of the first vector displacement method with $tf \times ig$ term extraction measure, the following formula was employed:

$$\frac{1}{40} \times \sum_{d=1}^{40} \frac{|R1^+_{Td}|}{|T|}$$

Where Td is the set of tags produced by 'Method 1', described in Section 3 combined with '$tf \times ig$' for document d and $R1^+_{Td}$ is the set of tags considered relevant by at least 1 of the users. An analogous procedure applies when using other precision formulas or method combinations.

The evaluation results show that the precision value decreases as the relevancy condition gets stricter, which reinforces the lack of consensus when deciding whether a tag is relevant or not. In terms of vector displacement methods, 'Method 3' closely followed by 'Method 5'. As to term extraction measures, '$boost(tf \times ig)$', '$boost(w \times ig)$', and

'$boost(w)$' are usually the strongest ones. Also, the term extraction measures always perform better when using the boost feature. The first version of the information gain measure, ig, outperforms the second one, $ig2$. As for the worst term extraction measures, idf-related ones have the lowest precision values, which is, probably, because they bring out words that are too unusual to be considered good tags. The results when using the topic suggested by the system's categorizer are very similar to those obtained using the topic from *Digg*. In some cases, the categorizer can help improve the performance of some methods combinations, although the difference is not very significant. These results may be explained by the existence of some degree of similarity among different topics, or through the use of inappropriate topics, by some of *Digg* users.

Coverage is a user-oriented evaluation measure, similar to Recall. It indicates the amount of tags the users wrote down that were also suggested by the system. The original formula was adapted to better suit Tess:

$$coverage = \frac{|T \cap U|}{|U|} \tag{21}$$

Where T is the set of tags recommended by Tess and U is the set of tags written by the users. Analogously to what was done with the precision measures, the average coverage was computed for each combination of methods.

The results for coverage are consistent with the precision ones. The best vector displacement method is 'Method 3', closely followed by 'Method 6'. Regarding term extraction, '$boost(w)$' and '$boost(w \times ig)$' turn out to be the best measures for 'Method 3', though '$boost(tf \times mi)$' and '$boost(tf \times ig)$' are not too far behind. In 'Method 6', on the other hand, '$boost(tf \times mi)$ does not stands out as a top measure. On average, over 40% of the user-tags are covered by the system.

Another meaningful measure is the novelty ratio, which can be used to find out whether the system is revealing new relevant tags the user was previously unaware of. It is computed as follows:

$$novelty = \frac{|R_T - U|}{|R_T \cup U|} \tag{22}$$

Where R_T is the set of suggested tags considered relevant by the users and U is the set of tags wrote down by the users.

The results indicate 'Method 1' as the vector displacement method with highest novelty ratio. However, this method's precision and coverage are low so, it should not be considered a strong method. As for 'Method 3' and 'Method 6', the methods with highest precision and coverage rates, their novelty ratio is quite similar. The best term extraction measures, '$boost(tf \times ig)$', '$boost(w \times ig)$', '$boost(w)$', have an average novelty ratio of over 50%. That is, in average, more than half of the suggested tags found to be relevant, were new to the users. Altogether, despite useful, the novelty ratio cannot be a decisive measure by itself, as it is less important than precision and coverage.

To understand how the precision varies with the number of suggested tags a precision vs. number of tags graph was plotted. The resulting curve should be descent, indicating that the appearance of lower ranked tags decreases the precision, as they are less relevant. This was not the case, which suggests a faulty ranking mechanism. Ideally, less relevant words should not appear before more important ones.

To understand how users choose their tags, we applied the following formula:

$$tagsInDoc_i = \frac{|U_i \cap D_i|}{|U|} \tag{23}$$

Where U_i is the set of tags wrote down by the users and D_i is a set with all the words in the document being tagged, that is, document On average, approximately 80.6% of the user-suggested tags can be found in the document. This result complies with the conclusions drawn from the precision and coverage charts, as the strongest vector displacement methods are those who give more importance to the terms in the query document.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a tag suggesting system, aimed at textual resources only. Despite using *Digg*'s hierarchy of topics and resources, Tess is supposed to be adaptable to other types of websites, provided their resources are text-based.

The proposed solution is inspired in information retrieval and data mining techniques. Several methods were implemented. Some were discarded for their disappointing performance, whereas others were combined as an attempt to improve results. Indeed, the combinations proved to be advantageous, as their outcome was indicated as being the best by the evaluation procedure.

The preliminary results were promising. The system can obtain an average precision of 60%, which means over half of the tags it suggests are considered relevant. Apart from some performance tweaks Tess integration with websites and exposure to real-life situations could be experimented. According to the evaluation carried, there are still some drawbacks to solve and further improvements to work on. In fact, precision should be enhanced as well as the ranking mechanism. Also, some other combinations could be experimented. Also, more general term extraction measures should be developed to ensure the system's adaptability to other kinds of websites.

As to future work, we present some guidelines on how to further improve the system. Some authors, mentioned in *Related Work*, explore the graph-like structure of folksonomies. Their algorithms can be used to guess users interests, resources relatedness and to detect some relations between tags. Tess could take advantage of such information.

Currently, Tess deals only with single-word tags. However, there are occasions where tags with multiple words are more appropriate. For instance, 'Star Wars', a popular science fiction movie, if used as single-word tags, would lose their meanings. To add support for this feature, an inference technique based on n-gram frequency could be employed [8]. Such method estimates the probability of two or more words appearing together.

Intuitively, there is a strong relation between topics and dates, as people's interests are moved by trends and change as time goes by. Thus, the system could use dates to investigate trends and better decide on the resources subjects. Moreover, when dealing with documents with few words, this additional information can help selecting the similar

ones, since that cosine similarity is a method based only on words.

Usually, not all the information on a webpage is relevant. In fact, many webpages display, other than the main story, navigation structures with hyperlinks to other sections, advertisement and, sometimes, user comments. Hence, the irrelevant data should be stripped off from each file. One alternative would be using a machine learning technique to find patterns on a set of annotated documents and build a classifier. Another possible alternative would be to use natural language processing techniques. Usually, both advertisement and navigation structures are either small sentences or words. A simple part of speech tagger could be used as well, to classify words. Short strings where almost every word is either a noun, a verb or an adjective will probably be part of the navigation structure. This sort of reasoning may also work with advertisements. Alternatively, tools like Adblock [9] can help removing ads from the documents. Regarding user comments, they follow the same pattern. So, their structure may be deducted. At this point it is hard to predict which approach would perform better.

Additionally, some other data sources could be integrated in the tag suggesting system. A thesaurus could be used to detect synonymy and related words; *del.icio.us*, *BibSonomy*, for instance, could provide the tags assigned to the stories submitted on both *Digg* and the social bookmarking website; *Wikipedia* could be used to determine whether a word is strong by checking if there is an article about it and by the number of times it is referred on other articles. However, this would reduce the system's independence and could introduce an extra load on the algorithms. These factors must be considered, since, no matter how precise the system is, it cannot take longer to respond than what the users are willing to wait for.

In conclusion, Tess is a tag suggesting system, completely functional and ready to integrate in a social content website. It was started from scratch, so, it took a lot of experimentation to define which direction would be best. Now that it is more stable and mature, there are some enhancements that may improve its performance. The scope is, for now, limited to text-based resources only, which still accounts for the majority of the resources on *Digg*'s front pages and, probably, of other popular websites.

6. REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] Christopher H . Brooks and Nancy Montanez. An analysis of the effectiveness of tagging in blogs. In *Proceedings of the 2005 AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, pages 9–14. Stanford, CA., 2006.

[3] Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.

[4] Digg.com. Digg / frequently asked questions, accessed July 31, 2007. http://www.digg.com/faq.

[5] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, USA, 2000.

[6] Thomas Hofmann and Jan Puzicha. Mixture models for co-occurrence and histogram data. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, pages 192–194, Washington, DC, USA, 1998. IEEE Computer Society.

[7] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC '06: Proceedings of the 3rd European Semantic Web Conference*, pages 411–426, 2006.

[8] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[9] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, New York, NY, USA, 2006. ACM Press.

[10] Adam Mathes. Folksonomies - cooperative classification and communication through shared metadata. Available at: http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html, 2004.

[11] Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM Press.

[12] Sushmita Mitra and Tinku Acharya. *Data Mining: Multimedia, Soft Computing, and Bioinformatics*. Wiley-Interscience, 2003.

[13] Rudy Prabowo and Mike Thelwall. A comparison of feature selection methods for an evolving rss feed corpus. *Inf. Process. Manage.*, 42(6):1491–1512, 2006.

[14] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[15] Win Treese. Web 2.0: is it really different? *netWorker*, 10(2):15–17, 2006.

[16] Darren Waters. Readers are digging the news online. *BBC News*, June 2006. Available at: http://news.bbc.co.uk/2/hi/technology/5128386.stm.

[17] Harris Wu, Mohammad Zubair, and Kurt Maly. Harvesting social knowledge from folksonomies. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 111–114, New York, NY, USA, 2006. ACM Press.

[18] Xian Wu, Lei Zhang, and Yong Yu. Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 417–426, New York, NY, USA, 2006. ACM Press.

[19] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997.

[9] http://www.adblock.org/