Personalizing Entity Detection and Recommendation with a Fusion of Web Log Mining Techniques*

Kathleen Tsoukalas[1] Bin Zhou[1] Jian Pei[1] Davor Cubranic[2]
[1] Simon Fraser University, Canada [2] Business Objects, Canada
{kjtsouka, bzhou, jpei}@cs.sfu.ca, davor.cubranic@sap.com

ABSTRACT

Given the proliferation of technology sites and the growing diversity of their readership, readers are more and more likely to encounter specialized language and terminology that they may lack the sufficient background to understand. Such sites may lose readership and the experience of readers may be impacted negatively if readers cannot quickly and easily find information about terms they wish to learn more about. We developed a system using a fusion of web log mining techniques that extracts, identifies, and recommends personalized terms to readers by utilizing information found in individual and global web query logs. In addition, the system presents relevant information related to these terms inline with the text. Our system outperforms some other related systems developed in the literature with special regard to usability.

1. INTRODUCTION

News articles, blogs, and other online media containing specialized language or highly technical terms are becoming increasingly available, especially with the proliferation of technology sites such as Engadget (`http://www.engadget.com`), Gizmodo (`http://gizmodo.com`), and Ars Technica (`http://www.arstechnica.com`). In addition, in accordance with the Web 2.0 philosophy, these sites leverage social networking to encourage user participation through the use of user feedback in the form of comments and recommendations. A solid understanding of the content of such sites is fundamental to participation in these communities. However, given the growing diversity of internet users, not all have a sufficient background to understand technical content. Although users are often highly motivated by a topic of a site or article, unknown terminology can lead to confusion and frustration. These readers will resort to search engines or sites such as Wikipedia (`http://wikipedia.org`) to find definitions and explanations of concepts, leading away from the original task and causing more frustration.

Usability studies have shown that one of the chief concerns expressed by users who read online articles is the ability to find information about technical terminology quickly. Helping users quickly learn relevant information about terms they do not understand will not only improve the user experience of the site, but also increase reader loyalty; if this information is provided inline or at least on the same page as the article currently being read, the reader may be discouraged from switching to search engines or Wikipedia, preventing navigation away from the site. This has implications for business and educational sites, too. Consumers who can readily find product information on a site they are reading may be more inclined to purchase those products from that site and return to it in future. If shoppers can quickly find the information they need to understand a product review's technical terminology, they are more likely to remain at that site and even make a purchase there. Similarly, students conducting research on pages that explain difficult terminology without requiring extensive searches will be able to learn more quickly and integrate new knowledge with less effort. Students may be able to complete a higher quality report more efficiently if the information needed to understand what they are reading is more readily available to them.

These challenges are addressed by our system, which determines and extracts entities a given user may find interesting or want to know more about and then provides related information on those entities inline with the article being read. Here, we define an *entity* as a word or meaningful phrase of no more than three words. The system mines individual and global query logs for popular concepts. Entities related to those concepts are identified in the article being read, such that for different users different entities may be recommended. Rather than the obtrusive pop-up windows of Intellitxt (`http://www.vibrantmedia.com`) which appear when an entity is hovered over, our system presents information only when a user clicks on a recommended entity. This reduces the frustration users experience when pop-up windows appear in case an entity is hovered-over unintentionally, and also allows us to gather important information on the interests of users.

The rest of the paper is organized as follows. We review the background research related to our system in Section 2.

*This work was supported in part by an NSERC Undergraduate Student Research Award, an NSERC Discovery grant, and an NSERC Discovery Accelerator Supplements grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

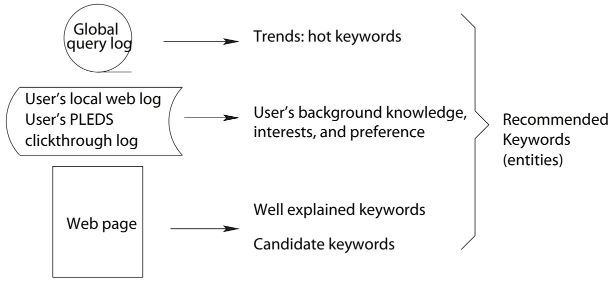

Figure 1: The architecture of the system.

We then provide an overview of the technical strengths of our system in Section 3. We report the experimental results in Section 4, and conclude the paper in Section 5.

2. RELATED WORK

There are several existing systems related to our work. First, Wikipedia provides a rudimentary entity recommendation system whereby certain terms found on a given Wikipedia page are labeled as hyperlinks. However, Wikipedia is not personalized, since all users see the same links on a given page, some users will see links for terms they already understand. In addition, this system forces the user to navigate away from their current page in order to find the related information.

Improvements are presented by various advertisement systems, such as Google's Gmail (`http://mail.google.com`) and AdSense (`http://www.google.com/adsense`). These systems have limited personalization that is obtained by mining related sources such as users' emails. However, they lack the ability to present the information inline. Another group of systems such as Kontera (`http://www.kontera.com/demo.aspx`) or Intellitxt have inline labeling, but fail to include personalization. Lastly, a new system, Contextual Shortcuts [3], provides both inline labeling and limited personalization.

We improve the personalization aspect of such systems and introduce the ability to adapt to a user's changing interests, addressing the main interaction challenges presented. We combine the strengths of the previous systems in several entity detection heuristics and incorporate natural language processing (NLP) techniques.

3. AN OVERVIEW OF THE SYSTEM

Our system aims at identifying a small number of entities within the text of the page a user is reading. These entities should be chosen such that the user is interested in learning more about them. We consider first, the text of the page (are the keywords explained well?), second, the user's background knowledge (does the user already understand these keywords?), and third, the topic trends (is the topic of a given keyword currently trendy?). Our system combines the results of each of the above factors to determine which entities should be labeled inline with the text. If the user wishes to learn more about a particular entity, they can click on it and the information will appear in a pop-up window. The information presented consists of a summary extracted from the results of a search query of the entity and user's interest.

Our system's conceptual architecture is shown in Figure 1. Next, we describe techniques for mining information to provide personalized entity detection.

3.1 Mining Trends in Global Query Logs

A keyword that has been searched for recently by a large population of users is also quite likely to be interesting to an individual [3]. This leads to a heuristic for currently trendy topics. We mine a window W in a global web query log, where W is taken to be the queries found in the log from the last 30 days. We then identify candidate entities in the query log and find each entity's frequency. We take this frequency to represent the entity's popularity in the current time window.

3.1.1 Pre-computing Frequencies

We use a co-occurrence frequency calculation in the query log to identify a set of candidate entities. First, we must find the frequency of each word and the co-occurrence value of each two- and three-word phrase in the query log. These values are pre-computed in an offline step due to the expensive nature of such calculations for large logs. The precomputation is as follows: first, we find the frequency of each valid word in the global query log in a single scan of the entire log. Valid words are those that are not found in a stop list. Also, words consisting solely of punctuation are removed. Each valid word and its corresponding frequency are then stored in a background database. We scan the log again to compute the frequency of each pair of adjacent words in the log, and once again for contiguous three word phrases.

The co-occurrence frequencies are important in determining the meaningfulness of an entity. For each two-word phrase $\langle w_1, w_2\rangle$ in the our table of two-word phrases and their frequencies, we calculate the co-occurrence frequency $CoFreq(\langle w_1, w_2\rangle) = \frac{f(\langle w_1, w_2\rangle)}{f(w_1)f(w_2)}$, where $f(\langle w_1, w_2\rangle)$, $f(w_1)$ and $f(w_2)$ are the frequency of the two-word phrase $\langle w_1, w_2\rangle$, the frequencies of one-word phrase w_1 and w_2, respectively, in the current window W of the global query log.

We ensure that each normalized co-occurrence passes a certain threshold value θ in order to keep only those co-occurrences which are likely to be meaningful entities.

3.1.2 Extracting Entities from the Current Web Page

Extracting entities and their co-occurrences from web logs may suggest entities of interest to users, but it is also necessary to consider the text being read by a user and the entities contained therein. Thus, we analyze words found in each sentence of the document D being read. We then determine if a word segment $\langle w_1, w_2\rangle$ belongs to a candidate entity, and incorporate "concept extension" [3] to determine entities of a maximum length of three words. For example, if the phrase "Simon Fraser University" has a total frequency similar to that of the length-2 subsequences "Simon Fraser" and "Fraser University", we extend the entity to contain all three words. If we determine that the three-word phrase is not an entity, we then consider its constituent two-word phrases and check if they themselves are meaningful. We do not consider four word phrases since they are statistically rarely likely to occur.

Several previous studies [3] show that using word co-occurrences alone may not be very accurate, which is why we apply this method to the document being read as well as our query log to find meaningful entities. Only terms found in the document that also appear frequently in the query log will be considered as candidate entities. Our use of "concept extension" is biased towards longer and more specific entities as well, which allows us to obtain meaningful results when combined with our co-occurrence mining technique.

3.2 Mining Local Logs

A user's background knowledge is also important in determining a user's interest. Our model takes this into account by considering whether an entity has already appeared in the text or if a user has recently clicked on it. In these cases, it is not likely that the user will want to learn about the entity again, so our system is less likely to label subsequent appearances. We mine two data sources to determine the background interest of users, a user's local web log as well as her/his click-through history.

3.2.1 Mining Local Web Log Data

Local web logs contain information on, for an entity, whether a user has recently queried the entity or some other entities highly related. Our system uses this information in two ways.

First, we consider whether an entity has recently been queried by a user. If so, the user may not be interested in the entity in the near future. We model this by the query freshness of the entity. If an entity e was queried at time instants $t_1, \ldots, t_m$, the *query freshness* of e is defined as $QueryFresh(e) = 1 - \sum_{i=1}^{m} \alpha^{t-t_i}$, where t is the current time instant and α is a decaying factor between 0 and 1. The larger the query freshness, the more interesting an entity is.

Second, we consider whether a user has an interest in the category of a particular entity. That is, if many entities in the same category of e were queried before, e may also be of interest to the user. To model the entity ontology, we use the concept of *sense* arisen from WordNet [4], which refers to the meaning of a word it belongs to. Each word may have several senses, with each belonging to a different *synset*, which in turn is a group of synonyms. The senses in WordNet have a taxonomy structure.

The Adapted Lesk Algorithm [1] disambiguates between words by comparing a target word w_i to surrounding words using a measure of semantic similarity. We thus finding the most appropriate sense and part of speech for w_i. We use the WordNet [4] semantic lexicon and its .NET library, WordNet.NET (`http://opensource.ebswift.com/`) for the implementation.

We find and maintain an *interest vector* for each user, which contains the most popular *senses* found in the user's local log, as well as the frequencies of the senses. To find the most popular senses, we find the most likely sense for each word in each query of the local log. The k most frequent senses are included in the user's interest vector. The interest vector for a user u_i is $V(u_i) = \langle (sen_1, freq_1), \ldots, (sen_j, freq_j), \ldots \rangle$, where sen_j represents a sense and $freq_j$ represents the frequency of sense sen_j.

The *interestscore* of an entity e_j for a given user u_i is determined as follows. If e_j's total number of senses is n, we assume each sense of e_j has probability $\frac{1}{n}$. Then, if e_j's senses $sen_{i_1}, \ldots, sen_{i_t}$ also appear in the user's interest vector, $V(u_i)$, we calculate the interest score of e_j as $IS_{u_i}(e_j) = \sum_{k=1}^{t} \frac{1}{n \times freq_{u_i}(sen(i_k))}$, which measures the likelihood that an entity will be interesting to the user.

3.2.2 Mining System Click-through History

Our system relies on click-through data as an implicit user feedback. That is, if a user has previously clicked an entity labeled by our system, then she/he is less likely to click it again in the near future. On the other hand, if the same entity labeled by our system is never clicked, then it is also unlikely to be clicked on in the near future.

We model these ideas by mining a user's historical click-through data. Each time a user clicks on an entity, we record the data and compute the *click freshness* by incorporating a decaying factor. In this way, the longer ago an entity has been clicked, the lower the *click freshness* is. Thus entities with a higher *click freshness* score are less likely to be clicked again than entities with a lower score.

Click freshness is calculated as follows: if an entity e was clicked at time instants $t_1, \ldots, t_m$, the *click freshness* of e is defined as $ClickFresh(e) = 1 - \sum_{i=1}^{m} \alpha^{t-t_i}$, where t is the current time instant and α is a decaying factor between 0 and 1. The lower the *click freshness*, the more interesting an entity is assumed to be.

3.3 Mining the Current Web Page

If an entity e on the current page is well-explained, then it is unlikely the user will need to learn more about it by clicking on it. Thus we propose an *explanative score* to address whether an entity is well explained or not. Given an entity e, the explanative score of e is computed by checking all entities surrounding e (i.e., in a small window centered at e) for their semantic relatedness to e. To measure semantic relatedness, again we base our algorithm on the Adapted Lesk Algorithm [2]. Let $e_1, \ldots, e_n$ be the set of entities surrounding e in a window. Then, the explanative score of e is calculated as $ES(e) = \frac{\sum_{i=1}^{n} \frac{1}{dist(e,e_i)}}{n}$, where $dist(e, e_i)$ is the semantic distance between e and e_i, as used in [7]. The larger the *explanative score*, the better explained the entity.

Since each entity determined via the co-occurrence method must be compared to each of its surrounding entities when determining *explanative score*, efficiency is a challenge. To keep it efficient, we limit the window size. This reduces computational time, but also reduces the accuracy of the part-of-speech tagging. Therefore, we introduce some precomputation techniques to mitigate the problem. The idea is to keep a list of pairs of entities that have previously been compared, along with their corresponding similarity scores. Then, during the *explanative score* calculation, we check if the pair of entities in question is already in this list. If it is, we can simply use the pre-computed score. Otherwise, we assign the pair a score of 0 and store it in the list. This models our assumption that an entity pair is not very common if it has not been previously encountered, which indicates that the pair of entities are not highly semantically related. However, if we encounter the pair a second time, the pair of entities are no longer assumed to be uncommon and the previous score of 0 is replaced with the actual calculated score. Now the score has been calculated, and will be used in subsequent encounters of this pair. This negates the need to compute the semantic similarity every time.

An entity may appear multiple times in a single page. In this case, we find the largest explanative score among the multiple occurrences and use that in our overall model. Thus, if an entity is quite well-explained somewhere on the page, there is a decreased need to label and explain it again

System	Precision	Recall	F-Measure
Wikipedia	0.251777922	0.405978836	0.310803368
PLEDS (NP)	0.101616162	0.192486772	0.133013069
PLEDS (P)	0.531689113	0.645704949	0.58317653

Table 1: The precision and the recall for the system comparison (NP: Without Personalization. P: With Personalization).

since the reader has already encountered its explanation elsewhere on the same page.

3.4 Fusing the Mining Results

Our system combines mining techniques for several sources, including the global query log, the local log, and the click-through data, as well as the text found in the web page being read, to estimate the probability that an entity e will be clicked on by a user. Thus we have five factors x_1, x_2, x_3, x_4 and x_5, where x_1 is the explanative score (Section 3.3), x_2 is the global frequency (Section 3.1.2), x_3 is the query freshness (Section 3.2.1), x_4 is the click freshness (Section 3.2.2), and x_5 is an interestingness score (Section 3.2.1) computed using the interest vector. We then use logistic regression to recommend a list of entities to be labeled.

In order to learn a logistic regression model, we take a training set of data. The formula of the logistic regression is $ln\frac{p}{1-p} = \beta_0 + \sum_{i=1}^{5} \beta_i \cdot x_i$, where $\beta_0, \dots, \beta_5$ are the coefficients. In other words, the probability p that an entity will be clicked on can be measured as $p = \frac{1}{1+e^{-(\beta_0+\sum_{i=1}^{5}\beta_i \cdot x_i)}}$. The "Newton-Raphson" method [5] is used to learn the regression coefficients.

Once the model has been trained, the system uses the model to retrieve entities in each page based on whether they have probabilities above a threshold value. The threshold may be tuned by the user to adjust the degree of aggression with which the system detects and displays entities. Entities above the threshold will be labeled.

The full technical details are provided in [6].

4. EXPERIMENTAL RESULTS

A prototype of the system, called PLEDS, was implemented in Microsoft .NET using C$\sharp$. Microsoft SQL Server 2000 was used as the background database management system.

In our user studies, a large, real web search query log from AOL (`http://www.aol.com/`) was used. To increase performance, it was reduced in size through data cleaning. At the start of user testing, the global query log contained $97,471$ tuples (however the size increases as the system is used). On average, each user had 140 tuples in their local web query log, with 696 users initially in the system. This initial global query log results in $43,014$ distinct co-occurrence phrases and this is reduced to $4,287$ distinct co-occurrence phrases once they are normalized and filtered by a threshold as described above.

We initially made a comparison of our system with and without personalization. The results of entity recommendation before and after the training period are shown in Table 1. Here `PLEDS (P)` refers to the performance of PLEDS after the 15 minute training period. Once these results were collected, we analyzed the same page with the user logged out of the system, which removes the effect of personalization. This second analysis provided the results for `PLEDS (NP)`. The `Wikipedia` results were found by analyzing the entities shown as links in a Wikipedia page viewed in a normal web browser.

After the training period, PLEDS provides good precision and recall scores. Once personalization has been activated there is remarkable improvement in these scores. During each user session, PLEDS only had 15 minutes to adapt to a participant's preference, and it is conceivable that with longer length of use, the results would improve further. In addition, precision scores may have been affected by the limited length of user-specific query logs, since not many queries can be executed in 15 minutes (there were on average only 10 tuples per user). As the local query log becomes larger, PLEDS becomes more accurate. Thus, giving users more time with the system may lead to better results.

We also made a comparison between PLEDS and Wikipedia. In its initial state (without personalization), Wikipedia's link labeling outperforms PLEDS with respect to precision and recall. However, as PLEDS adapts to the user's preference, the performance of PLEDS significantly increases and is better than Wikipedia. An interesting observation is that PLEDS allows users to limit the size of the resulting set of entities to 10% of all entities present. During the study, participants often switched from higher entity percentage settings to lower ones, indicating that too many entities are overwhelming and impact the reading experience negatively.

5. CONCLUSIONS

Our techniques allow us to provide personalized, meaningful entity recommendations in text, which help readers to quickly and easily find information required for an adequate understanding of the text. We create a more comprehensive model of user behavior by including new measures, including *Interest Score*, *Explanative Score*, *Query Freshness*, and *Click Freshness*, as well as more traditional *Frequency* measures. This results in improved performance over related systems such as Wikipedia. By learning a user's interest, PLEDS recommends and retrieves more relevant entities for specific users.

6. REFERENCES

[1] S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing'02*.

[2] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI'03*.

[3] V. von Brzeski *et al.* Leveraging context in user-centric entity detection systems. In *CIKM'07*.

[4] C. Fellbaum, editor. *WordNet: an electronic lexical database.* MIT Press, 1998.

[5] T. Hastie *et al. The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001.

[6] K. Tsoukalas *et al.* PLEDS: A personalized entity detection system based on web log mining Techniques. (Invited Paper) In *WAIM'08*.

[7] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *ACL'94*.