Autonomous agent for gathering information to build focused index from distributed environment

Rushdi A. Hamamreh

Department of Computer Engineering, Al-Quds University

Abstract:
In this paper describes architecture of autonomous agent that gathering information from distributed environment as Internet to build sub-specific collection, and to extract information from documents used latent semantic indexing algorithm and two filters one for collection and the other for queries.

Keywords:
Internet autonomous agent, distributed systems, Collections, latent semantic index, thematic filter.

1. Introduction

It’s well known that search engines with centralized architecture can’t index the whole Internet because the exponential growth of the number of documents published in the Internet. Search engine with distributed architecture is scalable solution of this problem. In the framework of this architecture we use a set of subject specific collections of electronic documents published in the Internet. These collections belong to different owners who are responsible for their content, indexing and quality of search. User’s query is automatically propagated to one or more collections with topics relevant to the query topic[1,2,4,10].

Conventional method of generation of subject-specific collection is preparation of the collection core which consists of a relatively small set of documents relevant to the collection topic. Administrator of the collection is responsible for preparation of this core. After that we can use information agent whose goal is to scan Internet and seek documents relevant to the collection core. Usually collection filter is used to filter documents relevant to the collection topic. This filter is based on the analysis of the collection core.

In this paper we propose to use additional filter based on the analysis of archive of user’s queries previously received by this collection. This queries reflect information needs of the whole community of users and information agent should take into account these information needs. So a new document is recommended to the collection by our information agent if it’s accepted by filter based on analysis of the collection core or by filter based on analysis of the user’s queries archive.

2. Architecture of the autonomous agent

Our agent contains the following main components:

- **Analyzer of the collection content**

  The goal of this component is to analyze the whole set of documents from this collection and create the collection description which reflects the main subjects presented in this collection. We’ve used for this propose probabilistic latent semantic indexing [3,5].

  The goal of the latent semantic indexing is extraction of latent factors which reflect a set of narrow topics presented in the given collection.

  Let $Z = \{z_1, \ldots, z_k\}$ be set of these factors. Let denote

  - $P(z_i)$ – probability that randomly selected document from the collection best of all corresponds to the topic $z_i$ (see Eq. 2).
  - $P(d \mid z)$ – probability that for the given factor $z_i$ this factor best of all corresponds to the document $d_j$ (see Eq. 3).
  - $P(w \mid z)$ – probability that for the given factor $z_i$ this factor best of all corresponds to the word $w_j$ (see Eq. 4).

  Here $D = \{d_1, \ldots, d_n\}$ is set of all documents from the collection and $W = \{w_1, \ldots, w_m\}$ is set of all words from this collection.

  Functions $P(z)$, $P(d \mid z)$ and $P(w \mid z)$ can be estimated in the process of a likelihood function maximization. This function is presented in the following form.
\[ L = \sum_{d} \sum_{w} tf(d, w) \log(P(d, w)). \]  

(1)

Standard Expectation Maximization algorithm is used for maximization of this function. Two steps are executed at every iteration of this algorithm. The first one is Estimation 

\[ P(z | d, w) = \frac{P(z)P(d | z)P(w | z)}{\sum_{z'} P(z')P(d | z')P(w | z')}. \]  

(2)

The second one is Maximization 

\[ P(w | z) = \frac{\sum_{d} tf(d, w)P(z | d, w)}{\sum_{d, w'} tf(d, w')P(z | d, w')}, \]  

\[ P(d | z) = \frac{\sum_{d, w} tf(d', w)P(z | d', w)}{\sum_{d, w} tf(d', w)P(z | d', w)}, \]  

\[ P(z) = \frac{\sum_{d, w} tf(d, w)P(z | d, w)}{\sum_{d, w} tf(d, w)}. \]  

(3) (4) (5)

To generate the collection filter we’ve selected the most heavy words from W. Weight of the word w is calculated as 

\[ \text{weight}(w) = \sum_{z \in Z} P(z)P(w | z). \]  

(6)

Every vertex should have a weight which reflects the role of this word in the collection subject. Some of these words are presented in the collection core and we can use probabilistic latent semantic indexing to calculate their weights. But a part of words presented in the users queries can be new (not presented in the collection core). To estimate their weights we’ve used the following method.

We suppose that weight of every new word should be equal to the average value of weights of words which are neighbors of this word. We’ve used iteration algorithm to estimate weights of all new words according to this proposal. All information about queries words and their weights is stored as query statistics.

**Generator of thematic filters**

Agent uses two different filters. The first one reflects the content of the collection content. We use collection description to select the most major subjects presented in the collection core and generate a list of weighted words which present selected topics. Collection filter is a part of this list which contains specified by the collection administrator number of most heavy words.

The second filter reflects information needs of the users community. It’s presented as weighted list of words from users queries. These weighted are calculated as described previously.

**Wget**

Agent uses wget unix utility for document downloading.

**Filter**

This component filters newly downloaded document. It uses Collection Filter (CF) and filter based on the analysis of users queries, is called Queries Filter (QF). The document is recommended for inclusion to the collection if it’s accepted any of these two filters. The document is accepted by a filter if dot product of tf-profile of the document and filter exceeds the given threshold.

**URL parser**

This component parses text of a document to select URL of html document presented as links in the given document. This URL are used to seek new document relevant to the collection topic.
URLs queue

URL queue starts from a set of start URLs presented by the collection administrator. Every URL from this queue is assigned estimation of usefulness of this URL for seeking of new relevant documents. At the first step the newly included to this queue URL is assigned 1 as its usefulness. At every next step agent chose from queue URL with maximum value of estimation of its usefulness, downloads it and filters it. If this document is accepted by filter then at next steps agent randomly chose links presented in its text and includes them into URL queue with usefulness estimation equal 1. If a downloaded document isn’t accepted by filter, then estimation of usefulness of a URL of a document, where link to this document occurs, is decreased. As a result, estimation of the URL usefulness is approximation of probability of relevance of a link from the document to the collection topic.

![Architecture of the agent](image)

3. Result

To test our agent we’ve used it to generate a collection on the information retrieval topic. The collection core contains about 100 documents which were selected by an expert. These documents were used to generate the collection filter. We’ve used a set of queries (about 500) as query archive. These queries were supplied by an expert also and were used for query filter generation. Agent starts from a set of start URL. This set contains about 50 links on relevant html documents and was presented by the expert.

The goal of the test was to find best values for thresholds for collection filter and query filter. After 5000 documents which included collection through two filters CF and QF with threshold of
collection filter $T_{CF} = 0.1$ and threshold query filter $T_{QF} = 0.5$.

Result of the test shows that in the best case the precision of the agent is 0.81. This means that our agent downloaded a set of new documents, about 80% of which were recommended by this agent to the collection, and our expert estimate that almost all recommended documents are relevant to the collection topic, (see figure 2).

4. Application

For our model implemented simple interface (see figure 3), which determine the five options: directory of core collections, path of file queries, directory of new files and two numbers the $T_{CF}$ and $T_{QF}$.

---

**Autonomous Focused Agent**

- **Directory of core collections**: C:\agent\core
- **Path of file queries**: C:\agent\queries
- **Directory of new files**: C:\agent\new

**Parameters**

- **$T_{CF}$ Threshold of Collection Filter**: 0.1
- **$T_{QF}$ Threshold of Query Filter**: 0.5

---

**References**