WebSail: From On-line Learning to Web Search

Zhixiang Chen¹, Xiannong Meng², Binhai Zhu³, Richard H. Fowler¹

¹Department of Computer Science, University of Texas-Pan American, Edinburg, Texas, USA
²Department of Computer Science, Bucknell University, Lewisburg, Pennsylvania, USA
³Department of Computer Science, Montana State University, Bozeman, Montana, USA

Abstract. In this paper we report our research on building WebSail, an intelligent web search engine that is able to perform real-time adaptive learning. WebSail learns from the user’s relevance feedback, so that it is able to speed up its search process and to enhance its search performance. We design an efficient adaptive learning algorithm TW2 to search for web documents. WebSail employs TW2 together with an internal index database and a real-time meta-searcher to perform real-time adaptive learning to find desired documents with as little relevance feedback from the user as possible. The architecture and performance of WebSail are also discussed.

Keywords: Adaptive learning; Document ranking; Relevance feedback; Vector space; Web search

1. Introduction

In this paper, we investigate the applicability of adaptive learning algorithms to web search. We design a tailored version TW2 of the well-known algorithm Winnow2 (Littlestone, 1988) for the particular purpose of web search. We have implemented a real-time adaptive web search learner WebSail¹ with TW2 as its learning component. WebSail learns from the user’s relevance feedback and helps the user to search for the desired documents with as little relevance feedback as possible. WebSail has been implemented on a Sun Ultra One workstation with storage of 27 Gb hard disk on an IBM R6000 workstation. It has an internal index database of about 800,000 documents and a meta-search component through AltaVista². Each document in the internal database is indexed using about 300 keywords. When the user performs a search process, WebSail first searches its

¹ WebSail: www.cs.panam.edu/chen/WebSearch/WebSail.html.
internal database. If no matches can be found for the query within the internal database, then it turns to its meta-search component to receive the matched documents through AltaVista and then performs the learning process locally. There have been considerable efforts applying machine learning to web search-related applications, for example, scientific article locating and user profiling (Bollacker et al., 1998; Lawrence et al., 1999), and collaborative filtering (Nakamura and Abe, 1998; Billsus and Pazzani, 1998). Our work related to this paper can be found in Chen et al. (1999), Chen and Meng (2000) and Chen et al. (2001).

The rest of the paper is organized as follows. In Section 2, we examine the property of web document indexing and design the learning algorithm TW for web search. In Section 3, we discuss practical issues such as document ranking and equivalence query simulation regarding the actual employment of TW as a learning component in WebSail. We also discuss the architecture and performance of WebSail. We conclude the paper in Section 4.

2. The Algorithm TW

When a set of \( n \) binary-valued index terms (or keywords) \( T_1, \ldots, T_n \) are used to index web documents, a document is represented as a vector in the \( n \)-dimensional binary vector space \( \{0, 1\}^n \). Given any document \( d \), let \( v_d(x_1, \ldots, x_n) \) denote its vector representation, where for \( i = 1, \ldots, n \), \( x_i \) is a binary-valued variable for the index term \( T_i \). If \( x_i = 1 \) in the vector \( v_d \), then the document \( d \) has the index term \( T_i \), otherwise \( d \) does not. Define

\[
\begin{align*}
MD[x_i, \ldots, x_s] &= \{d | v_d(x_1, \ldots, x_n) \Rightarrow x_i \vee \cdots \vee x_s\}, \\
MD(k) &= \{MD[x_i, \ldots, x_s] | 1 \leq s \leq k\}
\end{align*}
\]

In other words, \( MD[x_i, \ldots, x_s] \) is a collection of documents whose vectors satisfy the monotone disjunction of \( x_i, \ldots, x_s \), and \( MD(k) \) is the class of all collections of documents represented by monotone disjunctions of at most \( k \) relevant index terms. Many theoretically efficient algorithms exist for learning disjunctions of at most \( k \) relevant index terms. But few are applicable to web search, because a user may have no patience to try, say, more than 10 iterations of learning. The challenging question for us is whether we can design an efficient learning algorithm that can yield significant search precision increase with about four to five iterations of learning from the user’s relevance feedback. Moreover, the algorithm must be practically efficient to process its learning tasks.

**Definition 2.1.** Given any index term \( x \) and any document \( d \), \( x \) is said to be an index term for the document \( d \) if \( d \) has \( x \), or in other words, the corresponding component of the index term \( x \) in the document vector \( v_d \) is 1.

Although a huge collection of index terms (in the simplest case, keywords) is needed and used to index web documents, for each particular document \( d \), the number of its index terms is relatively small. One can easily note that a web hypertext document may have several hundreds of distinct keywords, while a good dictionary may have over 100,000 words. One may argue that there are long documents. This is certainly true. But as far as indexing is concerned, not all words in a long document are needed to index it. Instead, a small portion of the words may be used. To the end, indexing is closely related to classification.