INTELLIGENT QUERY AND BROWSING INFORMATION RETRIEVAL (IQBIR) AGENT

Jong-Min Park

Department of Electrical and Computer Engineering
University of Wisconsin - Madison
1415 Engineering Drive
Madison, WI 53706-1691 USA

ABSTRACT

Reported in this paper is an intelligent agent that aids users to conduct efficient Internet Web information retrieval through query formulation, information collection, information clustering, and analysis. The underlying mechanism is a probabilistic Sample-at-the-boundary learning algorithm for clustering the search results and learning and matching the user concept. Kohonen's "windowed" Learning Vector Quantization algorithm is shown to be related to this Sample-at-the-boundary learning algorithm. A prototype system has been developed and evaluation has been conducted.

1. INTRODUCTION

Due to the tremendous popularity and growth of the heterogeneous information on the Internet that is dynamically increasing ([4]), it is difficult to index and categorize the Internet documents efficiently to speed up searching and browsing by broad spectrum of users for newly generated concepts and interests. This is apparent as the internal representation of the information retrieval systems may not always match the user's concept of interest, thus requiring several cycles of query and possibly user feedback to match the retrieved results to the user's concept of relevance.

Some of the related works in closing the gap between the user concept and retrieved documents from dynamically increasing information, especially on the Internet and the Web, include intelligent Web browsing agents (e.g. [6]) for assisting the user to efficiently browse the web pages, meta-search engines (e.g. [8]) that are interface agents to multiple search engines and Internet resources, and the user concept matching for Lotus Notes database ([3]).

In this work, a novel intelligent query and browsing information retrieval framework is developed to aid in search and learning of user concept from multiple existing information retrieval systems. The framework is applied to accessing multiple Web search engines in parallel.

The goal of the system is to reduce the number of query cycle for efficient query and retrieval, learn to match the user concept to increase accuracy in classifying documents relevant to the user, and aid in the browsing process, by clustering the results and actively learning from user feedback.

In the sections that follow, the intelligent query and browsing framework is described, the underlying probabilistic Sample-at-the-boundary method ([5]) is formulated, and Kohonen's LVQ 2.1 learning algorithm ([1]) is shown to be a special case of the learning method, followed by evaluation and conclusion.

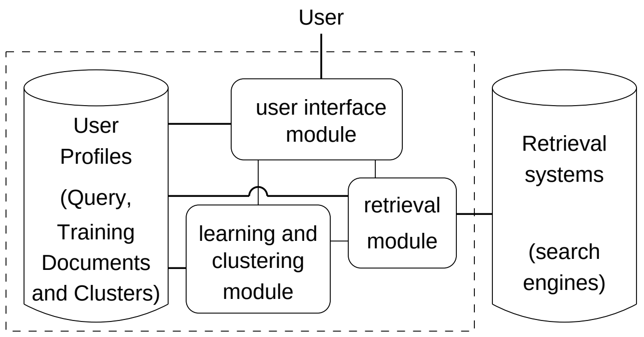

Figure 1: Intelligent Query and Browsing System Framework

2. SYSTEM FRAMEWORK

Figure 1 shows the model of the intelligent query and browsing system, consisting of the following components: User query and browsing interface module, Information retrieval module, Clustering and learning module, and User Profile.

The user query interface and the result browsing interface allows access control, query entry, result display, document browsing, and user feedback.

The browsing module generates a display of the clusters generated by the learning and clustering module. The user may click on a representative cluster from a list of representatives to show a list of documents within the cluster, and retrieve full content of a document.

The retrieval module accesses multiple information repositories (web search engines in this case) using a given query (lists of terms for search engines) and collects the results of documents or summaries of documents. Documents are parsed and mapped into training feature vectors for subsequent clustering and learning.

The clustering module groups collected documents into a fixed number of clusters, with a representative document and a codebook on each cluster.

User relevance feedback are recorded for subsequent on-line training by the learning module. Feedback may be given both for the clusters and for individual documents using one of three labels, “relevant”, “irrelevant”, and the default “dont’ care”.

Terms that occur together with the query terms in each cluster are extracted and displayed as suggestions for further refinement to the query, and possible entry to a personal thesaurus and a personal list of stop-words.

Each user profile consists of a set of query searches for interests or concepts to be searched for a user. It contains the user-supplied initial query, the reformulated query, codebook clusters, and sample documents for on-line training and fine-tuning.

3. CLUSTERING AND LEARNING

Documents are collected by querying and retrieving results from multiple search engines. These search results usually consist of URL (Universal Resource Locator) and titles of web pages in a ranked list, including a summary or a contextual content of each web page, which are parsed accordingly and stored locally.

The documents are represented using the term vector model with tf-idf weighting (term-frequency inverse document frequency weighting) ([7]). Terms are pruned by removing stop-words and stemming. Contextual term weighting is applied, where higher weight is assigned to terms occurring in the title and the URL, while lower weight is given to terms within the summary or partial context.

To minimize the clustering time, the large dimensionality of the term vector space, and the network congestion, only the title and the summary list from the search engines are used as documents, while avoiding access to the full content of the web pages. The length of the vector is limited to a fixed number of terms having highest tf-idf weights.

Initial collection from a new query is clustered using the Kohonen’s SOM (Self-Organizing Map, [1]) with a fixed number of codebooks in a two-dimensional mapping. Documents with labels from user feedback are used for training the codebooks using the Sample-at-the-boundary learning method ([5]).

Subsequent results by additional queries for the same concept search are classified and clustered accordingly by the codebooks.

3.1. Boundary Sampling and LVQ 2.1

To re-train the codebooks given the user feedback, the active Sample-at-the-Boundary method (theoretical formulation for the method is found in [5]) is applied, where existing samples are chosen that are near the boundary with Gaussian probability distribution instead of generating new samples.

Let the decision boundary satisfy $\vec{w}^* \cdot \vec{x} = w_0$, where $\vec{w}^*$ is the true boundary vector, w_0 is the threshold, and $\vec{x}$ is a feature vector.

Given a sample $\vec{x}^{(n)}$ chosen near the boundary with the label $y^{(n)}$, the update of the boundary estimate and the threshold at step $n+1$ is

$$\begin{aligned} \vec{w}^{(n+1)} &= \vec{w}^{(n)} + \epsilon u(y^{(n)})\vec{x}^{(n)} && (1)\\ w_0^{(n+1)} &= w_0^{(n)} - \epsilon u(y^{(n)}), && (2) \end{aligned}$$

where $u(y^{(n)}) = 1$ if the label $y^{(n)}$ matches the side pointed to by the boundary estimate vector $\vec{w}^{(n)}$, and -1 otherwise.

The active Sampling-at-the-Boundary learning method will update each decision boundary locally, thus effectively fine-tuning the classification by considering one boundary at a time without the complexity of multiple clusters and higher dimensions. This is applied to clusters with codebooks by defining each boundary to be a piecewise-linear hyper-plane at the center of two codebook vectors of differing label.

Define two codebook vectors of differing class labels, $\vec{c}_A^{(n)}$ and $\vec{c}_B^{(n)}$, as the decision boundary estimate. The next training sample, $\vec{x}^{(n)}$, is chosen near the boundary as described later, and has the same class label with either codebook vector, with $u(y^{(n)}) = 1$ if the label is the same as $\vec{c}_A^{(n)}$ and $u(y^{(n)}) = -1$ otherwise.

The codebook vectors are updated as

$$\begin{aligned} \vec{c}_A^{(n+1)} &= \vec{c}_A^{(n)} + \epsilon u(y^{(n)})[\vec{x}^{(n)} - \vec{c}_A^{(n)}] && (3)\\ \vec{c}_B^{(n+1)} &= \vec{c}_B^{(n)} - \epsilon u(y^{(n)})[\vec{x}^{(n)} - \vec{c}_B^{(n)}]. && (4) \end{aligned}$$

We now show that this method is related to the active Sampling-at-the-Boundary method. We also show that LVQ 2.1 is a special case of this learning method.

Current boundary estimate and threshold defined by these two codebook vectors are

$$\begin{aligned} \vec{w}^{(n)} &= \vec{c}_A^{(n)} - \vec{c}_B^{(n)}, && (5)\\ w_0^{(n)} &= \frac{1}{2}[\vec{c}_A^{(n)} + \vec{c}_B^{(n)}] \cdot \vec{w}^{(n)} && (6) \end{aligned}$$

and the new boundary estimate and the threshold for the updated codebooks are thus

$$\vec{w}^{(n+1)} = \vec{c}_A^{(n+1)} - \vec{c}_B^{(n+1)}, \qquad (7)$$

$$w_0^{(n+1)} = \frac{1}{2}[\vec{c}_A^{(n+1)} + \vec{c}_B^{(n+1)}] \cdot \vec{w}^{(n+1)} \quad (8)$$

Translating the mid-point between the initial codebooks,

$$\vec{m}_{AB}^{(n)} = \frac{1}{2}[\vec{c}_A^{(n)} + \vec{c}_B^{(n)}],$$

to be on the origin of the coordinate system, the sample point $\vec{x}^{(n)}$ is translated to

$$\vec{x_t}^{(n)} = \vec{x}^{(n)} - \vec{m}_{AB}^{(n)},$$

the vector $\vec{w}^{(n)}$ remains the same, and the threshold $w_0^{(n)}$ is translated to $w_{t,0}^{(n)} = 0$.

Applying equations (3)-(4) to (7)-(8) using the translated vectors,

$$\vec{w}^{(n+1)} = \vec{w}^{(n)} + \epsilon_t u(y^{(n)})\vec{x_t}^{(n)} \quad (9)$$
$$w_{t,0}^{(n+1)} = w_{t,0}^{(n)} - \epsilon_t^* u(y^{(n)}). \quad (10)$$

where $\epsilon_t = 2\epsilon$ and $\epsilon_t^* = \epsilon_t * \{\frac{1}{4}|\vec{w}^{(n)}|^2\}$. Thus it is closely related to actively updating the boundary estimate in the equations (1)-(2).

In this particular prototype, samples are chosen near the boundary with normal probability distribution,

$$\exp^{(1-r)^2/\sigma^2} > U$$

where $r = distance_{small}/distance_{large}$ ($distance_{small}$ and $distance_{large}$ are the distance to the closest codebook vector and the next closest codebook vector, respectively, as shown in Figure 2, left), and U is a uniform random variable in the range $[0, 1)$ (Figure 2, right).

Since the LVQ 2.1 uses a window around the boundary using the formula

$$distance_{small}/distance_{large} > R_{ratio},$$

LVQ 2.1 is related to a special case of Sampling-at-the-boundary method, where samples are chosen from a uniform distribution within a window around the boundary estimate, in contrast to normal distribution.

4. PERFORMANCE EVALUATION

The system performance is measured using the standard *precision* and *recall* ratios that is widely used in the information retrieval field. The usability performance is measured by computing the *search length* defined as the number of "irrelevant" or "don't care" documents that have been traversed before reaching a certain number of "relevant" documents.

To compare with existing search engines, a fixed number of documents (40 documents) from the retrieved list from each search engine are combined as the full document space. Three search engines and one meta search engine are used for comparison. Ten sample query sessions are recorded for each search engine.

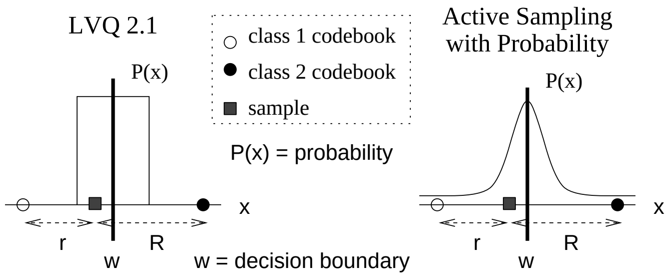

Figure 2: LVQ2.1 and normal distribution sampling

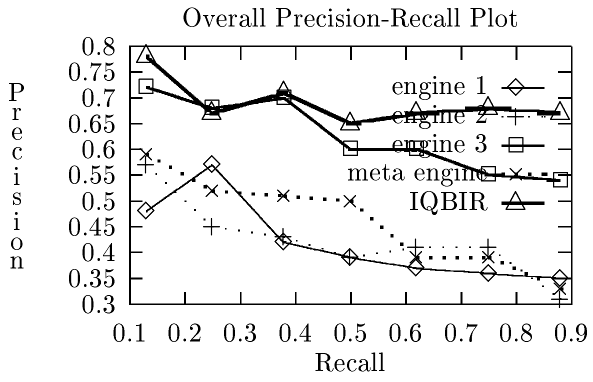

Figure 3: Overall Precision-Recall Plot

For the IQBIR system, the ranking of documents is assumed to be in the order of traversal of clusters, that is, documents within the cluster that has been accessed first are ranked first, and so on. Documents within each cluster are assumed locally ranked. For all other search engines, including the meta search engine, the single list result is assumed ranked from top to bottom. The resulting plot is shown in Figure 3.

Since the existing search engines do not include learning, the system improvement over time could not be compared. Further work on the evaluation framework for learning agents will be needed for more objective comparison.

The *expected search length* are computed by averaging over all query sessions for several fixed values of a parameter, in this case a number of relevant documents browsed. The search length here is defined as the number of "irrelevant" or "don't care" documents that had to be traversed (and also given user feedback) to reach certain number of "relevant" documents.

The number of documents traversed within each set are then recorded for each session. These values are averaged

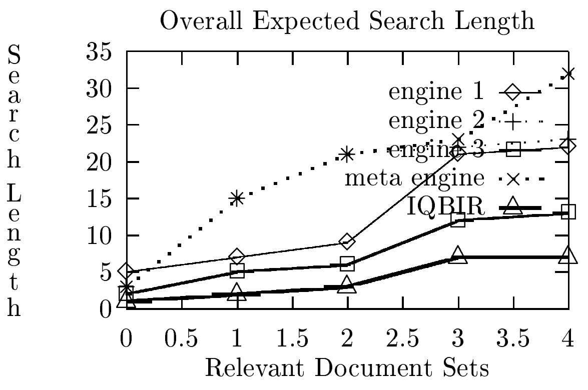

Figure 4: Overall Expected Search Lengths

over all query sessions for each set and plotted in Figure 4. This plot shows the user effort of traversing through the documents and giving user feedback in terms of search lengths (the lower the value, the better).

The overall precision-recall plot shows the clustering system improves upon existing search engines by allowing the user to only look at the documents within clusters that are relevant, thus reducing user effort in both browsing and user feedback required to re-train the system.

The overall search length plot also empirically shows that the number of traversal and user feedback that needs to be done are reduced through clustering and learning.

The meta search engine does not improve much as it still gives a static combined result of multiple search engines. The meta search engine tested here is an improvement over other meta search engines that re-ranks the combined result into another single ranked list. It uses a simple single group clustering by the terms presented, and several group clustering by the URL address (geographical location). It still lists the groups into one long list.

Although not presented as a figure, the "miscellaneous" group was found to be very useful. In all query sessions tested, none of the relevant documents fell into the group, and almost always the documents that were not related to the initial query were placed in this group. This shows that many of the search engines' internal representation and indexing are not quite as trouble-free or consistent as a user wants. Thus the IQBIR system groups them into a separate space to reduce user effort of weaving through non-relevant documents.

5. CONCLUSION

The system in this work aids in the search and learning the user concept of interest from multiple existing information retrieval systems, in this case the search engines for the Internet. Clustering and active learning is applied to reduce the number of user feedback and training and fine-tuning the clustering classifier to match the user's concept to the unknown internal representation of multiple search engines.

The prototype is developed using the Java language for the main prototype, Javascript, html, and cgi-bin protocol for the client interface, and C for clustering and learning by modifying the LVQ/SOM package ([2]).

Some of the novel methods that are addressed in this work include intelligent learning agent between existing search engines and the user, reducing manual user feedback and query cycles by active clustering, and clustering multiple concepts with representatives during retrieval.

Preliminary tests from the prototype show promising results in the efficiency and increasing accuracy for searching and browsing user concepts.

6. REFERENCES

[1] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[2] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "Lvq_pak: A program package for the correct application of learning vector quantization algorithms," in *Proc. Intl. Joint Conf. Neural Networks*, pp. 725–730, 1992.

[3] B. Krulwich, "Learning user interests across heterogeneous document databases," in *Proc. 1995 AAAI Spring Symp. Info. Gathering from Heterogeneous, Distributed Environments*, 1995.

[4] H. S. Nwana, "Software agents: An overview," *Knowledge Engineering Review*, vol. 11, pp. 1–40, Sep. 1996.

[5] J.-M. Park and Y.-H. Hu, "On-line learning for active pattern recognition," *IEEE Signal Processing Letters*, Nov. 1996.

[6] M. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & webert: Identifying interesting web sites," in *Proc. 1996 AAAI Spring Symp. on Machine Learning in Information Access*, 1996.

[7] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

[8] A. F. Smeaton and F. Crimmins, "Using a data fusion agent for searching the www," in *Sixth Int. World Wide Web Conf.*, 1997.