Dynamic Web Log Session Identification With Statistical Language Models

Xiangji Huang
School of Analytical Studies and Information Technology, York University, Toronto, Ontario, Canada M3J 1P3.
E-mail: jhuang@yorku.ca

Fuchun Peng
Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts,
Amherst, MA 01003. E-mail: fuchun@cs.umass.edu

Aijun An
Department of Computer Science,York University, Toronto, Ontario, Canada M3J 1P3. E-mail: aan@cs.yorku.ca

Dale Schuurmans
Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E8.
E-mail: dale@cs.ualberta.ca

We present a novel session identification method based on statistical language modeling. Unlike standard timeout methods, which use fixed time thresholds for session identification, we use an information theoretic approach that yields more robust results for identifying session boundaries. We evaluate our new approach by learning interesting association rules from the segmented session files. We then compare the performance of our approach to three standard session identification methods—the standard timeout method, the reference length method, and the maximal forward reference method—and find that our statistical language modeling approach generally yields superior results. However, as with every method, the performance of our technique varies with changing parameter settings. Therefore, we also analyze the influence of the two key factors in our language-modeling–based approach: the choice of smoothing technique and the language model order. We find that all standard smoothing techniques, save one, perform well, and that performance is robust to language model order.

Introduction

Because of the rapid expansion of the World Wide Web, the use of automated Web-mining techniques to discover useful, relevant information has become an increasingly important research area. One important subarea is Web usage mining, wherein one attempts to discover patterns of Web usage from Web log data. Although Web log data are usually noisy and extremely ambiguous, there remains a potential for discovering useful structure in the interactions between a Web site and its users. Such data can be studied to generate inferences about Web site design, test prototypes of Web sites or their modifications, and test hypotheses about the effects of different design variables on Web-user behavior (Burton & Walther, 2001).

Generally speaking, Web logs record users' requests to a Web server. A request is recorded in a log file entry, which contains different types of information, including the IP address of the computer making the request, the user access date and time, the document or image requested, and so on. Depending on the popularity of the Web site, a Web log can record thousands or tens of thousands of requests every day. To find useful patterns (such as association rules or sequential patterns) from this vast amount of information, requests (or log entries) need to be grouped into usage sessions. A session is defined as a group of requests made by a single user for a single navigation purpose. A user may have a single session or multiple sessions during a period of time. Only once these atomic sessions have been identified can common usage patterns among sessions be discovered by Web usage mining algorithms.

The most commonly used session identification method is called *timeout,* in which a user session is usually defined as a sequence of requests from the same user such that no two consecutive requests are separated by an interval more

Accepted January 23, 2004

© 2004 Wiley Periodicals, Inc. ● Published online 13 August 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.20084

than a predefined threshold. This session identification method suffers from the problem that it is difficult to set the time threshold. Different users may have different navigation behaviors, and their time intervals between sessions may be significantly different. Even for the same user, intervals between sessions may vary. A dynamic session identification method that is based on the context of requests would seem to be much more appropriate.

The problem of determining session boundaries is not only important for analyzing Web sever logs but also for compiling database usage statistics. Currently, all database vendors use the timeout method to collect data on the use of their databases. These data are then used to compile statistics for various database library routines (such as query optimization). Library effectiveness depends critically on the accuracy of the supplied data. A better session detection method can improve the quality of the data collected for these purposes (Duy & Vaughan, 2003).

The goal of our study is to overcome the problems of existing session identification methods by proposing a novel, more accurate session detection method that can supply better data for Web mining and database usage analysis. Our proposed method is based on statistical language models. It does not rely on any time intervals when identifying session boundaries. Instead, it uses an information-theoretic approach to identifying session boundaries dynamically by measuring the change of information in the sequence of requests. To determine whether the proposed method works well in practice, we conducted a series of experiments that compare the language-modeling-based method with the timeout method, as well as two other methods. We also investigate the effects of different parameter choices for our statistical language models.

The remainder of this article is organized as follows. First, in the next section, Related Work, we describe current work on session identification in Web log data. Then, in Statistical Language Modeling, we introduce the basic elements of statistical language modeling. We then describe how statistical language models can provide a natural method for identifying session boundaries in the section on Session Detection Using Language Models. In Empirical Evaluation, we describe our method for evaluating session identification methods, describe the data set we use in our evaluation, and compare the performance of our proposed language-modeling-based approach against three standard session identification methods. In Analysis and Discussion, we go over the experimental results. Implications of our study for Web design are discussed in Implications of Findings for Web Design. Finally, we conclude the article in the last section, Conclusions and Future Work.

Related Work

There are several session identification methods reported in the literature. The most common and simplest method is *timeout*. In the timeout method, a session shift is identified between two requests if the time interval between the two requests is more than a predefined threshold. He and Goker (2000) reported the results of experiments that used the timeout method on two sets of Web logs. In their experiments, the threshold was set large initially and then gradually decreased. The authors concluded that a time range of 10 to 15 minutes was an optimal session interval threshold. Catledge and Pitkow (1995) also reported the results of an experiment where a Web browser was modified to record the time interval between user actions on the browser's interface. One result was that the average time interval between two consecutive events by a user was 9.3 minutes. Assuming that the most statistically significant events occur within 1.5 standard deviations from the mean, 25.5 minutes was subsequently recommended as the threshold for session identification. However, the optimal timeout threshold clearly depends on the specific problem. Once a site log has been analyzed and its usage statistics obtained, a timeout threshold that is appropriate for the specific Web site can be fed back into the session identification algorithm. Despite the application dependence of the optimal interval length, most commercial products use 30 minutes as a default timeout.

Cooley, Mobasher, and Srivastava (1999) proposed a transaction identification method, called *reference length*. This method assumes that the amount of time a user spends on a page is correlated with whether the page is an "auxiliary" or "content" page for that user. By analyzing the histogram of page reference lengths, the authors found that the time spent on auxiliary pages is usually shorter than that spent on a content page, and also that the variance of the times spent on auxiliary pages is smaller than content pages. If an assumption is made about the percentage of auxiliary references in a log, then a reference length can be calculated that estimates the optimal cutoff between auxiliary and content references based on the histogram. Once pages are classified as either auxiliary or content pages, a session boundary will be detected whenever a content page is met. The problem with this method is that only one content page is included in each session. This may not be a good model for real sessions because users may obviously look at more than one content page for a single retrieval purpose.

A final session identification method, referred to as *maximal forward reference,* is from Chen, Park, and Yu (1998). In this approach, each session is defined as the set of pages from the first page in a request sequence to the final page before a backward reference is made. Here, a backward reference is naturally defined to be a page that has already occurred in the current session. One advantage of the maximal forward reference method is that it does not have any parameters that make assumptions about the characteristics of a particular data set. However, it has the significant drawback that backward references may not be recorded by the server if caching is enabled at the client site.

Statistical Language Modeling

The original motivation for statistical language modeling comes from speech recognition, where the goal is to predict the probability of natural word sequences. Given a word sequence, $s = w_1 w_2 \ldots w_N$, its probability can always be written using the probability chain rule as:

$$P(s) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_N|w_1\ldots w_{N-1})$$

$$= \prod_{i=1}^{N} P(w_i|(w_1\ldots w_{i-1})$$

The simplest and most successful statistical language models have been n-gram language models. In n-gram language modeling, it is assumed that the probability of a word only depends on its at most $n - 1$ preceding words. Thus, the probability of a word sequence s becomes

$$P(s) = \prod_{i=1}^{N} P(w_i|(w_{i-n+1}\ldots w_{i-1})$$

A statistical language model, then, is a specific choice of conditional probabilities for all possible n-grams: $P(w_i|w_{i-n+1}\ldots w_{i-1})$.

The quality of a given statistical language model can be measured by its empirical perplexity and entropy on a given corpus of text s (Bahl, Jelinek, & Mercer, 1983), where the empirical perplexity of the model on s is defined as

$$Perplexity(s) = P(s)^{-\frac{1}{N}}$$

and the empirical entropy of the model on s is

$$Entropy(s) = \log_2 Perplexity(s)$$

$$= -\frac{1}{N}\log_2 P(s)$$

That is, we would like the language model to place high probability on natural test sequences s, and hence obtain a small value of empirical perplexity or entropy.

The key issue in statistical language modeling is how to estimate the n-gram probabilities from a given corpus of training data. A straightforward method for estimating n-gram probabilities uses the observed frequencies of word sequences in the training corpus as follows:

$$P(w_i|w_{i-n+1}\ldots w_{i-1}) = \frac{\#(w_{i-n+1}\ldots w_i)}{\#(w_{i-n+1}\ldots w_{i-1})} \qquad (1)$$

where #(.) is the number of occurrences of a specified gram in the training corpus. Although one could attempt to use this simple n-gram model to capture long range dependencies in language, such a simple approach to estimation suffers from the sparse data problem. For example, to train a trigram model with a vocabulary size of 20,000, there are 8 trillion free parameters to be estimated. However, any reasonable training set may only contain a sequence of a few million words. In general, using grams of length up to n entails estimating the probability of W^n events, where W is the size of the word vocabulary. Because of the heavy tailed nature of language (i.e., Zipf's law), one is likely to encounter novel n-grams that were never witnessed during training in a test corpus, and the probability for these unseen n-grams should clearly not be zero. Therefore, a mechanism for assigning nonzero probability to novel n-grams is a central and unavoidable issue in statistical language modeling. One standard approach to smoothing probability estimates to cope with the sparse data problem (and to cope with potentially missing n-grams) is to use some sort of back-off estimator as follows (Katz, 1987).

$$P(w_i|w_{i-n+1}\ldots w_{i-1}) = \begin{cases}\hat{P}(w_i|w_{i-n+1}\ldots w_{i-1}), & \text{if}\#(w_{i-n+1}\ldots w_i) > 0\\ \beta(w_{i-n+1}\ldots w_{i-1}) \times P(w_i|w_{i-n+2}\ldots w_{i-1}), & \text{otherwise}\end{cases} \qquad (2)$$

where

$$\hat{P}(w_i|w_{i-n+1}\ldots w_{i-1}) = \frac{discount\#(w_{i-n+1}\ldots w_i)}{\#(w_{i-n+1}\ldots w_{i-1})} \qquad (3)$$

is called *discounted probability* and $\beta(w_{i-n+1}\ldots w_{i-1})$ is a normalization constant calculated to be

$$\beta(w_{i-n+1}\ldots w_{i-1}) = \frac{1 - \sum_{x:\#(w_{i-n+1}\ldots w_{i-1}x)>0} \hat{P}(x|w_{i-n+1}\ldots w_{i-1})}{1 - \sum_{x:\#(w_{i-n+1}\ldots w_{i-1}x)>0} \hat{P}(x|w_{i-n+2}\ldots w_{i-1})} \qquad (4)$$

Different methods can be used for computing the discounted probability in Equation (3). Typical discounting techniques include absolute smoothing (ABS), Good-Turing smoothing (GT), linear smoothing (LIN), and Witten-Bell smoothing (WB) (Chen & Goodman, 1998). The objective of smoothing is to reserve a small amount of probability mass for unobserved events. Different discounting techniques have different assumption on how to reserve this probability mass. In the next paragraphs, we briefly introduce the four discounting methods we considered.

Absolute discounting. In absolute discounting, the frequency of a word is subtracted by a constant c. The probability of w_i given $w_{i-n+1}\ldots w_{i-1}$ is then calculated as:

$$\hat{P}(w_i|w_{i-n+1}\ldots w_{i-1}) = \frac{\#(w_{i-n+1}\ldots w_i) - c}{\#(w_{i-n+1}\ldots w_{i-1})}$$

where c is often defined as

$$c = \frac{n_1}{n_1 + 2n_2}$$

where n_r denotes the number of n-grams that occur r times. The definition of n_r also applies to the other smoothing techniques, described below.

Good-Turing discounting. In standard Good-Turing discounting, the frequency r is discounted as

$$GT_r = (r + 1)\frac{n_r + 1}{n_r}$$

where the probability of w_i given $w_{i-n+1} \dots w_{i-1}$ is calculated as:

$$\hat{P}(w_i|w_{i-n+1} \dots w_{i-1}) = \frac{GT_{\#(w_{i-n+1} \dots w_i)}}{\#(w_{i-n+1} \dots w_{i-1})}$$

Linear discounting. In linear discounting, the probability of a word w_i given $w_{i-n+1} \dots w_{i-1}$ is calculated as:

$$\hat{P}(w_i|w_{i-n+1} \dots w_{i-1}) = \alpha \frac{\#(w_{i-n+1} \dots w_i)}{\#(w_{i-n+1} \dots w_{i-1})}$$

where α is defined as:

$$\alpha = 1 - \frac{n_1}{N}$$

where N denotes the number of events (uni-grams).

Witten-Bell discounting. Witten-Bell discounting is similar to the linear discounting. The probability of a word w_i given $w_{i-n+1} \dots w_{i-1}$ is calculated as:

$$\hat{P}(w_i|w_{i-n+1} \dots w_{i-1}) = \alpha \frac{\#(w_{i-n+1} \dots w_i)}{\#(w_{i-n+1} \dots w_{i-1})}$$

where α is defined differently as:

$$\alpha = 1 - \frac{C}{\#(w_{i-n+1} \dots w_{i-1}) + C}$$

where C denotes the number of distinct words that can follow $w_{i-n+1} \dots w_{i-1}$ in the training data.

Session Detection Using Language Models

Statistical language modeling has traditionally been used in speech recognition. However, it has recently become more widely used in many other application areas. Although the original motivation of language modeling is to estimate the probability of naturally occurring word sequences, language modeling actually provides a general strategy for estimating the probability of *any* sequence—regardless of whether the basic units consist of words, characters, or any other arbitrary alphabet. In this sense, many problems can be formulated as a language-modeling problem. In Web usage mining, Web pages (or objects) are visited sequentially in a particular order, similar to the word sequences that occur in a natural language. If we consider each visited object as a basic unit, like a word or character in natural language, we can then attempt to estimate the probability of object sequences using the same language modeling tools described earlier.

The basic goal of session identification is to group sequential log entries that are related to a common topic and segment log entries that are unrelated. Language modeling provides a simple, natural approach to segmenting these log sequences. Imagine a set of objects on a common topic that are frequently visited one after another. In this case, the entropy (or perplexity) of the sequence is low. However, when a new object is observed in the sequence that is not relevant to the original topic (but in fact indicates a shift to a new topic), the introduction of this new object causes an increase in the entropy of the sequence because it is rarely visited after the preceding objects. Such an entropy increase serves as a natural signal for session boundary detection. If the change in entropy passes a threshold, a session boundary could be placed before the new object. In other words, the uncertainty (which is measured by entropy) within a session should be roughly constant, allowing for a fixed level of variability within a topic. However, whenever the entropy increases beyond a threshold, this presents a clear signal that the user's activity has changed to another topic. Thus, we should set a session boundary at the place where the entropy changes. The threshold on the entropy change can be tuned to adjust the number of sessions generated.[1]

Figure 1 shows the entropy sequence we obtained in our Web log data set, where the *X*-axis is the position of objects and *Y*-axis is the entropy of the sequence from the first object to the current object. As one can see, the entropy changes radically at some points, although it remains stable in other places. This figure gives an intuition how entropy could be used for session boundary detection.

Empirical Evaluation

In this section, we present an empirical evaluation of the effectiveness of our language-modeling-based session-detection method on a real application data set. Here, we first describe the data set and data-preprocessing methods we used. We then discuss our method for comparing different session identification methods and finally present the evaluation results we obtained. The section, Analysis and Discussion, then follows with an analysis of the results.

The Data Sets

The log files used in our experiments were extracted from Livelink access data over a period of two months. Livelink is a Web-based system[2] that provides automatic management and retrieval of a wide variety of information objects over an intranet or extranet. The size of the raw data is 7GB. The data set describes more than 3,000,000 requests made to a Livelink server from around 5,000 users. Each request corresponds to an entry in the log files, where each entry contains the following:

1. The IP address the user is making the request from,
2. The cookie of the browser the user is making request from, which can be as long as 5,000 bytes,

[1] A general principle for setting the threshold is to generate the number of sessions whose average length is in a reasonable range (say, 30 objects). However, more principled ways for setting the threshold remain to be investigated.

[2] Developed and sold by Open Text Corporation (http://www.opentext.com).

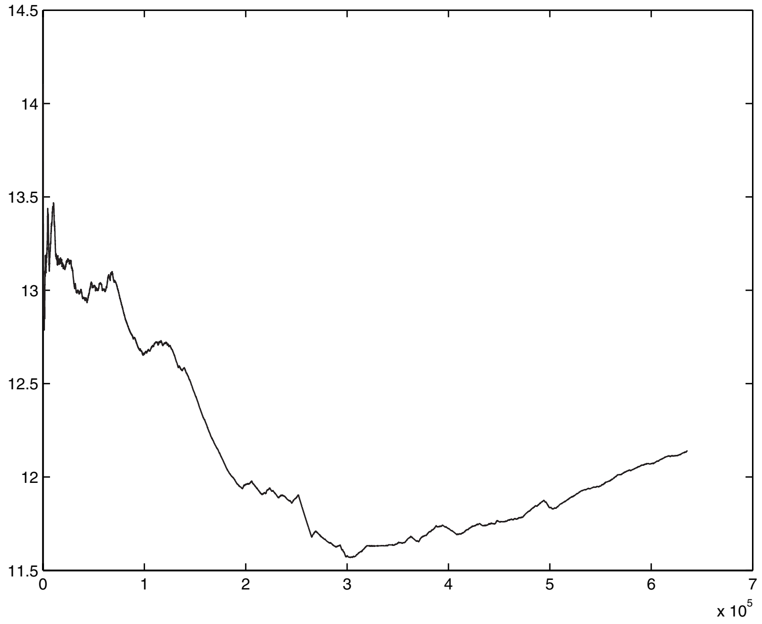


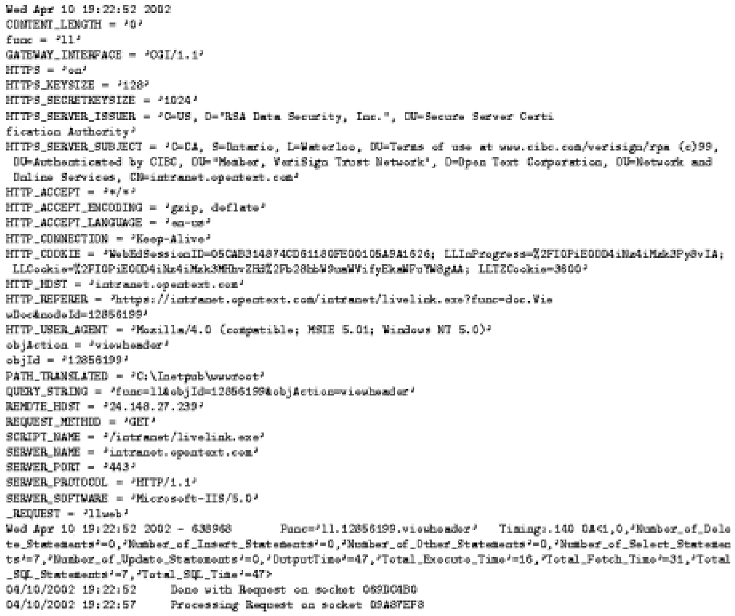
FIG. 1. Entropy evolution in our Web log data set.

3. The time the request is made and the time the required page is presented to the user,
4. The name of the request handler in the Livelink program,
5. The name of the method within the handler that is used to handle the request,
6. The query strings that can be used to identify the page and the objects being requested, and
7. Some additional information, such as URL addresses for error-handling.

A sample log entry of Livelink is shown in Figure 2. For privacy and security reasons, some of the lines are removed.

Data Preprocessing

Not all the information in a log entry is relevant to the task of learning usage patterns of Livelink. To extract relevant information, data preprocessing was conducted on the



```
Wed Apr 10 19:22:52 2002
CONTENT_LENGTH = '0'
func = '11'
GATEWAY_INTERFACE = 'CGI/1.1'
HTTPS = 'on'
HTTPS_KEYSIZE = '128'
HTTPS_SECRETKEYSIZE = '1024'
HTTPS_SERVER_ISSUER = 'C=US, O="RSA Data Security, Inc.", OU=Secure Server Certi
fication Authority'
HTTPS_SERVER_SUBJECT = 'C=CA, S=Ontario, L=Waterloo, OU=Terms of use at www.cibc.com/verisign/rpa (c)99,
 OU=Authenticated by CIBC, OU="Member, VeriSign Trust Network", O=Open Text Corporation, OU=Network and
 Online Services, CN=intranet.opentext.com'
HTTP_ACCEPT = '*/*'
HTTP_ACCEPT_ENCODING = 'gzip, deflate'
HTTP_ACCEPT_LANGUAGE = 'en-us'
HTTP_CONNECTION = 'Keep-Alive'
HTTP_COOKIE = 'WebEdSessionID=05CAB314874CD61180FE00105A9A1626; LLInProgress=%2FI0PiE00D4iNz4iMzk3Py8vIA;
 LLCookie=%2FI0PiE00D4iNz4iMzk3MHbvZHH%2Fb28bbW9uaWVifyEkaWFuYW8gAA; LLTZCookie=3600'
HTTP_HOST = 'intranet.opentext.com'
HTTP_REFERER = 'https://intranet.opentext.com/intranet/livelink.exe?func=doc.Vie
wDoc&nodeId=12856199'
HTTP_USER_AGENT = 'Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)'
objAction = 'viewheader'
objId = '12856199'
PATH_TRANSLATED = 'C:\Inetpub\wwwroot'
QUERY_STRING = 'func=ll&objId=12856199&objAction=viewheader'
REMOTE_HOST = '24.148.27.239'
REQUEST_METHOD = 'GET'
SCRIPT_NAME = '/intranet/livelink.exe'
SERVER_NAME = 'intranet.opentext.com'
SERVER_PORT = '443'
SERVER_PROTOCOL = 'HTTP/1.1'
SERVER_SOFTWARE = 'Microsoft-IIS/5.0'
_REQUEST = 'llweb'
Wed Apr 10 19:22:52 2002 - 638968       Func='11.12856199.viewheader'   Timing:.140 OA<1,0,'Number_of_Dele
te_Statements'=0,'Number_of_Insert_Statements'=0,'Number_of_Other_Statements'=0,'Number_of_Select_Statemen
ts'=7,'Number_of_Update_Statements'=0,'OutputTime'=47,'Total_Execute_Time'=16,'Total_Fetch_Time'=31,'Total
_SQL_Statements'=7,'Total_SQL_Time'=47>
04/10/2002 19:22:52     Done with Request on socket 069D04B0
04/10/2002 19:22:57     Processing Request on socket 09A87EF8
```

FIG. 2. A Livelink log entry.

raw log data. The following steps were performed to preprocess the data in our investigation:

1. Identifying the user from each log entry,
2. Extracting the time the request was made,
3. Identifying the information objects requested in each entry, and
4. Removing noisy entries (which request no interesting objects).

We use both IP addresses and some background knowledge to identify users in Livelink. The background knowledge that we use is the object hierarchy of the information maintained by Livelink. In Livelink, information objects are organized into a forest that contains over 2,000 trees. A leaf node of a tree corresponds to a document, such as a PDF file, a PPT file, a project description, a picture, and the like. A nonleaf node of a tree represents a folder that holds links to other folders or objects. According to our domain experts, about 10% of the trees are "public trees," which can be visited by all or some of the users. The other 90% of the trees are "private trees," which can only be visited by a single user. When identifying users, we first extract the IP address from each log entry and then combine the IP addresses that visited the same private tree into a single user. According to our domain experts, making use of IP addresses to identify users of Livelink is safer than using cookies. This is because, first, most often users access Livelink from the desktop in their office, and therefore most of the accesses are associated with a fixed IP address. Also, in Livelink, cookies are related to browser instances. Different browser instances invoked by the same user on the same machine have different cookies in Livelink logs. Therefore, instead of using cookies to identify users, IP addresses and object tree structures are used.

Another task, identifying objects from the large number of dynamic Livelink pages, is a unique part of the problem. An object could be a document (such as a PDF file), a folder, a picture, and so on. Different types of objects have different domains of identities. Based on Livelink domain knowledge, we can extract the identities of the objects being requested from the query string of the log entry. Most entries contain exactly one object, although some entries contain no objects or multiple objects. We ignore all entries that contain no information objects. The total number of different objects identified from the two-month Web log data is nearly 40,000. After these preprocessing steps, each entry contains the user ID, the time-stamp, and the IDs of the requested objects. The entries are then sorted according to the user ID first and the time-stamp as the second sorting key.

Session Identification

After the log entries are sorted by user ID and time-stamp, the next step is to identify session boundaries in each user's request sequence. We evaluate four kinds of session detection methods in the experiments. The first one uses the timeout method, in which we set the fixed time thresholds to be from 5 to 40 minutes in the experiments. The second one uses a reference length method, in which we treat an object on the leaf level of the object tree structure as a content page and a nonleaf object as an auxiliary page. A session ends after a user visits a content page. The third session identification method is the maximal forward reference method, in which a session ends right before an object already contained in the session is requested.

The last method we evaluate is the *n*-gram language-modeling-based method. In the experiments, we set *n* to be from 1 to 6. We also investigate the effectiveness of four different smoothing methods for the language-modeling-based session identification.

Evaluation Method

To determine whether the language-modeling-based method is effective, we conducted association rule learning from the sessions files generated by different session identification methods. We implemented the a priori algorithm (Agrawal & Srikant, 1994) to learn association rules from the presegmented Web log data. We then used these discovered association rules to evaluate the quality of our session detection method.

An association rule describes the association relationships between the information objects. For example, an association rule

$$\langle o1, o2, o3\rangle \rightarrow \langle o4, o5\rangle \text{ [support = 0.01 confidence = 0.6]}$$

means that 1% of the sessions contain objects *o1*, *o2*, *o3*, *o4*, and *o5*, and that 60% of the sessions containing *o1*, *o2*, and *o3* also contain *o4* and *o5*. The number of association rules generated from a session file depends on the *support* and *confidence* thresholds. For our data set, we found that the number of rules generated is not significantly affected by changing the confidence threshold. However, changing the support threshold affects the number of retrieved rules substantially. If the support threshold is set high, a small number of rules are generated and few of them are interesting. However, if we set the support threshold to be low, a huge number of rules are generated, which contain both interesting and uninteresting rules. Table 1 shows how the number of rules varies with the support threshold.

TABLE 1. Number of generated association rules (confidence threshold = 0.5).

Support threshold	0.02	0.01	0.008	0.005	0.003	0.0028	0.0025	0.002	0.001
Number of rules	2	14	39	88	723	4,556	74,565	4,800,070	$>10^9$

From Table 1, one can see that a large number of rules can be discovered if the support threshold is set very low. For evaluation purposes, to find interesting rules from a large number of discovered patterns, we rank the discovered rules according to some interestingness measure and prune redundant rules based on the structural relationship among rules (Huang, An, Cercone, & Promhouse, 2002a,b).

We considered five interestingness measures for the purpose of evaluating our new session detection method. These measures were used to measure the interestingness of an association rule $A \rightarrow B$,[3] as shown next.[4]

1. *C2* (Bruha, 1997). The C2 formula measures the agreement between A and B, and it has been evaluated as a good rule quality measure for learning classification rules (An & Cercone, 2001). It is defined as

$$C2 = \frac{P(B|A) - P(B)}{1 - P(B)} \times \frac{1 + P(A|B)}{2}$$

2. *Confidence (CS).* The confidence of a rule is expressed as $P(B|A)$, which denotes the probability that objects in B occur in a session conditioned on the occurrence of objects in A. With this measure, rules are ranked according to their confidence value as the main key and their support value as the secondary key. Therefore, this measure is denoted as CS.
3. *IS.* Derived from statistical correlation, the IS measure (Tan & Kumar, 2000) is defined as

$$IS = \frac{P(AB)}{\sqrt{P(A)P(B)}}$$

This measure is designed to be better suitable for situations where the support value of the rule is low.

4. *Measure of Discrimination (MD).* The MD measure (An & Cercone, 2001) was inspired by a query term weighting formula used in information retrieval, and it has been used to measure the quality of classification rules (An & Cercone, 2001). We adopt the formula to measure the extent to which an association rule $A \rightarrow B$ can discriminate between B and $\bar{B}$:

$$MD = \log \frac{P(A|B)(1 - P(A|\bar{B}))}{P(A|\bar{B})(1 - P(A|B))}$$

5. *Mutual information (MI).* The mutual information (Fano, 1961) between A and B is defined as

$$MI(A, B) = \log_2 \frac{P(AB)}{P(A)P(B)}$$

Informally, mutual information compares the probability of observing A and B together (the joint probability) with the probabilities of observing A and B independently.

The use of an interestingness measure can help identify interesting association rules by ranking the discovered rules according to the measure. After the rules are ranked according to an interestingness measure, we then evaluate the interestingness of the top-ranking rules by asking the domain experts in Open Text, who determine whether a rule is interesting or not according to the usefulness of the rule. In this way, for each ranked list of discovered association rules generated by an interestingness measure from a session file, we can calculate the percentage of the interesting rules in the top 10, 20, or 30 rules. We call this percentage *precision*. For example, if there are 6 interesting rules in the top 10 rules of a ranked list, the precision for the top 10 rules in this ranked list is 60%.

Experimental Results

Results of the timeout method. For the timeout method, we conducted experiments with a number of timeout thresholds: namely, 5, 10, 15, 20, 25, 30, 35, and 40 minutes. The results of these timeout methods in terms of precision in the top 10, 20, or 30 ranked association rules are shown in Tables 2, 3, and 4 respectively. The entries in these three tables represent the precision of the top n (where $n = 10$, 20, or 30) association rules discovered by each interestingness measure with respect to different time thresholds. The last row shows the average precision of the five interestingness measures. The best performance obtained in the top 10, 20, and 30 is **70%**,

TABLE 2. Top 10 precision of the timeout method.

	Time interval								
Interestingness Measure	5 min	10 min	15 min	20 min	25 min	30 min	35 min	40 min	Average
C2	30%	60%	60%	60%	60%	60%	60%	60%	56.25%
CS	20%	10%	10%	40%	40%	40%	40%	40%	30.00%
IS	30%	50%	50%	40%	50%	60%	50%	50%	47.50%
MD	60%	70%	70%	100%	100%	80%	80%	80%	80.00%
MI	100%	80%	80%	100%	100%	80%	80%	80%	87.50%
Average	48%	54%	54%	68%	**70%**	64%	62%	62%	60.25%

[3] In association rule, $A \rightarrow B$, A and B are sets of objects.

[4] We chose these interestingness measures because they have been proven to be among the best measures in our previous research (Huang, An, Cercone, & Promhouse, 2002).

TABLE 3. Top 20 precision of the timeout method.

Interestingness Measure	Time interval								Average
	5 min	10 min	15 min	20 min	25 min	30 min	35 min	40 min	
C2	20%	60%	55%	55%	60%	55%	60%	60%	53.125%
CS	15%	25%	20%	40%	25%	30%	30%	30%	26.875%
IS	25%	50%	40%	40%	40%	50%	50%	60%	44.375%
MD	50%	70%	70%	80%	90%	90%	90%	90%	78.750%
MI	100%	90%	90%	100%	100%	90%	90%	90%	93.750%
Average	42%	59%	55%	63%	63%	63%	64%	**66%**	59.375%

TABLE 4. Top 30 precision of the timeout method.

Interestingness Measure	Time interval								Average
	5 min	10 min	15 min	20 min	25 min	30 min	35 min	40 min	
C2	30%	56.7%	53.3%	50%	56.7%	63.3%	70%	73.3%	56.7%
CS	16.7%	26.7%	26.7%	40%	40%	40%	40%	40%	33.8%
IS	36.7%	60%	53.3%	80%	60%	60%	66.7%	63.3%	60.0%
MD	60%	80%	80%	100%	86.7%	86.7%	86.7%	86.7%	83.3%
MI	100%	93.3%	93.3%	100%	100%	93.3%	93.3%	93.3%	95.8%
Average	48.7%	63.3%	61.3%	**74%**	68.7%	68.7%	71.3%	71.3%	65.9%

66%, and **74%** under time thresholds 25, 40, and 20 minutes, respectively.

The performance of the standard timeout session detection method obviously depends on the time threshold. We find that generally time thresholds between 20 and 40 minutes are good. A threshold that is too small (e.g., 5 minutes) leads to poor performance. Figure 3 illustrates the influence of different time thresholds in the top 10, 20, and 30 results.

Results of the reference length and maximal forward reference methods. Table 5 shows the precisions of the top 10, 20, and 30 association rules for the reference length session identification method with respect to each of the five interestingness measures. The last row gives the average among all the interestingness measures. Table 6 shows the same results for the maximal forward reference session identification method. It can be observed that the two methods are not as good as the timeout method.

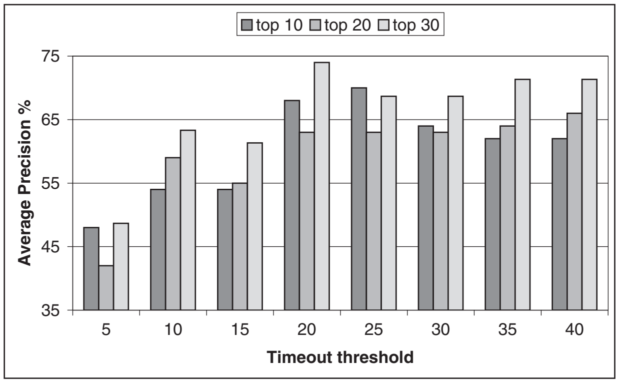


FIG. 3. Comparison of timeout thresholds.

Results of the language-modeling-based method. For the language-modeling-based methods, we experimented with 1-, 2-, 3-, 4-, 5-, and 6-gram models using four smoothing methods. Three different entropy thresholds are set for each model: namely, 0.0005, 0.0003, and 0.00025. The results for the top 10, 20, and 30 rules are shown in Tables 7, 8, and 9, respectively. In the tables, ABS, GT, LIN, and WB represent absolute discounting, Good-Turing discounting, linear discounting, and Witten-Bell discounting, respectively. Each entry in the three tables represents the average precision in the top 10, 20, and 30 rules averaged over five interestingness measures. The last row in the three tables gives the average results for each smoothing method. For example, the average result for top 10 ranking using GT smoothing is 85.88%.

TABLE 5. Results (precision) for the reference length method.

Interestingness measure	Top 10	Top 20	Top 30
C2	30%	45%	43%
CS	30%	35%	40%
IS	20%	40%	33%
MD	20%	50%	63%
MI	40%	55%	57%
Average	28%	45%	47.2%

TABLE 6. Results (precision) for the maximal forward reference method.

Interestingness measure	Top 10	Top 20	Top 30
C2	0%	0%	0.0%
CS	0%	5%	3.3%
IS	0%	0%	0.0%
MD	0%	0%	0.0%
MI	20%	20%	16.7%
Average	4%	5%	4.0%

TABLE 7. Top 10 precision of the language modeling method.

Order	Entropy threshold	ABS	GT	LIN	WB	Average
1-gram	0.00025	78%	76%	74%	74%	75.5%
	0.0003	76%	78%	72%	72%	74.5%
	0.0005	86%	88%	88%	88%	87.5%
2-gram	0.00025	74%	74%	76%	78%	75.5%
	0.0003	82%	82%	72%	70%	76.5%
	0.0005	28%	30%	0%	0%	14.5%
3-gram	0.00025	90%	80%	84%	84%	84.5%
	0.0003	84%	90%	78%	82%	83.5%
	0.0005	84%	82%	0%	80%	61.5%
4-gram	0.00025	94%	96%	80%	86%	89.0%
	0.0003	94%	96%	60%	80%	82.5%
	0.0005	80%	86%	0%	80%	61.5%
5-gram	0.00025	100%	98%	94%	96%	97.0%
	0.0003	98%	98%	90%	94%	95.0%
	0.0005	96%	100%	94%	96%	96.5%
6-gram	0.00025	98%	98%	86%	100%	95.5%
	0.0003	96%	98%	90%	98%	95.5%
	0.0005	96%	96%	76%	96%	91.0%
Average		85.22%	85.88%	67.44%	80.77%	79.83%

Figure 4 compares the language-modeling methods with the timeout, reference length, and maximal forward reference methods in terms of average precision. The average precision for the timeout method is taken from the 20-minute results, which are among the best results for the different time thresholds. One can observe that the average performance of the language-modeling-based methods, using either ABS, GT, or WB smoothing, is significantly better than the best performance of the timeout method, and it is also significantly better than the performance of the reference length and maximal forward reference methods. The performance of the linear (LIN) smoothing technique is comparable to the best performance of the timeout method, but it is significantly better than the reference length and maximal forward reference methods. Using a statistical t-test at the level of $\alpha = 0.001$, we find the following relationships in performance: $ABS = GT = WB >$ timeout optimal $= LIN >$ reference length $>$ maximal forward reference.

Analysis and Discussion

We analyze our experimental results in three respects: (1) we compare each of the different session identification methods, and we assess the effects of the (2) different smoothing methods and (3) the different n-gram orders on the language-modeling-based approach.

Our results indicate that the language modeling method is better than the timeout method, which is in turn better than

TABLE 8. Top 20 precision of the language modeling method.

Order	Entropy threshold	ABS	GT	LIN	WB	Average
1-gram	0.00025	81%	81%	81%	81%	81.00%
	0.0003	82%	84%	79%	79%	81.00%
	0.0005	83%	87%	87%	87%	86.00%
2-gram	0.00025	80%	80%	76%	77%	78.25%
	0.0003	88%	85%	73%	71%	79.25%
	0.0005	19%	20%	0%	0%	9.75%
3-gram	0.00025	89%	80%	83%	85%	84.25%
	0.0003	82%	87%	81%	86%	84.00%
	0.0005	85%	86%	0%	85%	64.00%
4-gram	0.00025	92%	90%	80%	85%	86.75%
	0.0003	87%	91%	33%	86%	74.25%
	0.0005	82%	85%	0%	83%	62.50%
5-gram	0.00025	99%	97%	95%	96%	96.75%
	0.0003	97%	97%	94%	97%	96.25%
	0.0005	97%	98%	97%	97%	97.25%
6-gram	0.00025	98%	97%	88%	97%	95.00%
	0.0003	97%	99%	94%	98%	97.00%
	0.0005	98%	97%	96%	96%	96.75%
Average		85.33%	85.61%	68.72%	82.55%	80.55%

TABLE 9. Top 30 precision of the language modeling method.

Order	Entropy threshold	ABS	GT	LIN	WB	Average
1-gram	0.00025	85.33%	84.67%	84.67%	84.67%	84.84%
	0.0003	84.67%	84.00%	82.67%	82.67%	83.50%
	0.0005	88.00%	85.33%	85.33%	85.33%	86.00%
2-gram	0.00025	83.33%	80.67%	76.67%	78.00%	79.67%
	0.0003	85.33%	81.33%	75.33%	70.00%	78.00%
	0.0005	19.33%	20.67%	0.00%	1.33%	10.33%
3-gram	0.00025	85.33%	78.67%	81.33%	84.00%	82.33%
	0.0003	82.00%	84.67%	84.67%	84.00%	83.33%
	0.0005	85.33%	84.00%	0.00%	88.00%	64.33%
4-gram	0.00025	92.00%	92.00%	70.67%	84.00%	84.67%
	0.0003	89.33%	89.33%	24.67%	84.00%	71.83%
	0.0005	82.67%	86.67%	0.00%	84.67%	62.50%
5-gram	0.00025	97.30%	97.30%	95.30%	97.30%	96.83%
	0.0003	98.00%	98.00%	94.70%	97.30%	97.00%
	0.0005	98.70%	97.30%	95.30%	97.30%	97.17%
6-gram	0.00025	98.00%	98.00%	92.00%	97.30%	96.33%
	0.0003	96.70%	97.30%	94.70%	98.00%	96.67%
	0.0005	98.00%	97.30%	85.30%	98.00%	94.67%
Average		86.07%	85.40%	67.96%	83.10%	80.63%

the reference length method, which is further better than the maximal forward reference method. In our experiments, the maximal forward reference method performs poorly because the assumption that a session ends at every backward reference is incorrect. According to our domain experts, backward references often take place within a single session, and therefore the maximal forward reference method tends to break a session into too many small pieces. As for the reference length method, the reason for its poor performance is that the assumption that only one content page occurs in each session is inappropriate. In Livelink, it is common for a leaf-level information object to be related to some other (leaf-level) objects, and therefore during a single session a user may request several related objects. The timeout method performed better than these two techniques in our experiments. However, the strategy of assuming that a fixed time interval threshold defines session boundaries is also problematic. Clearly, in reality, users do not take a fixed amount of time between sessions. A user may continue to a different topic immediately after completing a task. An advantage of the language-modeling-based approach is that it does not incorporate any of these fixed assumptions about what constitutes a boundary between sessions. Instead, it uses a much more dynamic criterion that detects changes in information in the sequence of requested objects.

With respect to the smoothing technique for the language modeling method, we observe that generally absolute discounting (ABS) and Good-Turing discounting (GT) are a bit better than Witten-Bell discounting (WB). All three smoothing techniques are significantly better than linear discounting (LIN). However, their performance also depends on the order of the language model. For a 1-gram language model, the performances of the four smoothing techniques are all similar. If we look at the entropy evolution curves for

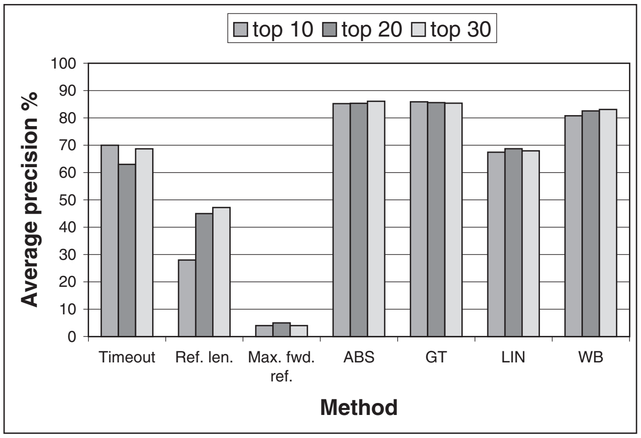


FIG. 4. Comparison of smoothing methods for language modeling against timeout.

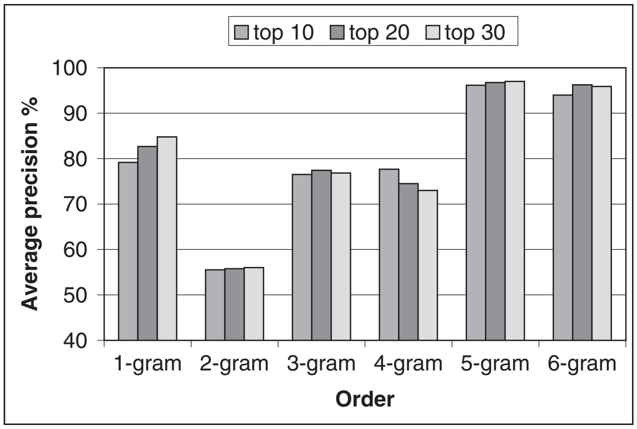


FIG. 5. Comparison of *n*-gram language models with different choices of *n*.

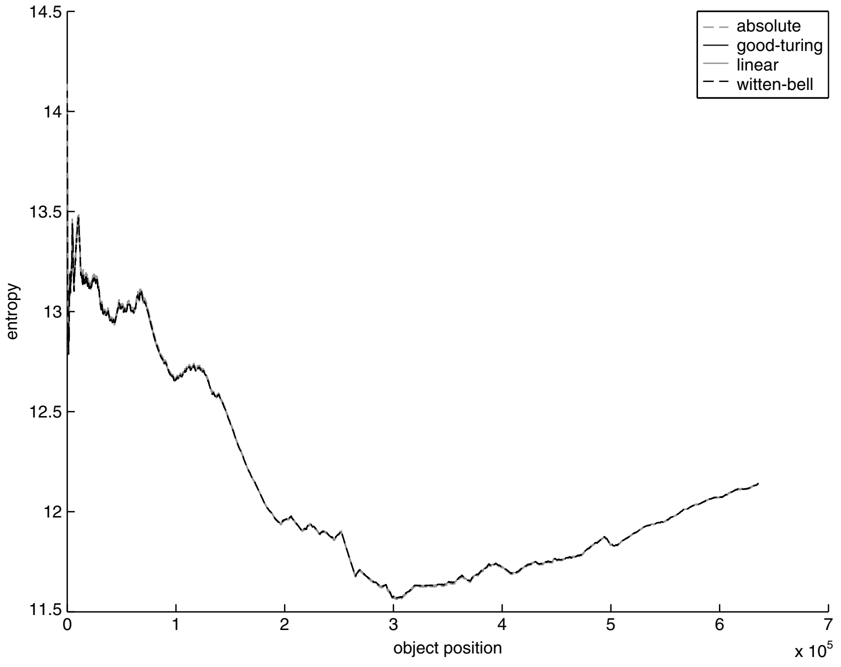


FIG. 6. Entropy evolution curves for the language-modeling method with $n = 1$.

1-gram models (see Figure 6), we can observe that the curves for the four smoothing methods overlap and the entropy changes greatly along the sequence. The performance of the four smoothing methods for 2-gram models is also similar (see Tables 7, 8, and 9). Looking at Figure 7, one can observe that the curves for the four smoothing methods are all pretty flat, which explains why the performance in this case is not very effective. However, the performance of the four smoothing methods for 3-, 4-, 5-, and 6-gram models are distinguishable (see Tables 7, 8, and 9), especially for the 3- and 4-gram models. Figures 8, 9, 10, and 11 show the entropy evolution curves for the 3-, 4-, 5-, and 6-gram models, respectively. It can be observed that the curves for ABS and GT smoothing change more rapidly than the curves for the WB and LIN methods. Between the curves for WB and LIN methods, the curve for LIN is flatter than that for WB. This explains why ABS and GT perform better than WB, and why LIN is the worst performing method among smoothing techniques.

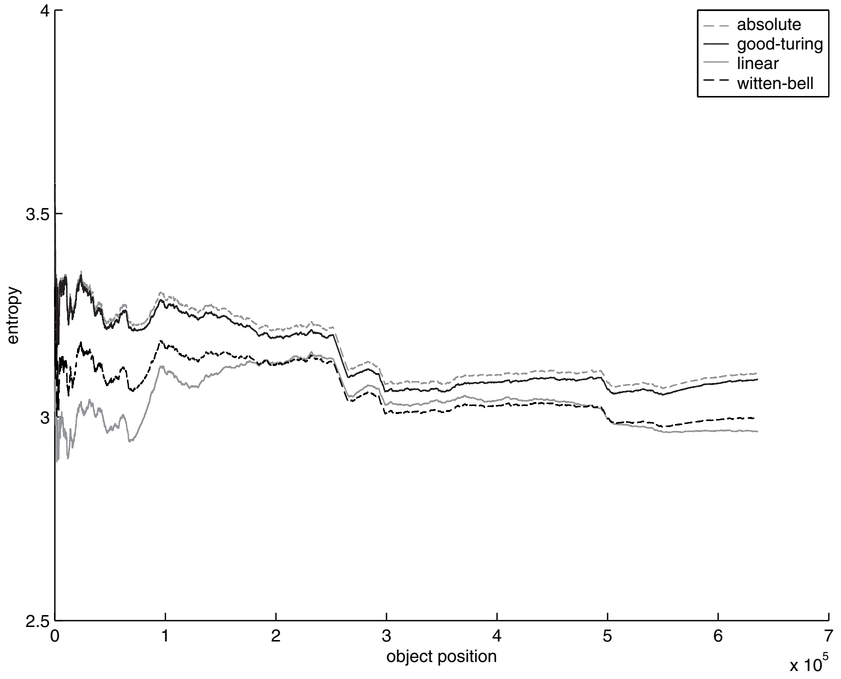


FIG. 7. Entropy evolution curves for the language-modeling method with $n = 2$.

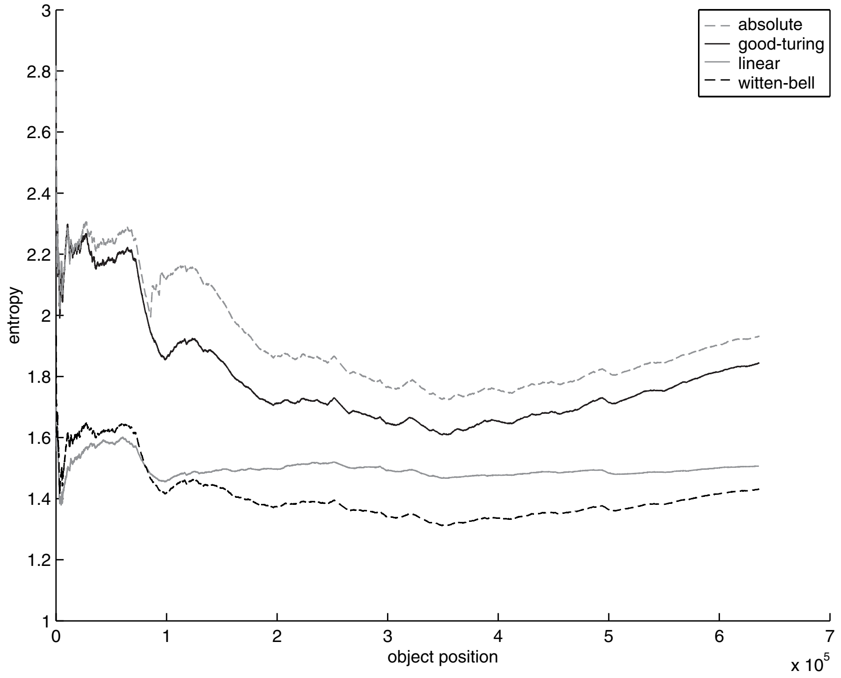


FIG. 8. Entropy evolution curves for the language-modeling method with $n = 3$.

Finally, to determine the effect of n-gram order on performance, we compare the average performance of n-gram models with different choices of n; see Figure 5. This figure shows that the 5-gram model performs the best, with the 6-gram model a close second, followed by the 1-, 3-, 4-, and 2-gram models, in that order. This result indicates that the optimal performance of an n-gram model for session detection is achieved by a value of n that is neither too small nor too big.

Implications of Findings for Web Design

In this article, we proposed a novel session identification method based on statistical language models. We demonstrated through a series of experiments that the proposed method is superior to three existing session identification methods. Because session identification is a crucial data-processing step for Web log mining, our proposed method has indirect impact on Web design by providing better, more

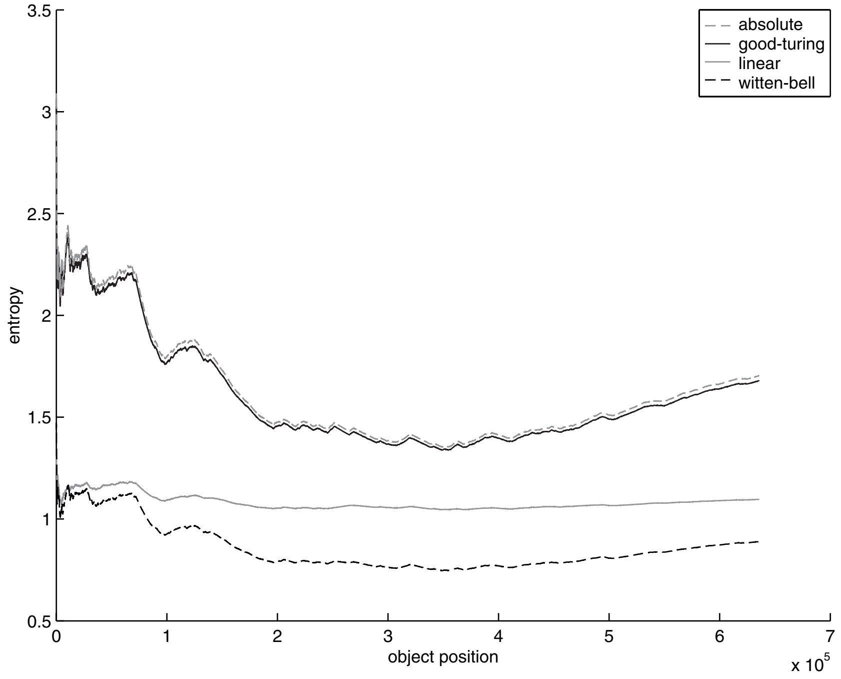


FIG. 9. Entropy evolution curves for the language-modeling method with $n = 4$.

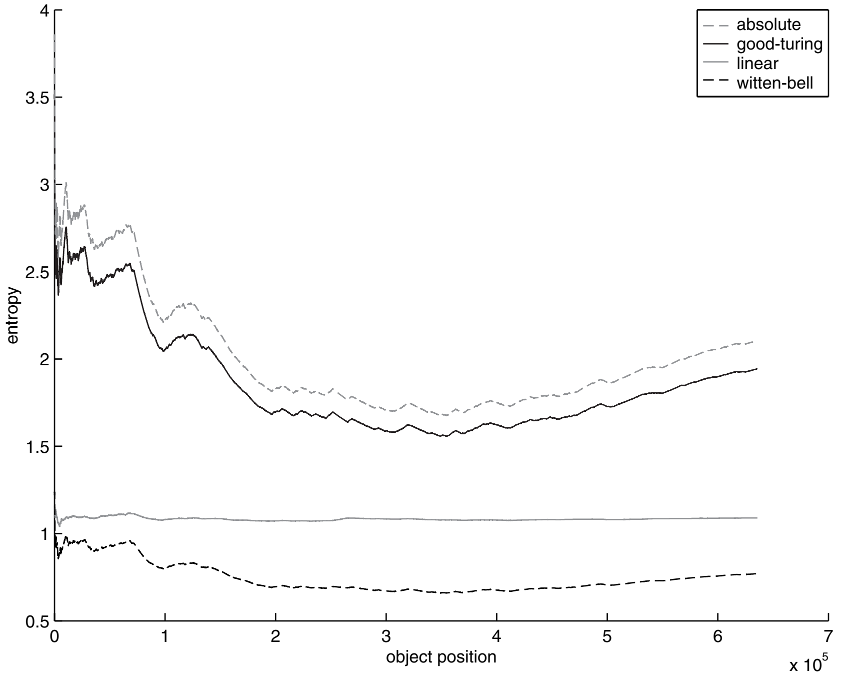


FIG. 10. Entropy evolution curves for the language-modeling method with $n = 5$.

accurate data for Web log mining. Its importance can be illustrated as follows in terms of what Web log mining can do for Web design.

Web log mining can be used to reorganize a Web site to better serve the users by finding patterns that relate pages frequently visited together. Those highly related pages should be directly linked together; or their content should be put into one page so that users can directly obtain the needed information. However, some existing direct links should be dropped if they connect unrelated pages. Web log mining can also be used to provide personalized Web service by discovering browsing patterns and interests of one or a group of users. The discovered patterns can be used to tailor Web pages to users' individual preferences, make personalized recommendations, and let users bypass irrelevant content. Personalization makes it easier and more pleasant for users

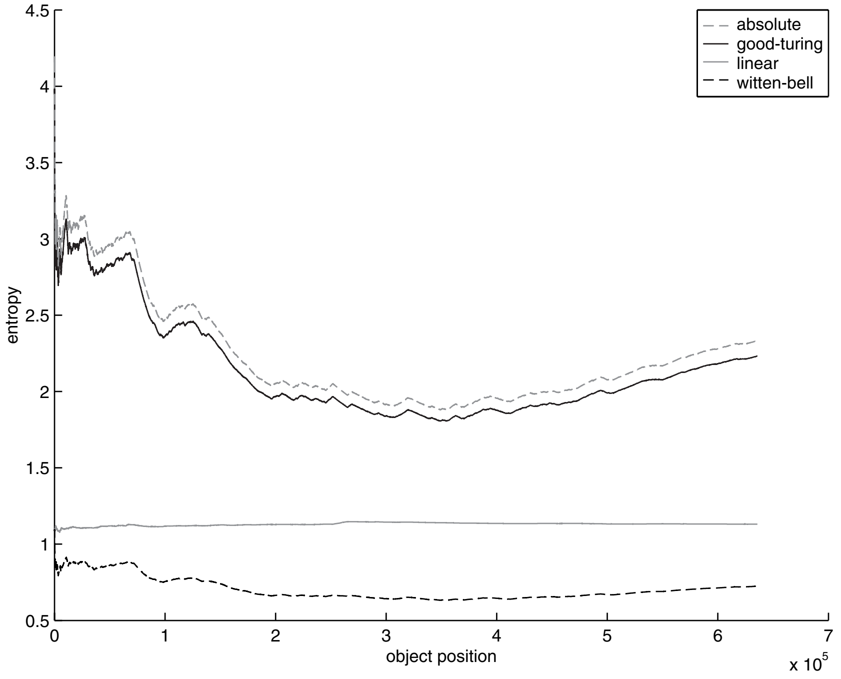


FIG. 11. Entropy evolution curves for the language-modeling method with $n = 6$.

to surf the Web and find what they want. For e-commerce Web sites, such service can lead to more customer loyalty and thus increase the profitability of the Web site.

Web log mining can also be used to enhance the performance of Web-caching systems (Yang & Zhang, 2003). The idea behind Web caching is to maintain a small set of retrieved Web pages in a local cache or a proxy server so that the system performance can be improved by answering users' later requests from the cache. A key issue in a caching system is its page replacement policy, which specifies conditions under which a new page will replace an existing one. In (Yang & Zhang, 2003), Web log mining is used to learn frequent access patterns that can be used to predict future Web requests. The prediction is then used to select the pages to be replaced in a cache when a request arrives. Web log mining can also be used to improve the performance of Web search by reranking the retrieved pages with mined patterns (Xue, Zeng, Chen, Ma, & Lu, 2002). All of these cannot be accomplished without a good session identification method.

Conclusions and Future Work

We have proposed a novel approach for session boundary detection based on statistical *n*-gram language modeling. Our approach is based on information theory and is intuitively understandable. Experiments on learning interesting association rules from the Livelink data set show that we obtain consistent improvements over the traditional timeout, reference length, and maximal forward reference methods. Our experiments also show that absolute smoothing, Good-Turing smoothing and Witten-Bell smoothing are effective smoothing techniques to use with the language modeling method for session boundary detection.

Some questions are still open for further investigation. For example, we have found that performance of the language-modeling-based approach is sensitive to the entropy threshold. A threshold value that is either too big or too small will give nonoptimal performance. An automatic threshold setting method should be investigated. Our future work also includes investigating the effectiveness of the language modeling approach on other data sets, and investigating the effectiveness of the approach for other Web usage mining problems, such as sequential pattern mining.

Acknowledgments

We would like to thank Gary Promhouse of Open Text for spending time on evaluating the discovered association rules in the reported experiments. Without his help and useful feedback, this research cannot be fully conducted. We would also like to thank Nick Cercone for his valuable comments on this work. This research is supported in part by Open Text Corporation and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Databases, Santiago, Chile (pp. 487–499). New York: ACM Press.

An, A., & Cercone, N. (2001). Rule quality measures for rule induction systems: Description and evaluation. Computational Intelligence, 17(3), 409–424.

Bahl, L., Jelinek, F., & Mercer, R. (1983). A maximum likelihood approach to continuous speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(2), 179–190.

Bruha, I. (1997). Quality of decision rules: Definitions and classification schemes for multiple rules. In G. Nakhaeizadeh & C.C. Taylor (Eds.), Machine learning and statistics, the interface (pp. 107–131). New York: Wiley.

Burton, M., & Walther, J. (2001). The value of Web log data in use-based design and testing. Journal of Computer-Mediated Communication, 6(3).

Catledge, L., & Pitkow, J. (1995). Characterizing browsing strategies in the World Wide Web. Proceedings of the Third International World Wide Web Conference, Darmstadt, Germany (pp. 1065–1073). New York: ACM Press.

Chen, S., & Goodman, J. (1998). An empirical study of smoothing techniques for language modeling (Tech. Rep. No. TR-10–98). Cambridge, MA: Harvard University.

Chen, M.S., Park, J.S., & Yu, P.S. (1998). Efficient data mining for path traversal patterns. IEEE Transactions on Knowledge and Data Engineering, 10(2), 209–221.

Cooley, R., Mobasher, B., & Srivastava, J. (1999). Data preparation for mining World Wide Web browsing patterns. Knowledge and Information Systems, 1(1), 5–32.

Duy, J., & Vaughan, L. (2003). Usage data for electronic resources: A comparison between locally collected and vendor-provided statistics. Journal of Academic Librarianship, 29(1), 16–22.

Fano, R. (1961). Transmission of information: A statistical theory of communications. Cambridge, MA: MIT Press.

He, D., & Goker, A. (2000). Detecting session boundaries from Web user logs. Proceedings of the 22nd Annual Colloquium on Information Retrieval Research (pp. 57–66). Cambridge, UK: British Computer Society.

Huang, X., An, A., Cercone, N., & Promhouse, G. (2002a). Discovery of interesting association rules from livelink Web log data. Proceedings of the IEEE International Conference on Data Mining (ICDM '02) (pp. 763–766). Maebashi City, Japan: IEEE Computer Society.

Huang, X., An, A., Cercone, N., & Promhouse, G. (2002b). Comparison of interestingness functions for learning Web usage patterns. Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM '02) (pp. 617–620). McLean, VA: ACM Press.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing, 35(3), 400–401.

Tan, P., & Kumar, V. (2000). Interestingness measures for association patterns: A perspective (Tech. Rep. No. TR00–036). University of Minnesota, Department of Computer Science, Minneapolis.

Xue, G., Zeng, H., Chen, Z., Ma, W., & Lu, C. (2002). Log mining to improve the performance of site search. WISE Workshops 2002, pp. 238–245.

Yang, Q., & Zhang, H. (2003, July/August). Web-log mining for predictive Web caching. IEEE Transactions on Knowledge and Data Engineering, 4, 1050–1053.