PLEDS: A Personalized Entity Detection System Based on Web Log Mining Techniques

Kathleen Tsoukalas #1, Bin Zhou #2, Jian Pei #3, Davor Cubranic *4

#*School of Computing Science, Simon Fraser University*
Burnaby, B.C., Canada
1kjtsouka@cs.sfu.ca
2bzhou@cs.sfu.ca
3jpei@cs.sfu.ca

**Business Objects*
Vancouver, B.C., Canada
4dcubranic@businessobjects.com

***Abstract*— With the expansion of the internet, many specialized, high-profile sites have become available that bring very technical subject matter to readers with non-technical backgrounds. While the theme of these sites may be of interest to these readers, the posts themselves may contain terms that non-experts may be unfamiliar with and may wish to know more about. We developed PLEDS, a personalized entity detection system which identifies interesting entities and provides related information for individual users by mining web logs and query logs. The experimental results of a systemic user study shows that with PLEDS's aid, users can experience the benefits of an enriched internet surfing experience.**

I. Introduction

With the rapid expansion of the internet, many specialized, high-profile sites have become available that bring highly technical subject matter to readers with non-technical backgrounds. For example, Gizmodo[1], Engadget[2], and Boing Boing[3] are all popular user-driven sites that present articles containing terms a non-technical reader might not be familiar with. As such, although readers are interested in the themes of such sites, they may get lost in such overly technical terminology, which may result in decreased readership for the site and a negative experience for the reader. For example, one post on digital cameras[4] discusses white balance, but the style of the camera is such that it is likely to be used by more amateur users who may be unfamiliar with the term. They may spend more time researching white balance on other sites, or may feel frustrated by the article and be less likely to return in future. In either case, the user is drawn away from the website and is left with a negative experience overall.

It is important to provide services to readers so that they can not only find additional information about more technical terms, but find it quickly as well. In fact, usability studies have shown that this is one of the chief concerns users express when reading articles online [8]. A naïve solution is to create hyperlinks for terms that contain more detailed information on a separate page, and thus allow users to navigate to those pages via the hyperlinks. This idea is exhibited by Wikipedia[5] in particular, but is problematic for several reasons. Less experienced users are often reluctant to navigate through hyperlinks as they worry about "getting lost" [8] and overlooking the original task. Also, this navigation away from the article presents an interruption in the flow of reading [9], resulting in a negative experience for the reader. Finally, using hyperlinks in this way presents the same information to all users. Some users may find that too many terms are tagged, while others find that not enough are tagged. In the former case, the extra information is not only unnecessary but, depending on the manner of display, excessive tagging may be distracting. In the latter case, frustration may arise from not being able to quickly find the information required. As such, it is necessary to develop a tool that is not only inline, but also personalized for each of its users.

To address the above challenges, we developed PLEDS, a personalized entity detection system which identifies interesting entities and provides related information inline. Here, an *entity* is a keyword or a meaningful short sequence of keywords. PLEDS mines individual and global query logs to find popular concepts, and tags entities related to those concepts, thus finding different entities for each user that they are likely to be interesting to the user. Information is presented in a small pop-up window only when a user clicks on a tagged entity, which solves the problem of numerous pop-up windows appearing as the user unintentionally moves their mouse across the screen.

The paper is organized as follows. We review the background research related to PLEDS (Section II) before providing an overview of the major technical strengths in PLEDS (Section III). We then describe the experiments and user studies we performed on PLEDS and the results of those studies (Section IV), and conclude with a discussion of the implications and future directions of this work (Section V).

II. Related Work

Our work is related to entity detection systems and various web log mining techniques. We briefly review some represen-

[1]http://gizmodo.com
[2]http://www.engadget.com
[3]http://www.boingboing.net
[4]http://www.engadget.com/2008/02/06/fujifilms-z10fd-and-z100fd-cameras-get-totally-rockin-firmware/
[5]http://www.wikipedia.com

tative work here.

A. Entity Detection Systems

Building on the approach provided by Wikipedia, several entity detection systems have been developed to address the challenges mentioned in Section I, with varying degrees of success. These systems and techniques can be categorized into three types. The first group of systems, which includes Google's Gmail[6] and AdSense[7], has some degree of personalization but does not present information for entities inline. The second group, including Vibrant Media's Intellitxt[8] and Kontera[9], does not include personalization, but does have inline entity tagging. Finally, a recently proposed system Contextual Shortcuts in [10] attempts some limited form of personalization as well as inline tagging capability.

An example of the first type of systems is Google's Gmail. Gmail tries to match a user's interest with some predefined topics by extracting some keywords from an email or a set of emails being viewed. It then presents advertisements related to those topic keywords, but does not present this information inline. As such, it requires an interruption in the flow of a user's reading. That is, a user has to leave the main paragraph of text to see the related information in a separate pane. This may impact negatively on the reader's experience, as indicated in [9]. Also, it only presents information for a limited number of terms and may not accurately reflect the complete interest of the user. Google's AdSense works in a similar manner.

The second group includes systems such as IntelliTxt, which mine the text in the web page currently being viewed by a user. Such a system extracts some keywords based only on the information found in the title of the page and the main body of text, thus relying only on the text within the page being read, and as such totally neglecting users' interest. The system is not personalized; this method results in the same entities being tagged regardless of which user is currently reading the page. In addition, related information for each entity is presented in the form of popups that appear when the links are hovered over. This information is thus presented inline, but implementing the tags in the form of popups has resulted in user dissatisfaction.

Most recently, a novel system called Contextual Shortcuts is presented in [10] as the representative of the third group in our categorization. The system uses global query logs to find topics that are frequently queried by a population of users, and uses that information to achieve more interesting entity extraction. In other words, it attempts to utilize user preferences. However, because it accesses information about a population as a whole, it presents the same information to all users, even though an individual's interest, knowledge, and background may lead them to find different entities in the text interesting.

B. Our System versus Previous Work

PLEDS builds on the work of previous entity detection systems by combining their strengths and solving some of the technical and interaction challenges they present. The largest improvement is due to the ability of PLEDS to adapt to each user and thus present information that will be of unique interest to them. We propose several effective heuristics to mine various useful information for entities in the document. We also incorporate natural language processing (NLP) techniques, such as using a taxonomy to measure similarity between words and phrases. Finally, we improve on the presentation method of the information about each entity so as to reduce user annoyance and frustration.

[6]`http://mail.google.com`
[7]`http://www.google.com/adsense`
[8]`http://www.vibrantmedia.com`
[9]`http://www.kontera.com/demo.aspx`

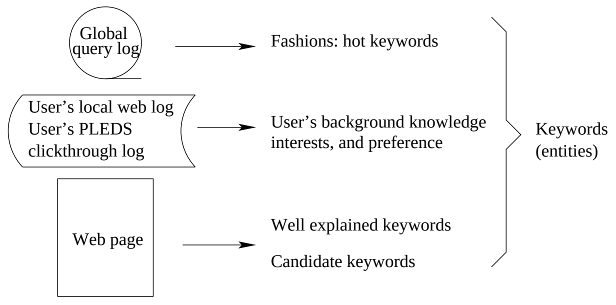

Fig. 1. The architecture of PLEDS.

III. PLEDS: An Overview

The personalized entity detection task in PLEDS involves identifying a small number of keywords (i.e., entities) from the page currently being read by a user so that the user may likely want to know the meaning of those keywords. A user's interest in keywords depends on three factors: the topic trends (what topics are currently trendy?), the user's background knowledge (what might the user already know?), and the content of the web page being read (what keywords are explained well on this page?). Correspondingly, PLEDS exploits four types of data to derive the information.

To find currently trendy topics, PLEDS uses the mining results from a global query log in a search engine to identify the currently popular keywords. To understand the user's background knowledge, PLEDS mines the individual user's web log to find the user's personal interest as well as what the user may already read and thus know. Moreover, PLEDS analyzes the individual user's click history while they use PLEDS to learn that user's preference and what keywords they have recently learned more about. To capture the keywords that are likely well explained in the web page being read, PLEDS scans the page. Integrating the above four different types of data and enabled by mining those data, the keywords identified by PLEDS are highly personalized.

Once the personalized entities are identified, PLEDS takes into account the user's interest and provides related information inline. The information is a summary extracted from the top results of a search query using the entity and user's interest.

The conceptual system architecture of PLEDS is shown in Figure 1. In the rest of this section, we will explain how the four kinds of data are mined and used in PLEDS for personalized entity detection.

A. Mining Trends in Global Query Logs

As suggested in [10], a keyword that has been searched for by many people recently is also very likely to be interesting to an individual user. PLEDS also uses this heuristic to model currently popular topics. By mining a current window W (e.g., the queries in the last 30 days) in a global web query log (e.g., one in a large search engine), PLEDS identifies candidate personalized entities in the query log and computes the global frequency of each entity as a measure of its popularity in the current time window.

1) Pre-computing Frequencies: A large query log may contain noisy information; there is some previous work focusing on web log cleaning [11]. As a pre-processing step, we need to conduct some procedures necessary for data cleaning. First, we remove unusual symbols and characters from the log. We then use a co-occurrence frequency calculation in the query log to identify a set of candidate entities.

In order to compute this co-occurrence frequency, it is first necessary to determine the frequencies of individual words as well as the co-occurrence values of two-word phrases in the query log. Since determining the frequency and co-occurrence can be very expensive if the query logs are large, we pre-compute those values. Those valid words and their corresponding frequencies are maintained in a table on the database server. We do the same thing for each pair of adjacent words found in the global query log, as well each set of three contiguous words.

Obviously, a pair of two adjacent words in a query log is not necessarily a meaningful entity. A simple yet effective way to detect meaningful entities from a query log is based on co-occurrence frequencies. If the co-occurrence frequency of two adjacent words is approximately comparable to the frequency of each single word within the pair, it is likely that the two words can be combined together so as to form a more specific entity. As a result, in the third step of pre-computation, for each two-word phrase $\langle w_1, w_2\rangle$ in the list, 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 $f(\langle w_1, w_2\rangle)$ is the frequency of the two-word phrase $\langle w_1, w_2\rangle$ and $f(w_1)$ and $f(w_2)$ are the frequencies of one-word phrase w_1 and w_2, respectively, in the current window W of the global query log. The more frequently two words occur together, the higher the co-occurrence frequency will be. Phrases with a co-occurrence frequency higher than a given threshold will be favored over their individual word components.

The use of a co-occurrence frequency measure may result in some non-traditional entities that would be better termed as phrases or concepts. It should be noted, though, that PLEDS' aim is to detect popular entities, and favors popularity over the traditional definition of "entity". For example, the phrase "good morning" may have a high co-occurrence frequency, and thus be labeled. However, it may not be selected by the user, and in subsequent encounters may not be labeled. In addition, when the co-occurrence frequency is combined with the factors discussed in the following sections, the entity may not be labeled at all. On the other hand, "good morning" may be interesting to a user who is researching greetings. In this case, PLEDS recognizes the user's interest in the topic and promotes entities related to it, so "good morning" may have a higher likelihood of being labeled.

Algorithm 1 The entity extraction algorithm

```
Input: A document $D$, a stop-word list $\mathcal{L}_{stop}$, the frequency list $f$, the co-occurrence list $CoFreq$, a co-occurrence percentage threshold $\delta$;
Output: A candidate entity list $\mathcal{E}(D)$;
1: for each sentence $S \in D$ do
2:   remove punctuation in $S$ and any words in $S$ that are in $\mathcal{L}_{stop}$;
3:   for each word $w_i \in S$ do
4:     if $w_i$ is the first or last word in $S$ then
5:       create a window for $w_i$ containing its one surrounding word;
6:     else
7:       create a window for $w_i$ containing its two surrounding words $W = \{w_{i-1}, w_i, w_{i+1}\}$;
8:     end if
9:     if $CoFreq(\langle w_{i-1}, w_i\rangle) \simeq CoFreq(\langle w_i, w_{i+1}\rangle)$ then
10:      form entity $\langle w_{i-1}, w_i, w_{i+1}\rangle$ and add it to $\mathcal{E}(D)$;
11:    else if entity $\langle w_i - 1, w_i\rangle \notin \mathcal{E}(D)$ then
12:      if $CoFreq(\langle w_i, w_{i+1}\rangle) \geq \frac{\delta}{f(w_i)}$ AND $CoFreq(\langle w_i, w_{i+1}\rangle) \geq CoFreq(\langle w_{i-1}, w_i\rangle)$ then
13:        form entity $\langle w_{i+1}, w_i\rangle$ and add it to $\mathcal{E}(D)$;
14:      else if $CoFreq(\langle w_{i-1}, w_i\rangle) \geq \frac{\delta}{f(w_i)}$ AND $CoFreq(\langle w_{i-1}, w_i\rangle) \geq CoFreq(\langle w_i, w_{i+1}\rangle)$ then
15:        form entity $\langle w_{i-1}, w_i\rangle$ and add it to $\mathcal{E}(D)$;
16:      else
17:        form entity $\langle w_i\rangle$ and add it to $\mathcal{E}(D)$;
18:      end if
19:    end if
20:  end for
21: end for
```

2) Extracting Entities from the Current Web Page: Entity extraction from a web page currently being read by a user relies on mining sentences in the text of the document as well as the frequencies and co-occurrence frequencies of those phrases extracted from the global query logs. To determine whether keyword segment $\langle w_1, w_2\rangle$ in the current page is part of an candidate entity, PLEDS analyzes words found in each sentence of the document D being viewed by a user.

In our current PLEDS implementation, we determine entities of a maximum of three words in length, where the three-word entity has a frequency comparable to that of its individual words or two-word phrases. Basically, this idea can be referred to "concept extension", which has been used in some previous studies [10]. For example, the phrase "Simon Fraser University", has a total frequency similar to those of length-2 subsequence, "Simon Fraser" and "Fraser University". As such, we extend the entity to contain all three words.

Algorithm 1 shows the entity extraction algorithm. We determine whether a single word entity should be extended to a two- or even three-word entity by checking the frequencies as shown in Line 9 to 17. Since longer and more specific entities are preferred, we first determine if the three-word phrase is a meaningful entity. The idea is to check if the co-occurrence frequencies of both two-word phrases in the window around w_i are approximately equal; if it is true, it is very likely that these three words appear together. We combine them together to form a three-word entity. If the three-word phrase is not likely to be an entity, we also need to determine whether the two two-word phrases in the current window W are meaningful entities. Intuitively, we compare the co-occurrence of the two-word phrase and the frequency of one single word. If it is comparable (e.g., θ percent of the frequency of the single word), we assume that the two-word phrase is likely to be an entity. If the co-occurrence is too small, we just treat the single word as a candidate entity.

Since the likelihood of adding a fourth word on either side of a three word phrase entity is quite low (that is, the likelihood of a four-word entity having a comparable high frequency to the three-word entity is low), we stop at adding the third word. This increases the efficiency of our entity extraction algorithm.

Once the initial candidate entities have been found, PLEDS fetches the pre-computed frequency $f(e)$ of each entity e in the current window W in the global query log. The global frequencies of entities reflect how popular and interesting the entities are for the general population within this current window. If an entity was popular over a long time ago but not recently, we can thus capture this with a lower global frequency for that entity.

Several previous studies [10], [4] concluded that entity detection based on word co-occurrence may not be very accurate. However, our method combines the word co-occurrence in the document with the word co-occurrence in the query log to identify meaningful entities. Only the terms (a set of continuous words) in the document that frequently appear in the query log are considered to be candidate entities.

B. Mining Users' Background Knowledge from Local Logs

If a user has already read something about an entity, or clicked the entity using PLEDS before, then it is less likely that the user will click the entity again. In other words, a user's background knowledge is important in determining her/his interest in entities.

1) Mining Local Web Log Data: By mining the local web log data, PLEDS can identify whether an entity or some highly related entities were queried recently. This information can be used in the following two ways.

First, if an entity was queried recently by a user, then the user may not be interested in the entity in the near future. We capture this by finding the *query freshness* of entities. 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, if an entity e has a high freshness score and many entities in the same category of e were queried before, the user may have a special interest in the category of e, and thus e may have a good chance of being clicked by that user. To model the entity ontology, we use the concept of *sense*. The term *sense* arises from WordNet [3], and refers to the meaning of the word it belongs to. Each word may have several senses. For example, the word "merit" has two senses: the first being "any admirable quality or attribute", as in the example "work of great merit"; and the second being "the quality of being deserving (e.g., deserving assistance)", as in the example "there were many children whose merit he recognized and rewarded". Each sense belongs to a different *synset*, which in turn is a group of synonyms. The senses in WordNet have a taxonomy structure. For simplicity, we only consider those most specific senses.

Algorithm 2 Calculating a user's interest vector

```
Input: User's local web log $\mathcal{L}$, a stop word list $\mathcal{L}_{stop}$, a user parameter $k$;
Output: User's interest vector $\mathcal{V}$;
 1: for each query $q \in \mathcal{L}$ do
 2:     remove punctuation in $q$ and any words in $q$ that are in $\mathcal{L}_{stop}$;
 3:     disambiguate all words $w_i \in q$ //get $w_i$'s part of speech and most likely sense $sen(w_i)$;
 4:     for each word $w_i \in q$ do
 5:         if $sen(w_i) \in \mathcal{V}$ then
 6:             increment the frequency of $sen(w_i)$ in $\mathcal{V}$ by 1;
 7:         else
 8:             insert a new tuple $(sen(w_i), 1)$ to $\mathcal{V}$;
 9:         end if
10:     end for
11: end for
12: find the top $k$ senses in $\mathcal{V}$ and remove all others from $\mathcal{V}$;
13: normalize the sense frequency by dividing the total number of senses in $\mathcal{V}$;
```

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. For example, a possible interest vector for a user u_i may look like $V(u_i) = \langle (sen_1, freq_1), \ldots, (sen_j, freq_j), \ldots \rangle$, where sen_j represents a sense and $freq_j$ represent the frequency of sense sen_j. We can then compare the most likely senses for a given entity to those of the user's interest vector, and use the overlap to calculate an *interest score* for each entity.

Algorithm 2 calculates a user's interest vector. The disambiguation in Line 3 of Algorithm 2 refers to that provided by the Adapted Lesk Algorithm [1], which compares words surrounding our target word w_i using a measure of semantic similarity, thus finding the most appropriate sense and part of speech for w_i. We use the WordNet [3] semantic lexicon for the English language and its .NET library, WordNet.NET[10], as well as some useful code developed in The Code Project[11] for the implementation.

The user's interest models the likelihood of the user being interested in a specific topic (sense in our model). The user's interest vector can be used to measure the likelihood that an entity will be interesting to a user. The measurement is based on an *interest score* for each entity.

Given a user u_i and the corresponding interest vector $V(u_i)$. For entity e_j, suppose its total number of senses is n and senses $sen_{i_1}, \ldots, sen_{i_t}$ are appeared in $V(u_i)$. The interest score of e_j for user u_i can be calculated as

$$IS_{u_i}(e_j) = \sum_{k=1}^{t} \frac{1}{n \times freq_{u_i}(sen(i_k))}.$$

If an entity e_j has n different senses, we can assume that

[10]`http://opensource.ebswift.com/`

[11]`http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx`

each sense has the probability $\frac{1}{n}$. As a result, by multiplying the frequency for each common sense between e_j and the user's interest vector, we can obtain the interest score to estimate the likelihood the user will be interested in the entity.

2) Mining PLEDS Clickthrough History: Search engine clickthrough data has been widely accepted as a useful source of implicit user feedback [6], [7]. In PLEDS, entity clickthrough data also can be used as an implicit user feedback. If a user has clicked an entity highlighted by PLEDS before, then she/he is unlikely to click the entity again in the near future. Moreover, if PLEDS presents an entity to a user a few times but the user never clicks it, then the chance of the user clicking this entity in the near future is also slim.

Carrying this idea forward, PLEDS mines historical clickthrough data. We keep a record of each time a user clicks on an entity, 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. We assume that 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.

C. Mining the Current Web Page

On the web page currently being read by a user, if an entity is well explained, then it is likely that the user will not click on that entity. Therefore, it is necessary to mine the current web page to understand whether an entity is well explained or not.

We propose an *explanative score* to address this issue. 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. 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 that used in [12]. The larger the *explanative score*, the better explained the entity.

The semantic similarity calculation is provided by [12], although other methods could be substituted. To use this measurement, we treat the WordNet taxonomy as an undirected graph, using the distance between two nodes as a measure of their semantic relatedness. A larger distance results in a score closer to 0, meaning the words are not highly semantically related. A smaller distance results in a score closer to 1, meaning the words are highly semantically related. Identical words (taking into consideration the part of speech; this must also be identical) will have a distance of 0, resulting in a score of 1. We also consider the depth of each node's least common ancestor.

Algorithm 3 Explanative score calculation.

```
Input: A document $D$, a stop word list $\mathcal{L}_{stop}$;
Output: The explanative score for each entity;
 1: for each sentence $S \in D$ do
 2:     extract the initial entities $e_i$ from $S$ using the method from
        Section III-A.2;
 3:     remove punctuation in $S$ and any words in $S$ that are in $\mathcal{L}_{stop}$;
 4:     for each word $w_i \in S$ do
 5:         disambiguate $w_i$ //this gets the part of speech of the word;
 6:     end for
 7: end for
 8: for each entity $e_i \in D$ do
 9:     Create a window which holds a maximum of 5 entities:
        $e_{i-2}, e_{i-1}, e_i, e_{i+1}, e_{i+2}$;
10:     for each pair of entities $(e_j, e_i)$ do
11:         if $(e_j, e_i)$ is in the word-pairs list and its corresponding
            similarity score is not 0 then
12:             use that score as the similarity score;
13:         else if $(e_j, e_i)$ is in the word-pairs list and its corresponding
            similarity score is 0 then
14:             calculate the new score, store it in the table, and use it
                as the similarity score;
15:         else
16:             assign a score of 0 and store in the word-pairs list;
17:         end if
18:     end for
19:     calculate the average of similarity scores for all pairs, and let
        it be the explanative score for$e_i$;
20: end for
```

Calculating *explanative score* presents an efficiency challenge, as each entity determined via the co-occurrence method must be compared to each of its surrounding entities. Limiting the window size helps reduce computational time, but also reduces the accuracy of the part-of-speech tagging. As such, some pre-computation techniques have also been introduced to mitigate this problem. A list of pairs of entities that have been compared previously is maintained, along with their corresponding similarity scores. Then, for each pair of entities encountered in the text, if the pair is in the list, we use its pre-computed score. If not, we assign the pair a score of 0. Here we are assuming that if an entity pair has never before been encountered, it is not very common and thus the entities it consists of are not highly related to each other. If the entity pair is encountered a second time, the semantic similarity is calculated and the previous score of 0 is replaced with this new score. Here we assume that since the pair has been encountered previously, it is now common enough to warrant performing the calculation. In subsequent encounters this pre-computed score is used, thus saving the cost of computing the semantic similarity every time. This is illustrated in Algorithm 3 to calculate explanative score $ES(e_i)$ for an entity e_i.

It is worth noting that an entity may appear more than once in a web page. For such an entity, we use the largest explanative score among the multiple occurrences. Here we assume that if an entity is well-explained at least once on the page, there is a decreased need of tagging and explaining this entity elsewhere on the same page. This is also illustrated in the last step of the algorithm above.

An interesting issue here is that the user may not consider a given web page as trustworthy. For example, if the current document is not well-written, even though an entity is well explained in the document, the user may still try to search the web for an explanation from a reputable source. A possible solution is to combine the trust score of each web page, such as HITS and PageRank, into our explanative score calculation. Another idea is to trace the user's browsing history. If a user immediately queries a currently well-explained entity, we may assume that the user does not trust the document's authority. Later on, its authority needs to be penalized. We leave this as a future improvement of PLEDS.

D. *Fusing the Mining Results*

By mining the global web query log, the local web log and PLEDS clickthrough data, as well as the current web page, we obtain information about currently trendy keywords, the user's background knowledge about the entities on the current web page, as well as how well-explained these entities are in the page. Based on these factors, PLEDS uses logistic regression to recommend a list of entities to be tagged for the user.

Technically, PLEDS estimates the probability p that an entity e will be clicked on by the user given five factors x_1, x_2, x_3, x_4 and x_5, where x_1 is the explanative score (Section III-C), x_2 is the global frequency (Section III-A.2), x_3 is the query freshness (Section III-B.1), x_4 is the click freshness (Section III-B.2), and x_5 is an interestingness score (Section III-B.1), which is computed using the interest vector.

PLEDS takes a training data set to learn the logistic regression model. The formula of the logistic regression is $ln\frac{p}{1-p} = \beta_0 + \sum_{i=1}^{5} \beta_i \cdot x_i$, where $\beta_0, \ldots, \beta_5$ are the coefficients. In other words, the probability p that an entity will be clicked on can be measured as

$$p = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{5} \beta_i \cdot x_i)}}.$$

We adopted the "Newton-Raphson" method [5] to learn the regression coefficients. The Newton-Raphson method is a common iterative approach to estimating a logistic-regression model. In the training data set, if a labeled entity is clicked, a training example is obtained with the response value y set to 1. If an entity is labeled by PLEDS but is not clicked by the user, a training example is also obtained with response value y set to 0.

We use $\mathbf{B} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5]^T$ to represent the coefficient vector. $\mathbf{B_t}$ represents the coefficient vector in t-th step of iteration. The Newton-Raphson method accepts the response vector and iteratively updates the coefficient vector until it converges. The algorithm is shown in Algorithm 4.

Once the model is trained, for each new web page PLEDS will use the model to retrieve entities which have probabilities above a certain threshold value. This threshold value can be tuned by the user to adjust how aggressive PLEDS should be in detecting and displaying entities. Those entities above the threshold will be labeled by PLEDS. Depending on PLEDS' recommendation confidence (the score calculated using regression), the entities are labeled using different colors, which display the confidence of the labeling results.

Once the entities within a web page have been identified, it is necessary to provide the appropriate information regarding those entities, depending on the user's interest. PLEDS submits a web search for the entity in association with the user's interest and extracts a summary of those top ranked search results. We also provide a short definition of the entity as found in WordNet, if that definition exists, displaying the top three ranked search results below it. To display this information, the user clicks on the entity they are interested in, and the information is displayed as a pop-up. Because the user is required to click on an entity rather than simply hover with the mouse, we avoid the problems of distraction exhibited by other systems, and ensure that the user is actually motivated to see this information.

Algorithm 4 The Newton-Raphson method to learn the regression coefficients.

```
Input: the model matrix $\mathbf{X}$ in which each row, denoted as $\mathbf{x_i}$, is a
    vector containing the values for $i$-th training data, the response
    vector $\mathbf{y}$ (containing 0's and 1's);
Output: The coefficient vector $\mathbf{B}$;
1: choose initial estimates of the regression coefficients, such as
   $\mathbf{B_0} = 0$;
2: let $t = 0$;
3: repeat
4:     let $t = t + 1$;
5:     let $\mathbf{p_{t-1}}$ be the probability vector of fitted probabilities from
       the previous iteration which can be calculated using $p_{i,t-1} = \frac{1}{1+e^{-\mathbf{x_i^T B_{t-1}}}}$;
6:     let $\mathbf{V_{t-1}}$ be the diagonal matrix with diagonal entries
       $p_{i,t-1}(1 - p_{i,t-1})$;
7:     let $\mathbf{B_t} = \mathbf{B_{t-1}} + (\mathbf{X^T V_{t-1} X})^{-1} \mathbf{X^T} (\mathbf{y} - \mathbf{p_{t-1}})$;
8: until ($\mathbf{B_t}$ is close enough to $\mathbf{B_{t-1}}$)
9: return $\mathbf{B_t}$;
```

IV. Experimental Results

In the following section, we first describe the methodology used to evaluate the utility of our PLEDS platform, and then present the results of a systematic user study.

The PLEDS prototype system was implemented in Microsoft .NET using C♯. Microsoft SQL Server 2000 was used as the background database management system. All the experiments were conducted on a PC computer running the Microsoft Windows XP SP2 Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk.

In our user studies, a large, real web search query log from AOL (`http://www.aol.com/`) was used, although reduced in size through data cleaning and to increase performance. Data cleaning consisted of removing tuples that consisted solely of punctuation symbols or single letters. At the start of user testing, the size of the global query log used by PLEDS contained $97,471$ tuples (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 passed through a threshold filter as described above.

A. *Evaluation Methodology*

Our evaluation methodology consisted of usability testing, which was conducted once optimized settings for certain parameters in PLEDS had been set. The goal of the evaluation was to measure the quality of entity recommendations provided by the system for a specific navigational session. We present the results of our study in the following section.

Usability testing was conducted on PLEDS using a set of volunteers with varying backgrounds, from non-technical users to those who are highly skilled in browsing and navigating the internet. In total we have 6 participants. Participants had a range of educational backgrounds, with 16.7% participants with a highschool diploma, 33.3% participants with a Bachelor's degree, and 50% participants with a Master's degree. Participants' use of computers also ranged from $6 - 10$ hours per week to 50+ hours per week, indicating that some have more opportunities to become familiar with the internet and other technical computer skills than others.

Testing was conducted to determine the ability of PLEDS to adapt to a user's interest and its performance in comparison with Wikipedia.

Each user session consisted of two stages. In the first, participants used PLEDS to browse various Wikipedia pages for 15 minutes. During this time they were permitted to click on entities they were interested in, thus training the system. At the end of this period, users were asked to look at one article of interest to them. The tagging of entities was disabled, so users only saw plain text. They were then asked to identify entities they were interested in. When the session was over, these entities were compared with those identified by the system in its initial and final states. They were also compared with entities as tagged by Wikipedia, and the precision and recall of all three stages were measured.

B. User Satisfaction

Users involved in our user study were asked to complete a questionnaire following the completion of the session. The questionnaire covered a range of topics from general user background, to their experience with PLEDS in comparison to Wikipedia, and finally users were asked to rate their level of satisfaction with PLEDS alone. Table I shows the results.

With regard to satisfaction with PLEDS, we asked participants to rate their experience with PLEDS according to several factors, including if the entities were recently clicked or queried by the user and whether the entities were well-explained in the document. On average, participants reported that the entities that PLEDS recommended to them were neither recently clicked nor queried by them, which is highly desirable. However, it should be noted that although the entities were still recommended, it may be that their recommendation level had changed; for example, entities, once clicked on, are sometimes downgraded from strongly recommended to weakly recommended as their scores change and are updated. The color of tag used for these entities does change, but the entity may still be recommended due to other factors.

Participants also reported that the entities recommended to them by PLEDS were somewhat explained in the document. This may have been affected by the short length of text.

Finally, participants reported that they were likely to use PLEDS frequently for surfing the internet, with one participant reporting that "to use this method would result in getting specific information more quickly than using broader search methods. This is a good method for scanning rather than having to read everything."

System	Precision	Recall
Wikipedia	0.16337014	0.33
PLEDS - Initial	0.072222222	0.194444444
PLEDS - After Adaptation	0.34023569	0.841666667

TABLE III
THE PRECISION AND THE RECALL FOR THE SYSTEM COMPARISON.

C. System Comparison

As mentioned in Section I and Section II, our PLEDS system is highly related to several existing systems proposed in the literature such as *IntelliTXT*, *Kontera* and *Contextual Shortcuts*. With no access to those systems, alternatively, we conducted a user study to compare the results for PLEDS and Wikipedia. The source for the text used in these studies originated in Wikipedia (English version), but for the PLEDS trials the text was extracted from Wikipedia articles, and all formatting, links, and tags are removed. The entities were then "labeled", and these labels are compared with the link entities in the original Wikipedia articles.

We examined the entities labeled by PLEDS and Wikipedia, as well as the entities participants identified as interesting. The results of the comparison between PLEDS in its initial state, PLEDS in its final state, and Wikipedia test are shown in Table III.

1) Comparison: PLEDS adapted vrs. PLEDS initial: The entity recommendation performed by PLEDS after the period of adaptation results in a low precision score (0.34), although the recall score (0.84) is high. However, these results show a marked improvement over PLEDS' in its intial state, where both precision and recall scores are lower by a factor of four (approximately). Recall that during usability testing, PLEDS only had 15 minutes to adapt to a participant's preference. With longer length of use, the results should improve further. In addition, this short length of time may also be to blame for low precision scores; users' local query logs were quite small because of this, having on average only 10 tuples. Historically, however, the global logs from the dataset show previous users with between 30 and 120 tuples each. As the local query log becomes larger, PLEDS becomes more accurate. Thus, giving users more time with the system may lead to better results.

2) Comparison: PLEDS vrs. Wikipedia: In its initial state, PLEDS does not perform as well as Wikipedia with respect to precision and recall. However, as PLEDS adapts to the user's preference, the results show that it outperforms Wikipedia. One reason for the disparity is that PLEDS results could be adjusted to only show the top k results, which is what we used to score the system. On the other hand, Wikipedia provides a set number of results, which means that for some pages, 20% of entities may be returned, while for others the number may be as high as 40%. During the study, participants often switched from higher entity percentage settings to lower ones, indicating that too many entities is not desirable on a page.

In comparing PLEDS and Wikipedia via the results of the questionnaire users filled out after their usability sessions.

ID	Question Description	SD	D	NO	A	SA
Q1	The entities recommended by PLEDS were NOT recently clicked by me.	0	16.7%	0	50%	33.3%
Q2	The entities recommended by PLEDS were NOT recently queried by me.	0	16.7%	0	83.3%	0
Q3	The entities recommended by PLEDS were NOT well-explained in the document.	0	16.7%	33.3%	33.3%	16.7%
Q4	I think that I would like to use PLEDs frequently for surfing the internet.	0	0	33.3%	50%	16.7%

TABLE I

THE SURVEY RESULTS (SD=STRONG DISAGREE, D=DISAGREE, NO=NO OPINION, A=AGREE, SA=STRONG AGREE).

ID	Question Description	SD	D	NO	A	SA
Q5	The words/phrases recommended by PLEDS matched my interests well.	0	0	0	83.3%	16.7%
Q6	The words/phrases recommended by Wikipedia matched my interests well.	16.7%	16.7%	16.7%	33.3%	16.7%
Q7	The words/phrases recommended by PLEDS were meaningful in the context.	0	0	16.7%	83.3%	0
Q8	The words/phrases recommended by Wikipedia were meaningful in the context.	0	16.7%	0	50%	33.3%

TABLE II

THE SYSTEM COMPARISON RESULTS (SD=STRONG DISAGREE, D=DISAGREE, NO=NO OPINION, A=AGREE, SA=STRONG AGREE).

The results are shown in Table II. 100% users reported that the entities recommended by PLEDS matched their interests well, while users were more mixed in their reaction to those entities recommended by Wikipedia; 33.3% did not feel the entities matched their interests well, 50% felt the entities did match their interests well, while 16.7% had no opinion either way. On the other hand, in terms of meaningfulness, 83.3% users felt that Wikipedia's recommendations were meaningful in the context of the text, while 83.3% users felt that the recommendations made by PLEDS were meaningful. 16.7% users had no opinion on the meaningfulness of the entities recommended by PLEDS.

D. Discussion

Our experimental results were affected by the length of time of each user session and the limitation of only having one session per participant. As discussed previously, allowing users to have longer and more varied access to PLEDS may provide more accurate precision and recall scores.

Our results also showed that Wikipedia's fixed number of tagged entities is a disadvantage in terms of precision and recall; with its precision being roughly half of PLEDS' and its recall only roughly 40% that of PLEDS'. The problem is that Wikipedia often shows too many results, may show results that are already well-explained in the text, and may show results that have recently been clicked or queried by the users. This resulted in general in less overlap between the entities the users desired to click on and the entities labeled in the text. On the other hand, Wikipedia tends to do a better job of displaying multi-word entities and phrase entities than PLEDS; one contributor to this may be the reduced size of the PLEDS global query log used in this test for performance reasons. If the global query log were to be expanded, it is expected that more multi-word and phrase entities may be discovered in the text. Wikipedia also outperforms PLEDS if PLEDS has not had any opportunity to learn a user's interest.

V. CONCLUSIONS

PLEDS builds on previous systems, such as *IntelliTXT*, *Kontera* and *Contextual Shortcuts*, to provide personalized, meaningful entity recommendations in text. We have shown how we can improve on these systems by introducing the new measures of *Interest Score*, *Explanative Score*, *Query Freshness*, and *Click Freshness*, as well as more traditional *Frequency* measures. Our results show that as it learns a user's interest, PLEDS recommends and retrieves more relevant entities for specific users than static systems such as Wikipedia.

There are several areas to explore with regard to the improvement of PLEDS. We would like to expand our initial entity detection particularly with respect to concept extension.

REFERENCES

[1] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. 2002.

[2] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 805–810, Acapulco, Mexico, 2003. Morgan Kaufmann.

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

[4] R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. A statistical model for multilingual entity detection and tracking. *NAACL/HLT*, 2004.

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[6] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 133–142, New York, NY, USA, 2002. ACM.

[7] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'05)*, pages 154–161, New York, NY, USA, 2005. ACM.

[8] J. Morkes and J. Nielsen. Concise, scannable, and objective: How to write for the web, 1997.

[9] Hartmut Obendorf and Harald Weinreich. Comparing link marker visualization techniques: changes in reading behavior. In *Proceedings of the 12th international conference on World Wide Web (WWW'03)*, pages 736–745, Budapest, Hungary, 2003. ACM.

[10] Vadim von Brzeski, Utku Irmak, and Reiner Kraft. Leveraging context in user-centric entity detection systems. In *Proceedings of the 2007 ACM International Conference on Information and Knowledge Management (CIKM'07)*, Lisbon, Portugal, 2007. ACM.

[11] Harald Weinreich, Hartmut Obendorf, and Eelco Herder. Data cleaning methods for client and proxy logs. In *Proceedings of WWW'06 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, 2006.

[12] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pages 133–138, Las Cruces, NM, 1994. ACL.