Explorations in Tag Suggestion and Query Expansion

Jian Wang and Brian D. Davison
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015 USA
{jiw307,davison}@cse.lehigh.edu

ABSTRACT

The query used in a search system is only an approximation to the user's true information need, and as a result, many factors can reduce the quality of search results. One is query ambiguity, causing searchers with different needs to issue the same query. For example, for the query *java*, some users may want to find *java tutorial* while others may want to download *java software*. Other factors include a vocabulary mismatch and a lack of knowledge regarding the contents of the document collection. In any case, many users benefit from assistance in forming a good query. As a result, some commercial services provide query suggestions for many queries.

In this paper, we propose a `Tag Suggestion System` that takes advantage of tags associated with query results to expand a searcher's query. Since not every web page is associated with existing tags, we first build an auto-tagging system which can assign multiple tags to web pages, including news, blogs, etc. The current system contains the most popular 140 tags in del.icio.us, with high precision performance.

A small user study is performed to evaluate anecdotally the performance of our Tag Suggestion System, showing better quality than the query suggestion mechanisms provided by Yahoo! and Google. The result pages of expanded queries generated by the Tag Suggestion System are also significantly better than those of the Google original system.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation

Keywords

Tag Suggestion, Query Expansion

SSM'08, October 30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

Today's search engines simply use the keywords in the user query, even though some queries have various meanings. For example, *rotten tomato* may indicate a food search or a search for movie reviews. Since there may be many meanings, displaying results from one or more categories in which the user is truly interested is much more efficient for the searcher than to display results from all interpretations. By using a suggestion list of expanded queries, the user can figure out what they are looking for exactly and it is easier for them to find the target pages.

This paper is concerned with exploring tags for query expansion and suggestion. Social bookmarking is a Web 2.0 service which allows users to save and organize their bookmarks online with text descriptions, i.e., *tags*. Tags can reflect various users' perspectives about a single unique article, including blogs, news, and more. Thus, several tags attached with an article could help to better understand the web page. Different from the traditional topical categorization, such as the dmoz Open Directory Project or Yahoo!, the tags would be much easier to understand and adaptive to emerging topics.

Since most online documents are not already tagged, we first need to build an automatic tagging system to assign tags to the input article. In the current system, we only use page content without any usage data from the current user nor anyone else. Services with usage data can recommend tags that users have used previously or tags that are popular for this document. Both precision and recall performance could be easily enhanced when usage data is available to combine with the text method which we propose here. Yet our system can be applied to most online web pages when prior usage data is not available.

Once an auto-tagging system is implemented, every result page returned by the search engine would be assigned relevant tags according to its content. Such tags can then be used in the process of query expansion and suggestion. After a list of query suggestions are produced by the system, we can either ask the user to choose the ones he or she prefers, or automatically combine the search results for these query suggestions and provide the results to the user instead of the original query. A user study was performed to compare the performance for these two methods. Furthermore, the results are also compared with current popular commercial search engine performance. The results of our user study show that users prefer the query suggestions provided by our Tag Suggestion System and the search results of the expanded query are better than that of the original query.

The contribution of this paper includes the following two aspects.

- To build an auto-tagging system which can assign multiple tags to a web page, with high precision performance.
- To provide a high quality list of query suggestions to a user, according to the original query submitted.

2. RELATED WORK

This paper brings together two areas: text classification used for auto-tagging, as well as query expansion and suggestion. Both of them have been the subject of prior effort. In this section, we review some of work in these two areas.

2.1 Auto-tagging System

Fujimura et al. [7] proposed a method of multi-autotagging, based on k-nearest neighbor algorithm (KNN). They also merged tags with the same meaning and identified informative tags. Their method to assign multi-tags is worth using for reference. In the paper, they used the KNN classifier and the method to collect similar entries by Hyper Estraier [9] which is an open-source full-text search software released under the terms of the GNU Lesser General Public License (LGPL).

Brooks and Montanez [4] analyzed the effectiveness of tags for classifying blog entries by gathering the top 350 tags from Technorati and measuring the similarity of all articles that shared a tag. They found that tags were useful for grouping articles into broad categories, but less effective in indicating the particular content of an article. Their approach to auto-tagging was based on keyword extraction, which considered terms highly scored in TF-IDF weighting as tags. It is quite different from ours since they did not build any classifier and took advantage of former documents associated with these tags.

Lee and Chun [10] proposed a novel approach to automatic tag recommendation for blogs. It made use of collective intelligence extracted from Web 2.0 collaborative tagging as well as word semantics to learn how to predict the best set of tags to use, using a hybrid artificial neural network. Collaborative tagging creates collective intelligence by observing how different users tag similar content. Our approach only focuses on the tags assigned to the document, instead of who assigned the tag, which simplifies the system, requires fewer resources than their approach and is more widely applicable. Their research made use of the collective intelligence to automatically generate tag suggestions to blog authors based on the semantic content of blog entries.

Heymann et al. [8] predicted tags based on page text, anchor text, surrounding hosts, and other tags applied to the URL. They found that tag-based association rules can produce very high-precision predictions and also give deep understanding into the relationships between tags. The relationships between all tags could be very useful during the automated prediction process. This analysis could be combined with our current auto-tagging system to provide better performance.

The efficiency of the automated tag recommendation is also a significant concern. Song et al. [13] proposed a highly-automated, novel framework for real-time tag recommendation. Their average tagging time for testing a document is around 1 second.

2.2 Query Expansion And Suggestion

Chirita et al. [5] proposed to improve short keyword queries by expanding them with terms collected from each user's Personal Information Repository, thus implicitly personalizing the search output. They introduced five broad techniques for generating the additional query keywords, which would be substituted by multiple tags in this paper. In addition, they also discussed factors which help to add a flexible number of keywords to the user query instead of adding a fixed number of keywords. This could be used to improve the performance of our system in the future.

The most effective query expansion methods rely on retrieving documents which are used as a source of expansion terms; however, retrieving those documents is costly. Billerbeck and Zobel [3] proposed a new method that drew candidate terms from brief document summaries that were held in memory for each document. While approximately maintaining the effectiveness of the conventional approach, this method significantly reduced the time required for query expansion by a factor of 5-10. Since our approach not only retrieves the documents, but also assigns tags based on document content, their approach is not applicable in this paper.

Cucerzan and White [6] investigated a novel query suggestion technique that selected query refinements through a combination of many users' post-query navigation patterns and the query logs of a large search engine. Their findings demonstrated the effectiveness of using landing pages for the direct generation of query suggestions, as well as the complementary nature of the suggestions it generated with regard to traditional query log based refinement methodologies. Instead of using landing pages, which requires more resources besides the original query word, we only use the initial results of the query and provide several suggestions to the user.

3. AUTO-TAGGING SYSTEM

3.1 System Overview

Our auto-tagging system is implemented by a set of binary classifiers, in which each classifier is focused on a single tag. The data is collected from del.icio.us, including the current 140 most popular tags (based on the number of web pages attached with this tag), and around 300 pages for each tag. A binary classifier is used for each tag under the assumption that tags are independent of each other.

A support vector machine (SVM) technique is adopted in the classification process since an SVM is expected to achieve high precision. We choose Rainbow [11] to build the model for classifier, with the basic features of SVM by default. Since SVM is sensitive to the ratio of positive and negative training documents, this parameter is tuned to achieve better performance.

3.2 Training Datasets

In this paper, the 140 most popular tags are chosen from a tag cloud in del.icio.us, which is shown in Figure 1. For each tag, around 300 associated web pages are crawled from del.icio.us. For each webpage, we record all associated tags in del.icio.us.

For each tag, we randomly select a number of documents (the total amount is referred as M below) that are associated with the current tag as positive training documents.

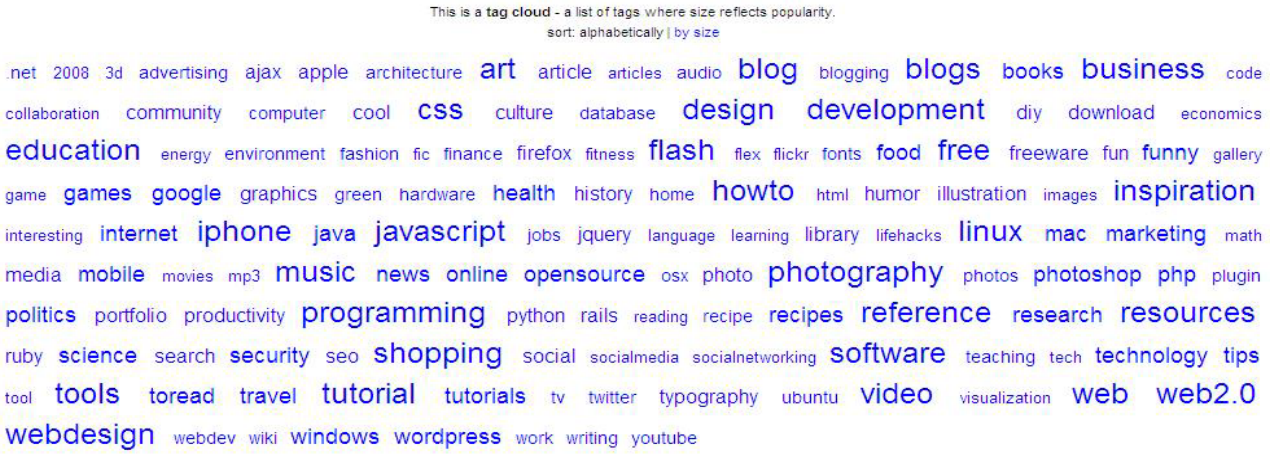

Figure 1: Tag Cloud in del.icio.us

For example, if the document originally has tags *java, tutorial, reference*, it would be selected as the positive training documents of *java, tutorial,* and *reference*.

For each tag, we similarly randomly select a number of documents (the total amount is referred as K below) as negative training documents if the document is not associated with the current tag. For example, if the document originally does not have tag *java*, it would be selected as the negative training documents of *java*.

3.3 Test Datasets

The test dataset contains 503 webpages which are originally associated with multiple tags in del.icio.us. All webpages are greater than 800 bytes.

The evaluation process is based on the original tags in del.icio.us. However, since every webpage is tagged by multiple users in del.icio.us, the accuracy of these tags is assumed to be high.

3.4 Pre-processing On Datasets

To prepare the data for use in our system, we:

- We extract the text of the web page by filtering the markup from the original document.
- In order to provide enough information for use in classification, documents are limited to be greater than 800 bytes. Smaller documents are removed from the dataset.
- Stopwords, as enumerated by the SMART Information Retrieval System[2], are removed from the documents.

3.5 SVM Classifier Performance

After we construct the training dataset for each tag, i.e., each class, we build an SVM model for this tag. Since SVM performance is sensitive to the ratio of positive training documents (M) and negative training documents (K), we built a series of SVM models, based on different values of M and K.

Every webpage in the test dataset is provided as input of the system and is likely to have multiple tags suggested for it. Since we cannot compare probability value from different SVM classifiers and rank the suggested tags, we retain all tags suggested for the web page. Precision, recall and F1-measure are used to evaluate the result.

In order to demonstrate that our auto-tagging system provides tag suggestions with high quality, we compare our results with two related works. The first is the TagAssist system in [14], which uses exact word match to compare tags. The second one is the AutoTag system in [12], which uses string distance to compare the tags for web pages which originally have 3 or more tags. In an attempt to show comparable results to these approaches, we use three evaluation methods for our auto-tagging system.

- **Exact word match**: use exact word match between original tags and tags suggested by our system for all 503 web pages.
- **String distance**: use string distance to compare the tags rather than exact string match for all 503 web pages, in order to reduce the affect of minor differences in tags (*blogs* and *blogging*).
- **String distance with min3**: use string distance to compare the tags for web pages which originally have 3 or more tags. Among 503 webpages, there are 328 webpages which have 3 or more tags.

3.5.1 *Fix M, vary K*

If we fix the value of M, i.e., let M be 60, 100, 200, 400 and 500, while changing the value of K, the trend is similar for every M and for all three methods.

As shown in Figure 2, if the number of positive training documents is fixed to be 400, precision continues to increase with the increase of K. Recall continues to decrease while F1-measure reaches its peak value during the increase of K. The number of result pages with 1 or more tag suggestions decreases with the increase of K.

3.5.2 *Fix K, vary M*

If we fix the value of K, i.e., let K be 60, 120, 240, 360, until 1500, while changing the value of M, the trend is also similar for every K and for all three methods.

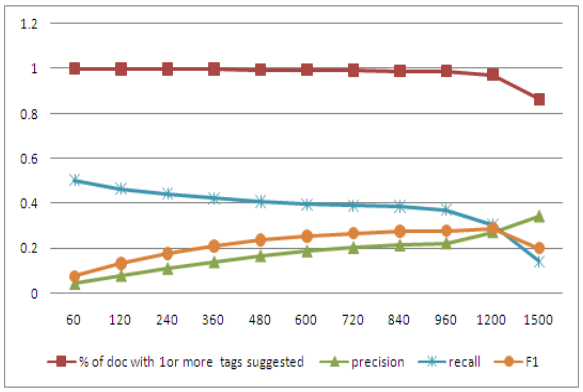

Figure 2: Performance of SVM classifiers on a series of K while M is fixed to be 400, with exact word match

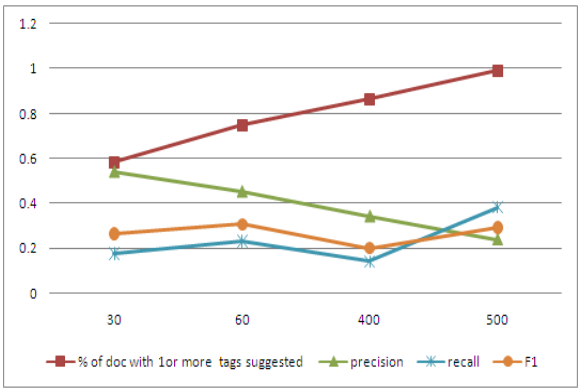

Figure 3: Performance of SVM classifiers on a series of M while K is fixed to be 1500, with exact word match.

Table 1: Performance of SVM classifiers with 60 positive and 1500 negative training documents.

Method	Precision	Recall	F1
Exact word match	0.45252	0.23243	0.30711
String distance	0.54206	0.25754	0.34918
String distance with min3	0.63104	0.23540	0.34289

As shown in Figure 3, if the number of negative training documents is fixed to be 1500, precision continues to decrease with the increase of K. Recall continues to increase while F1-measure fluctuates during the increase of K. The number of result pages with 1 or more tag suggestions increases with the increase of K.

3.5.3 SVM Classifier Analysis

In a tag suggestion system, it is more important for a user to always think the proferred tags are good ones than to have missed suggesting a valid tag. That is, the user is less likely to miss an absent tag and more likely to be annoyed at an inappropriate tag. Thus our target is to get the highest precision value. To achieve this, the best training set should contain a small number of positive documents and a large number of negative documents. The higher the ratio (number of negative documents / number of positive documents), the better the precision value. Yet considering the number of documents which are suggested with 1 or more tags, the recall value and the F1-value, we should not let the ratio (number of negative documents / number of positive documents) be too large. Besides, the total number of training documents should not be too large considering the running time and efficiency problem.

The overall best performance for SVM classifiers is achieved with 60 positive documents and 1500 negative ones in training dataset, targeting on high precision and acceptable performance on other factors.

Table 1 shows the perfomance for all three evaluation methods. It is obvious that if we use string distance to compare tags, instead of exact word match, precision would increase. If we only use webpages with 3 or more tags in del.icio.us, it is more likely for result tags to be included in the original tag sets, which leads to higher precision. For `Exact word match` and `String distance`, there are 503 documents in the test dataset after preprocessing. For `String distance with min3`, there are 328 documents in the test dataset after preprocessing that contain 3 or more tags. For all three methods, around 75% of documents receive one or more suggested tags from our system.

3.5.4 Comparison With Related Work

The TagAssist system provided tag suggestions for new blog posts by utilizing existing tagged posts. It performed lossless compression over existing tag data in order to increase the quality of suggested tags. Their study used human judges to evaluate the appropriateness of tags of a post. The 225 responses from 10 different judges gave the original tags 48.85% accuracy and TagAssist's tags 42.10% accuracy. For the sake of comparison to other systems, they also performed the evaluation by processing 1000 posts through their system and then comparing the suggested tags against those originally assigned. The precision and recall values for the system was 13.11% and 22.83%. Since they used exact word match, our result of `Exact word match` is comparable to theirs. Our best precision and recall for `Exact word match` is 45.25% and 23.24%, respectively, which gives an improvement of 245% and 1.8%, respectively. If we use 400 positive training documents and 720 negative ones, the recall is .39 with the precision being .20.

The AutoTag system suggested tags for weblog posts using collaborative filtering methods. For evaluating their method, AutoTag was used to tag 6000 of the "tagged posts" in their corpus, the posts of which were assigned tags by their authors. They used string distance to compare the tag rather than the exact string match and they only used posts with 3 or more tags. Their automated result for precision@10 was 0.40 and for recall@10 was 0.49. Their evaluation method is comparable to `String distance with min3` of our system. Our best precision for `String distance with min3` is 0.63 for precision and 0.24 for recall. The precision value is stronger than theirs while the recall value is not. If we use 500 positive training documents and 1500 negative ones, the recall is 0.386 with the precision being 0.388, which is more comparable to their results. In their system, they only assigned tags to weblog posts and they took advantage of the tags which the author used before. If the previously-used tags appeared in the ranked list, AutoTag boosted their score by a constant factor. This can explain why the recall value of our system is not as good as we only

analyze the content of the webpage itself without any usage data. Our approach would benefit significantly from usage data, but the proposed approach applies even when we do not have usage data for this user or this document.

4. QUERY EXPANSION

After we successfully construct an auto-tagging system with good performance, we can take advantage of it for suggesting query expansion to the user. We assume this system would be implemented by a search engine, and thus could perform the tag suggestion offline for every document (or at least for all docs that rank highly). Query expansion could thus be performed as soon as user submits the initial query.

4.1 Using Tags To Expand Query

Our initial goal is to provide a list of query suggestions according to the initial query. Every query suggestion should reflect an aspect of meaning for the original query, for example, *tomato food* and *tomato movie* are two aspects of the query *tomato*.

After the search engine returns a number of documents for the original query, we can analyze this set of documents to get hints of the possible meaning of the query. For instance, among the top 10 results of query *tomato*, if 5 documents are talking about movie reviews and another 5 are talking about vegetable and food, we can conclude that the query *tomato* has at least two meanings. Different users may want to pursue either of the meanings.

In order to analyze the initial documents which are returned to the original query, we can take advantage of the tags associated with this document. "A tag is simply a word you use to describe a bookmark", according to del.icio.us; what is more, a tag reflects the user's understanding of the document. This characteristic makes tags different from category information, which could be provided by ODP and Yahoo!. A tag could be the category name of any level in ODP structure, as well as other words describing one aspect of the document.

Since not every document in the web has been assigned tags, we can take advantage of our auto-tagging system to assign multiple tags to the document, including news, blogs, etc. After the search engine returns a number of results to a certain query, we can assign (or look up) tags for every document in the initial result list. The similar preprocessing work would also be performed on the documents. According to a tag's characteristic, the tags associated with the document would reflect the meaning of the document to some extent. Then we will analyze the tag results of all documents in the initial result and expand the query with appropriate tags. In this paper, we simply add the tag to the initial query term(s). For example, if the tag *programming* is popular in the initial results of query *java*, we will expand *java* to be *java programming*.

In this paper, we choose the top 8 popular tags to expand the original queries and the popularity is measured by how many web pages are assigned this tag.

4.2 Manually Selected By User

After we get a list of possible query expansions, we need to find a way to choose the best suggestion for the user. Many users may enter the same query such as *java*, yet have different search purposes. Some people may want to download java software while others may want to find a java tutorial. Without any information about user's preference, it is hard for search engine to tell what results for which the user is looking.

In order to provide search results which can best satisfy a user's needs, we can ask the user to choose the queries that they prefer among a list of query suggestions. It is referred below as `Tag Suggestion System`.

4.3 Automatically Combine Results

Other than asking the user to choose the suggestions they prefer, we can also combine the results for every suggestion automatically. The user study we described below will test whether this method is useful or not. It is referred below as `Tag Auto-combine System`.

5. USER STUDY

To evaluate the performance of Tag Suggestion System, we did a user study and compared our performance with that of two popular commercial search engines, Google and Yahoo!. Queries in Google's `Searches related to` and those in Yahoo!'s `Also Try` are regarded as their suggestions for the original query.

5.1 Data Set

The user study included four graduate students from Lehigh University. Every user chose five queries to search, resulting in 20 unique queries. To get the initial results for the original query, we choose Google as our search engine. The top 50 web pages could be provided by Google after some level of filtering [1]. The first is "Near-Duplicate Content Filter", which means that if multiple search results contain identical titles and snippets, then only one of the documents is returned. The second one is "Host Crowding", which means that if multiple results come from the same Web host, then only the first two are returned. After experimenting with both on results without filtering and those after filtering, we found that the results after filtering are better for our system. The top 50 web pages returned by Google are applied as input to our auto-tagging system, which outputs several tags for every page. The result of Tag Suggestion System and Tag Auto- combine System are based on these tags.

5.2 First Step

After the user submits the initial query, he or she will be provided with a list of suggestions to the query, which would be a random anonymous mixture of Google suggestions, Yahoo! suggestions and Tag suggestions. Interestingly, there is at most one suggestion which appears in both suggestion lists. Every user selects the ones he or she prefers, without restriction on the number of suggestions selected. It could be 0, which means none of the query suggestions satisfies his or her requirement, or it could be up to all suggestions. Some sample queries, query suggestion in Google Suggestion, Yahoo! Suggestion and Tag Suggestion Sytem, as well as user's selection are shown in Table 2.

The average performance of all 20 queries is listed in Table 3.

If we provide the top 8 suggestions from our Tag Suggestion System, on average the user chose 17% of Google suggestions, 9% of Yahoo! suggestions, and 23% of Tag Suggestion System. Among all of the suggestions that the user selected, 50% were provided by our Tag Suggestion System, while 39% from Google and 11% from Yahoo! on average.

Table 2: Sample query, query suggestion and user selection

Query	java
Google Suggestion	java update, java plugin, java vista, microsoft java, java 6, java 5, java 2, flash
Yahoo! Suggestion	java download, java games, sun java
Tag Suggestion System	java language, java software, java religion, java programming, java mobile, java finance
User Selection	java 5, java 2, java language, java programming
Query	philly
Google Suggestion	craiglist philly, eros philly, philly restaurants
Yahoo! Suggestion	philly cheese steak, philly news, q102 philly
Tag Suggestion System	philly food, philly travel, philly flickr
User Selection	philly restaurants, philly food, philly travel
Query	acadia
Google Suggestion	gmc, acadia reviews, gmc acadia reviews, acadia suv, acadia hospital, acadia realty trust, acadians, bar harbor
Yahoo! Suggestion	gmc acadia, acadia national park, acadia university
Tag Suggestion System	acadia travel, acadia technology, acadia tech, acadia environment
User Selection	acadia national park, acadia travel
Query	facebook
Google Suggestion	new facebook, facebook design
Yahoo! Suggestion	facebook login, facebook home, facebook login page
Tag Suggestion System	facebook work, facebook network, facebook social, facebook web2.0, facebook technology, facebook tech, facebook teaching, facebook resources
User Selection	facebook design, facebook network, facebook social

It is obvious that the performance of Tag Suggestion System is better than that of Google suggestions and Yahoo! suggestions. Users tend to select more from the suggestions provided by Tag Suggestion System.

5.3 Second Step

After the user selects the query suggestions according to his or her preference, the system provides the search results which contain the top 10 Google results with original query, top 10 Google results with each modified query (the query suggestions he or she select in the last step), as well as 4-16 results that Tag System generated automatically (without knowing the user's selection). All the results are mixed together randomly so the user does not know where each result comes from. The user was asked to give relevance judgment to the results, from 0-5 while 0 means he or she does not like the page at all and 5 means he or she thinks the page is exactly what he or she is looking for.

For every suggestion system, we calculate the improvement percentage with three metrics, based on the relevance score for original Google results, which is shown in following equations.

$Pa(a = 1, ..., m + n)$ is the average relevance score for Google original results for $Qa(a = 1, ..., m+n)$. For Google/ Yahoo!/ Tag Suggestion/ Tag Auto-combine System, $Qi(i = 1, ..., m)$ are those queries for which the user has chosen at least one suggestion in this system. Ri is the average relevance score of the suggestions selected by user in the system.

$Qj(j = 1, ..., n)$ are those queries which have no suggestion in this system or the user has not chosen any suggestions in this system. For Qj, we use the average relevance score of Google original results to represent the score for suggestions results during calculation. Thus, Rj, the average relevance score for Qj, is assigned to be equal to Pj.

Metric 1: (Average relative improvement, considering all Qi and Qj)

$$M1 = \frac{\sum_{a=1}^{m+n} \frac{Ra-Pa}{Pa}}{m+n}$$

$$= \frac{\sum_{i=1}^{m} \frac{Ri-Pi}{Pi} + \sum_{j=1}^{n} \frac{Rj-Pj}{Pj}}{m+n} = \frac{\sum_{i=1}^{m} \frac{Ri-Pi}{Pi}}{m+n}$$

, given that Rj=Pj (j=1,...,n)

Metric 2: (Improvement for average relevance score, considering all Qi and Qj)

$$M2 = \frac{\frac{\sum_{a=1}^{m+n} Ra}{m+n} - \frac{\sum_{a=1}^{m+n} Pa}{m+n}}{\frac{\sum_{a=1}^{m+n} Pa}{m+n}} = \frac{\sum_{a=1}^{m+n} Ra - \sum_{a=1}^{m+n} Pa}{\sum_{a=1}^{m+n} Pa}$$

$$= \frac{\sum_{i=1}^{m} Ri + \sum_{j=1}^{n} Rj - \sum_{i=1}^{m} Pi - \sum_{j=1}^{n} Pj}{\sum_{a=1}^{m+n} Pa}$$

$$= \frac{\sum_{i=1}^{m} Ri - \sum_{i=1}^{m} Pi}{\sum_{a=1}^{m+n} Pa}$$

, given that Rj=Pj (j=1,...,n)

Metric 3: (Average relative improvement, only considering Qi)

$$M3 = \frac{\sum_{i=1}^{m} \frac{Ri-Pi}{Pi}}{m}$$

In the first two metrics, the improvement score is affected by both the result page quality and the frequency that the user selects suggestions from this system, which is more comprehensive than the third metric. The improvement of average relevance score, in the second metric, is commonly used in other papers, and shows a more general result.

From Figure 4, we can see that for all three metrics, the performance of Tag Suggestion System is the best. If both

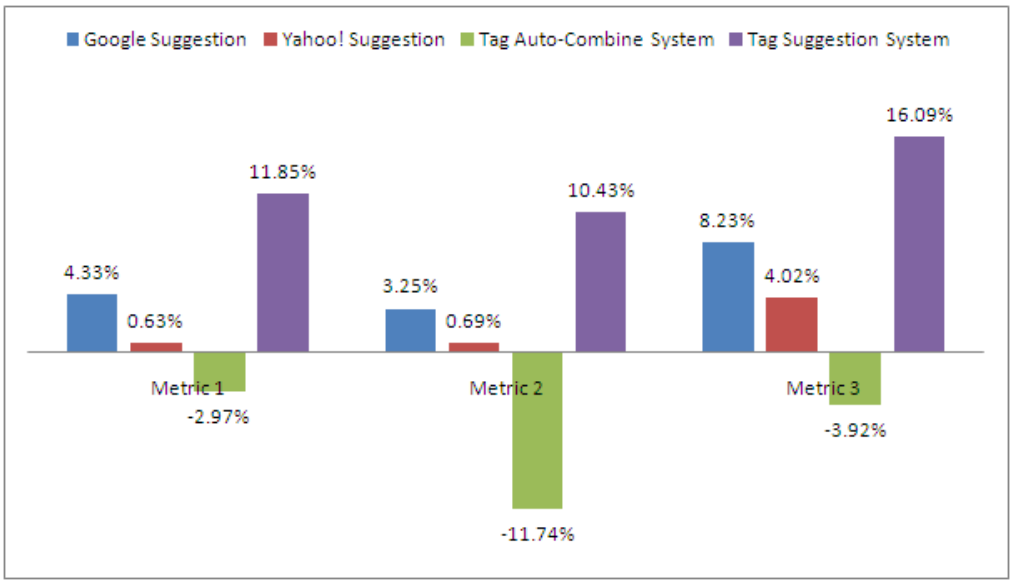

Figure 4: Improvement of the Suggestion System

Table 3: Performance of first step in user study

	Google	Tag Suggestion System	Yahoo!
# of suggestions provided in each system	6.65	5.70	3.15
# of suggestions selected by user in each system	1.05	1.25	0.32
% of suggestions selected in each system	17%	23%	9%
% of all selections from each system	39%	50%	11%

Qi and Qj are considered, the average relative improvement (Metric 1) of Tag Suggestion System is 11.85%, while 4.33% for Google Suggestion and 0.63% for Yahoo! Suggestion. The improvement for average relevance score (Metric 2) of Tag Suggestion System is 10.43%, while 3.25% for Google and 0.69% for Yahoo!. If only Qi is considered, i.e., only to consider those queries that have at least one Google suggestion, Tag Suggestion, or Yahoo! suggestion, the average relative improvement (Metric 3) of Tag Suggestion System is 16.09%, while 8.23% for Google and 4.02% for Yahoo!.

In contrast, the performance of the Tag Auto-combine System is noticeably poor, with scores below the baseline Google approach for all three metrics.

From this result, it is obvious that Tag Suggestion System can provide better performance, and the system should let user select suggestions by themselves instead of automatically combining results for each provided by Tag Suggestion System.

We also did a t-test among all the metrics, which is shown in Table 4. The statistic p value for Google's original result and the Tag Suggestion Suggestion result is 0.024, giving a confidence of 98% that a statistically significant difference between them exists. In addition, the p value for Tag Suggestion System result and Tag Auto-combine System result is 0.008. The p value for Tag Suggestion System result and Google Suggestion result is 0.039 and that for Tag Suggestion System result and Yahoo! Suggestion result is 0.035. So we are confident that our conclusion for Tag Suggestion System is valid and the result of Tag Suggestion System is significantly better.

6. CONCLUSION AND FUTURE WORK

In this paper, we build an auto-tagging system to assign tags to webpages. Our approach focuses exclusively on the textual content, and thus is applicable when no usage information is available. It takes advantage of the characteristics of tags to expand the original query and provide expansion options for the user to select. A user study is performed to evaluate the performance of queries suggested by our Tag Suggestion System, showing better performance than that of Google suggestion and Yahoo! suggestion. The result pages of expanded queries generated by the Tag Suggestion System are significantly better than those of the Google original system.

Besides a larger scale user study, questions to address in future work include:

- How would the system be improved to scale up to more tags, such as millions of tags? This would allow the system to cover a larger variety of queries, including less popular ones. A larger test dataset would be needed to evaluate the classifiers' performance.
- How to improve the performance of auto-tagging system? The misclassification costs are asymmetric; thus the classification could be performed with a cost-sensitive classifier instead of downsampling the collections.
- How can we reduce the run time for the auto-tagging system?
- How to use different suggestion mechanism for different type of query and search intent?

Table 4: Significance test among all system results

	Google Suggestion	Yahoo! Suggestion	Tag Suggestion	Tag Auto-Combine
Yahoo! Suggestion	0.217			
Tag Suggestion	0.039	0.035		
Tag Auto-Combine	0.051	0.123	0.008	
Google's Original Results	0.156	0.107	0.024	0.146

Acknowledgments

The authors would like to thank our study participants and the reviewers for their helpful comments. This work was supported in part by a grant from the National Science Foundation under CAREER award IIS-0545875.

7. REFERENCES

[1] Google soap search api reference. http://code.google.com/apis/soapsearch/reference.html.

[2] Smart information retrieval system. ftp://ftp.cs.cornell.edu/pub/smart/english.stop.

[3] B. Billerbeck and J. Zobel. Efficient query expansion with auxiliary data structures. *Inf. Syst.*, 31(7):573–584, 2006.

[4] C. H. Brooks and N. 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.

[5] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14, New York, NY, USA, 2007. ACM.

[6] S. Cucerzan and R. W. White. Query suggestion based on user landing pages. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 875–876, New York, NY, USA, 2007. ACM.

[7] S. Fujimura, K. Fujimura, and H. Okuda. Blogosonomy: Autotagging any text using bloggers' knowledge. In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 205–212, Washington, DC, USA, 2007. IEEE Computer Society.

[8] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *31st Annual International ACM Special Interest Group on Information Retrieval (SIGIR'08) Conference*. ACM, 2008.

[9] M. Hirabayashi. Hyper estraier: a full-text search system for communities. http://hyperestraier.sourceforge.net/index.html.

[10] S. O. K. Lee and A. H. W. Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ann semantic structures. In *ACOS'07: Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science*, pages 88–93, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[11] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[12] G. 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.

[13] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. Giles. Real-time automatic tag recommendation. In *The 31st Annual International ACM SIGIR Conference*. ACM, 2008.

[14] S. Sood, S. Owsley, K. Hammond, and L. Birnbaum. Tagassist: Automatic tag suggestion for blog posts. ICWSM, 2007.