Improving Web Personalization via User Interest Hierarchy and Scoring Techniques

Chris TANNER
Florida Institute of Technology
Melbourne, FL 32901, USA
ctanner@fit.edu

1 Introduction

The World Wide Web is rampantly growing, providing users with a vast amount of information from which to search and explore. However, the retrieval of information is rather basic in that search engines often do not make distinctions between users; most search engines provide different users with identical results when their search queries are the same. We assert that this is naive and rather shallow, for the actual users are not identical, suggesting that their desired results are also not exactly the same. For example, if two users each search for "Windows," maybe they desire two entirely different items: one user is interested in the aptly named operating system while the other is interested in purchasing glass panels through which to look. This scenario demonstrates the importance of learning what a particular user is interested in.

Moreover, most current systems now only consider the specific term that a user searches for; however, it would be more appropriate if the system knew some underlying model of what the user is interested in. For example, say a user searches for "travel." If the user has previously demonstrated interest in a particular scenic and distant country, it would be more useful to the user if his search results considered these past interests and considered that his traveling inquiries might concern going to that particular country. Additionally, a user's interest may change over time. For this reason, we desire adaptively learning a user's interests.

In attempt to learn what users are interested in, many methods have been attempted, some of which have demonstrated to have improvements over normal search engines' results [1][3]. Continuing our work [1], we aim to implicitly learn a user's interests, as we deem the explicit learning approach to be insufficient. The primary weaknesses for explicit learning are that it's a simple, time-expensive, and not too adaptive technique [1]. Rather, we aim to adapt to a user's potentially changing interest, and to do so seamlessly without having to hassle the user. Furthermore, some learning methods generate a long-term and/or a short-term model representing one's interest; however, we desire having a continuum of long-term to short-term interests [1]. Moreover, others have attempted to model one's interest by actually clustering documents/web pages. Because the web is always growing, we view this as an elementary and naive approach. We deemed it more sufficient to model one's interests in the sense of having a categorical hierarchy, similar to the principle of a library catalogue when one searches for a book [1].

In this approach, we construct a User Interest Hierarchy (UIH), which is a model that contains words that are interesting to a user. This model is basically a tree structure where each node is a cluster of terms/words. The terms that comprise the tree are obtained from pages that the user has previously bookmarked, for we assume that a user bookmarks pages that he finds interesting. From these bookmarked pages, we extract all of the "meaningful" words and attempt to construct the UIH. Moreover, the general, commonly found terms will appear at the root of the structure, and as one traverses down the tree, more specific terms of interest are found. Obviously, web pages that concern terms that are found at lower levels in the tree are rarer and consequently are more likely to be interesting to the user. The reason for constructing this model is that it'll provide us with a hierarchy of interesting words for the sake of later comparing future pages so as to rank the degree to which they are interesting.

2 Problem Statement

The goal is to develop a system that implicitly and adaptively learns what a user is interested in. Having constructed a UIH model, our system can score and rank new web pages that a user visits. It is ideal to rank the web pages in a manner that is highly consistent with a user's actual interests. So, the problem is two-part: to create a system that can accurately learn a user's interests, and to appropriately score new pages such that the ranking complies with a user's interests. Past work [1] demonstrated good results such that it "outperformed" Google. (Our quantitative measuring criterion is later defined.) However, we now explore areas in our past work that may be naive or vulnerable to weaknesses. So, our problem includes finding better ways to learn a user's interests and to score web pages.

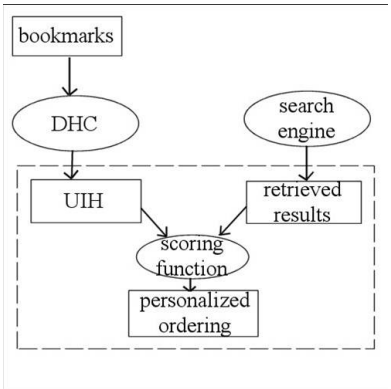

Figure 1: General Overview of System

In order to understand the problem of improving the current system, we must describe the current system (Figure 1). As previously mentioned, we constructed a tree structure model (called UIH) which allows us to measure the degree to which future pages are considered interesting. To construct this model that represents a user's interests, we use a user's provided bookmarks. Bookmarks were used because they have shown to be good representations of a user's interests [2]. Thus, to build our model we look at the content of the bookmarked web pages, where the content is ultimately the words within the page. However, not all words have the same degree of interest to the user. We implemented a Divisive Hierarchical Clustering (DHC) algorithm to find the relevance of each word. The input to the DHC will be the words from bookmarked web pages, and the output will be the UIH model of the user. First, we pre-process the input words by disregarding common trivial article and preposition words (i.e. a, by, for, then, and) because they are common amongst all Standard English sentences and suggest little towards what a user is interested in [4]. Next, we "stem" words so as to remove words that are much the same. For example, if the words "intelligence"

and "intelligent" are found, it would be not only wasteful to include both words in the model, but including both words could skew results. The stemmer function is responsible for removing these extraneous words and creating a stemmed-word that is inclusive to the possible variants. We are left with significant words that are input into the DHC algorithm:

Now that we know which words are meaningful to the user, we attempt to determine the degree to which each word is interesting to the user. We assert that words that occur closer together are more related to each other [1]. Our model desires having a structure that represents the groupings of similar words. Therefore, our DHC algorithm includes a similarity function that determines how similar two words are to each other. We used AEMI as our similarity function, and it basically computes a numerical probabilistic value that represents the likelihood of two terms appearing strictly together, considering the adverseness of one of the words appearing singularly without the other word. One option, though, is the window size in which to look when seeing if words occur together. For example, if the window size is the length of the entire web page, it may not be too meaningful for two words to occur together. Rather, it would be more useful and significant if our window size was the length of a paragraph, thus requiring terms to co-occur together within the same paragraph. For this case, it's likely that the co-occurring words are related to each other more so than in the case when the window size is that of the entire page.

The DHC algorithm first obtains the similarity values for every possible pair of terms, yielding us with a fully connected, undirected graph (Figure 2). The root node of our UIH tree model is a cluster of all words, as represnted in Figure 2. Subsequent children of a node will also be clusters, where each child cluster contains a subset of terms that were found within its parent's cluster. So, to determine the children clusters, we turn to the divisive part of

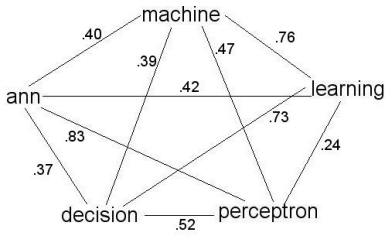

Figure 2: Fully connected weighted graph

our DHC algorithm: we eliminate some of the links from the root/parents cluster, yielding us with only the "strongest" pairs of similar words. However, how many links should we remove? We use a threshold function to determine how many links shall be removed. The removal of these pairs will directly affect and determine the shape of our tree. Currently, our basis is that we desire having a UIH for the same reason that a library catalogue is useful: it provides us with a categorical-like, grouping hierarchy. So, in keeping with this inherent idea, we believe that wide, short trees are ideal. This way we will have a breadth of groupings to be classified under a given topic. Yet, a problem arises: just because we believe this inherent tree shape naturally produces good results doesn't mean that it actually will. In theory, we have reason to believe the implementation will be sound. However, maybe a different tree shape will generally produce better results.

In addition, our current scoring algorithm partially bases its score on properties and elements within the constructed UIH. So, maybe adjustments and improvement could be made in each of these algorithms–the DHC and scoring function algorithms. Thus, the problem lies in attempting to explore other methods of constructing the UIH, or in scoring, so as to more accurately determine a user's degree of interest toward new pages.

3 Approach

One area mentioned that we shall investigate is our scoring method: A page's score is merely the summation of the score of each term that is found within both the web page and the user's UIH. Each term is scored by 3 main characteristics: (1) the probability that another word occurs at the depth D, where depth D is the lowest depth that the current term is found within the user's UIH; (2) the probability that another term within the web page has the same emphasis E, where emphasis E is either bold, italicized, or title'd word; (3) the probability that another word within the web page has the same frequency F, where frequency F is the merely the number of times that the given term occurs within the web page. For each of these cases, the lower the probability is, the more important and non-coincidence the term is. So, using information theory, the formula is:

$$S_t = -w_1 log(P(D)) - w_2 log(P(E)) - w_3 log(P(F))$$

Note: For our unweighted scoring algorithm, all weights are 0. For our weighted scoring algorithm, the weights are w1=.5, w2=.25, w3=.25. We weight P(D) more heavily due to our basis/assumption that this characteristic heavily suggests importance for the given term.

Having analyzed the contribution that each characteristic provides, it was apparent that the frequency characteristic strongly dominates the effect on the total score of a web page. However, we desire each term having relatively equal contribution, at least for our unweighted scoring algorithm. This unbalanced contribution was caused because many words appear a unique number of times within a web page. For example, not many words appear strictly 17 times, as many don't appear 16 times. However, these number of occurrences are similar, so we decided to group number of occurrences into 10 equally-sized "bins." This yielded a much more balanced contribution amongst the 3 characteristics that score a term.

We mentioned that a page's overall score is merely the summation of all of its terms' scores, where each term must be found within the UIH. However, this lends itself to be naturally biased towards web pages of larger size because more words will have a score. This may not seem problematic, for if more "interesting" words are found within a document, the page should reflect such and have a higher score. Yet, it is possible that many words within the longer web pages may not necessarily be too interesting; rather, the page may consist of many barely interesting terms and still consequently achieve a higher score due to its sheer number of terms. So, it's a tough situation to balance how much a term's score should count, and we question if should simply take a summation of all the found words. One approach we explored is to take the average score of all the scored terms. Results were worse, so we considered another idea. Example: 2 web pages each have 10 terms that are to be scored. The scores are identical, and thus the pages have equal score. However, what if one of the web pages also contains many more terms that aren't found within the UIH. Surely these no-scored, non-interesting terms should negatively impact the score of the page. So, we also scored terms based on the (total score of terms / total number of words that occur within the webpage). So, this is how our system scores a web page. Having scored all web pages, we can rank the pages in respect to how high their scores are. However, Google does a good job and provides us with information, so it would be useful to consider Google's ranking. So, our final ranking is a basic merger that equally weighs our ranking with Google's ranking.

4 Evaluation

11 volunteer users have previously been asked to surf the web for a rather extended period of time. In particular, users first supplied at least 50 of their previously bookmarked web pages. Next, they provided 2 search terms with which to query Google's search engine. Users then exhaustively visited each of the first 100 returned web pages for each search term, while ranking each web page in respect to the degree they found the page interesting. The optional degrees were "good," "fair," and "poor." Additionally, users specified if each page was "potentially interesting" or not. After frequenting all 200 search queried web pages, analysis could then be performed.

Each of the 200 pages that a user visited was run through our system and scored. Since the user specified if he is actually interested in each page, we can determine how correct we were in our predictions. So, currently, the analysis would include looking at different subsets of the web pages. For example, instead of looking at the first 100 Google-returned web pages, we may look at only the first 10 Google-returned web pages. In suit, we would then look at the 10 web pages that we scored as being most interesting, based on our scoring. We aim to include in our returned subsets a high number of web pages that the user specified as being "interesting." We use two measuring criterion:

$$Precision = \frac{\#\ of\ \text{“}interesting\text{”}\ pages\ returned}{\#\ of\ pages\ returned}$$

$$Recall = \frac{\#\ of\ \text{“}interesting\text{”}\ pages\ returned}{total\ \#\ of\ \text{“}interesting\text{”}\ pages}$$

So, with these two criterions, we look at varying sizes of subsets of returned web pages, comparing Google's ranked pages with our ranked web pages. Note: we look at our results from both of our aforementioned algorithms: Weighted Score and Unweighted Score. For a baseline, we also look at the results from a random ranking system. We implement this experiment of results for (1) web pages in respect to their degree of being interesting (good, fair, or poor) and (2) web pages in respect to being marked as either potentially interesting or not.

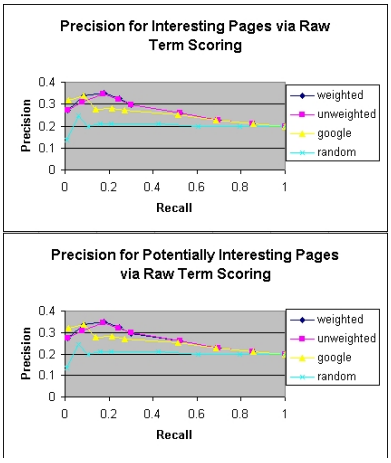

Figure 3: Precision vs Recall Graphs

Note, for our implementation, we used the discussed idea of frequency bins as a characteristic for a term's score. Additionally, we computed results having scored a web page based on (1) raw score of the summation of a web page's scored terms, and (2) the average scored term in respect to the total number of words in a web page. (Although the raw scoring technique always performs better.) Also, we are still merging our ranking with Google's ranking, thus providing some help towards outperforming Google.

The resulting precision v.s. recall graphs (Figure 3) provide us with a good idea as to how well our system performed. However, for this, we considered all web pages to be either "interesting" or "not interesting." Yet, the user had a 3-level evaluation scheme, so it's more appropriate to consider all 3 degrees of interest when measuring how well our system ranked

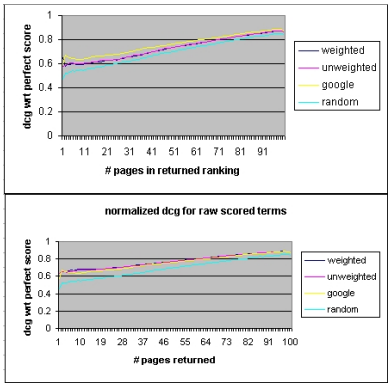

Figure 4: Normalized DCG Results

pages. So, we used a Discounted Cumulative Gain (DCG) to consider the 3 degrees of interest by which a user evaluated [3]. "Good" pages are marked as 3, "fair" pages were marked as 2, and "poor" pages as 1.

$$DCG(1) = G(1)$$

$$DCG(\forall i! = 1) = DCG(i-1) + \log(i)$$

So, for a given index within the ranking, the one with the higher DCG value signifies a better ranking scheme. Note that the algorithm logarithmically decreases the contribution that each subsequent page has on the overall score of the ranking at a particular index. This is because having correct and good ordering at the top of our ranks is most important than the ranking towards the bottom. The results with respect to the # of returned pages are shown in Figure 4.

We also report the DCG results in respect to the entire ranking of returned pages, per algorithm (Figure 5.

DCG Values for Raw Scoring of Terms

Weighted	**0.8911017736306949**
Unweighted	**0.889322829124885**
Google	0.8869919634085389
Random	0.8565146526388138

DCG Values for Average Scoring of Terms

Weighted	0.8704960629227444
Unweighted	0.8715606317478826
Google	**0.8869027404970705**
Random	0.8565271913788266

Figure 5: Normalized DCG Results for all pages

5 Conclusion

Overall, it's apparent that our system can help improve Google's ranking. Keep in mind that we incorporate Google's ranking in deciding our ranking. Despite any seeming success, there are many factors in our system that may be adjusted.

For example, we mentioned that we assume short, wide trees are the best for building our UIH model. However, maybe long, narrow tree model will provide better results, for this would more sharply contrast and distinguish between words that occur at different depths.

Additionally, we mentioned the battle of finding an appropriate way to score a page in respect to how many terms are contained. We don't want to be biased towards documents that are merely larger in size, yet simply taking the average score of terms seems to have worse results than the raw summation score. Ultimately, we want to score a web page based on its terms without allowing many insignificant words to skew the results. To solve this, and since we have already built a UIH, I believe it's appropriate for the lowest depth within the UIH that the given term occurs to be the overall determining factor on how much each term should contribute to the web page's total score.

Similarly, another approach to scoring a web page

is to look at the actual occurrences of a given term. We currently consider this in our frequency probability characteristic. However, we do not distinquish between terms that have the same frequency. For example, if a term occurs 5 times in a web page and occurs throughout the entire document, it is likely more interesting to the user than a web page where the term appears 5 times in one small concentrated area; the page with the larger span is most likely to concern that given term. Similarly, we may look at the distribution within that range of occurring for the sake of ensuring the term is well spanned within the seeming range.

As for our weighted scoring algorithm, we have arbitrary weights for each scoring characteristic. The lowest-depth-within-UIH characteristic has the most severe weight, but these weights are arbitrarily chosen based on past experiments and reasoning. However, it might be more appropriate to learn weight values from the bookmarked web pages. Using bookmarked pages as our guide, a simple perceptron could be implemented, or merely finding average contributions of the each characteristic might be appropriate.

A different approach could involve computing an overall significance-value for each term within the UIH, for ultimately, it appears that the depth within the UIH is the most important characteristic in determining how interested a user is in a given term. So, we could look at the aforementioned characteristics of word emphasis and frequency within the *bookmarked* web pages, and factor this in, along with the UIH depth, so as to assign each term an overall interesting-value. This could have the advantage of distinguishing between words that appear at the same depth in the UIH.

Also, it has been shown [3] that merely looking at a snippet of a web page, and not the entire document, may provide better results. So, what snippet of the web page shall we look at? It might be appropriate to look at only a given window of words, where each window is centered at a given search query for which the user searched. So, only terms that are within our appropriate window are scored. Thus, insignificant words that are outside our range of interest are not scored. This opens up ideas of how large should our window be? I suggest weighting the score of each term based on the distance it is from the given query term that is being scored. The contribution of each scored term may take on a bell-shape curve, where the peak is the query term. Another idea is to only look at the top given percentage of words within a given web page. The motivation for this is that the top of a web page often contains an overview of what the page concerns. A disadvantage of this is that maybe this top percentage of the page may have no words that we deem "interesting," and may consequently, inappropriately score a page as having no interest value.

Furthermore, we currently merge Google's ranking with our ranking by weighting each ranking's contribution evenly. It might be more useful to use a more advanced scheme that considers Kendall-Tau distance [3]. Kendall-Tau distance, also known as Bubble Sort distance, is defined as the number of pair-wise disagreements between two lists. Values are normalized by the maximal number of disagreements. For example, two identical lists will have a Kendall-Tau distance of 0. Lists that completely reversed will have a Kendall-Tau distance of 1.

In conclusion, results suggest improvement over Google's ranking, yet many ideas should be explored so as to improve the system.

References

[1] Hyoung R. Kim and Philip K. Chan Personalized Ranking of Search Results with Learned

User Interest Hierarchies from Bookmarks. *WEBKDD Workshop, SIGKDD Conf.*, 2005.

[2] Hyoung R. Kim and Philip K. Chan Learning Implicit User Interest Hierarchy for Context in Personalization. Proc. Intl. Conf. on Intelligent User Interfaces, pp. 101-108, Melbourne, FL 32901, 2003.

[3] Teevan, Jaime, et al. Personalizing Search via Automated Analysis of Interests and Activities *28th Annual International ACM SIGIR Conference, ACM Press*,2005.

[4] Frakes, W.B., and Baeza-Yates, R. Information Retrieval: Data Structures and Algorithms, Prentice Hall, 1992