A Consensus-Based Multi-agent Approach for Information Retrieval in Internet

Ngoc Thanh Nguyen[1], Maria Ganzha[2], and Marcin Paprzycki[3]

[1] Institute of Information Science and Engineering, Wroclaw University of Technology, Poland
thanh@pwr.wroc.pl
[2] Department of Management, Elbląg University of Humanities and Economy, Poland
ganzha@euh-e.edu.pl
[3] Computer Science Institute, Warsaw School of Social Psychology, Poland
marcin.paprzycki@swps.edu.pl

Abstract. This paper presents a consensus-based approach utilized within a multi-agent system which assists users in retrieving information from the Internet. In this system consensus methods are applied for reconciling inconsistencies among independent answers generated by agents (using different search engines) for a given query. Proposed agent system has been implemented and initial experimental results are presented.

1 Introduction

Information retrieval is one of tasks, which are the most often realized by computer users and a large number of methods, technologies and search engines have been proposed for aiding them in this task. However, most users utilize only a single search engine. They trust it that the obtained answer is relevant and complete. But an interesting question arises: with so many existing search engines – why to use only one? It is quite possible that different search engines provide "best" responses to different queries.

Consensus methods, based on the consensus theory that states that if the same task is entrusted to several experts (and their credibility is approximately the same) then their reconciled solution (the *consensus answer* or the *consensus* – we use these terms interchangeably in what follows) should be more credible than those generated by them individually, have been proved to be useful in dealing with data originating from multiple sources [1], [4], [6]. Similarly, consensus based on responses generated by multiple search engines should be more relevant than each of individual responses. This fact motivates usage of multiple search engines in retrieving information form the Internet.

The initial predicted role of agent systems was in the area of information management [12]. In this context agent autonomy and communication have been particularly useful [5]. These features make them also natural candidates for realization of "consensus computing systems," created with the goal of improving quality of information retrieval. At the same time, there are very few such systems in existence. Menczer [9] has designed and implemented *Myspiders*, a multi-agent system for information

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 208 – 215, 2006.

discovery in the Internet and has shown that augmenting search engines with adaptive intelligent search agents can lead to significant competitive advantages. Another approach was realized in the www.metacrawler.com, which utilizes multiple search engines.

In this paper we present a novel agent system designed to assist information retrieval from the Internet. Our system differs from the one of Menczer in using consensus methods [10, 11] for resolving differences in response sets and using multiple agents for the retrieval task. The advantage of this approach is that agents use different search engines (among themselves and for subsequent queries). To create the final answer we use the consensus method to select URLs' that are ranked as minimally distant from those provided by other agents. The *consensus answer* is displayed to the user and, due to the way it was obtained, it is expected to be more complete and relevant. Furthermore, each agent has its knowledge base containing information about results of past searches and used to select the search engine to be utilized in subsequent searches. Let us now describe in more details the proposed approach, followed by the set of initial experimental results.

2 Multi-agent System Design

2.1 Motivation

The aim of the project is to create a consensus-based multi-agent system to aid users in information retrieval from the Internet. This approach enables to solve the following two problems often occurring in the information retrieval processes:

- *Low relevance of answers generated by a search engine* – caused by non-effective work of filters. As a consequence, many non-related pages (e.g. advertisements, but also pages loosely associated with the subject of the query) may be displayed. For instance, for the query „Wroclaw University of Technology" *Yahoo* as the most relevant gives http://www.pwr.wroc.pl/~promocja/eng/main2.html which can be considered a "correct" answer. However, *Google* classifies this URL on the 2nd position, while *Onet* does not include it within first 60 displayed URLs.
- *Displaying repeating URLs* – which are identical or very similar to each other. This is a burden for the user because he loses a lot of time to scroll many screens to find information of interest.

The proposed system exploits answers generated by multiple search engines. Utilizing the consensus algorithm, their optimal combination is determined and displayed to the user. Answer-sets (and thus search engines) are evaluated on the basis of their differences from the *consensus answer*. This information is fed-back to search agents, stored and utilized in subsequent queries to select the search engine to be used.

2.2 System Design

The general structure of the system is represented in Figure 1 as a UseCase diagram. Within the system we can see two types of agents: (1) Search Agent (SA) and (2) Manager Agent (MA). We assume that for each user there will be a single MA, while multiple SAs will be created to utilize multiple search engines. For the time

being we assume also that the number of agents is smaller than the total number of available search engines. As can be seen the SA obtains a query form the MA and after selecting the search engine (which is based on the information stored in the knowledge base (KDB) and coordinated by the MA) it queries it and prepares the report for the MA (this function involves removal of multiple URLs and selecting *n* best answers). On the basis of evaluation of its work, performed by the MA, the SA updates its KDB. The MA, on the other hand, receives a query from the user and sends it to the SAs. It oversees the process of search engine selection (each SA is to utilize a different one). Upon reception of the search results from all agents it determines the *consensus answer* which will be presented to the user [11]. However, before the answer is presented its *consistency* is established. If consistency is high (results were similar), then the results are used immediately to rank SAs (results close to the consensus are ranked higher than these that are not). If consistency is low (search engines did not agree on the answer set) then user is asked to pick relevant answers. These picks are then used to rank answers and provide feedback to SAs. If the user does not pick any answers (for whatever reason, as she is not forced to do so), no feedbacks is send.

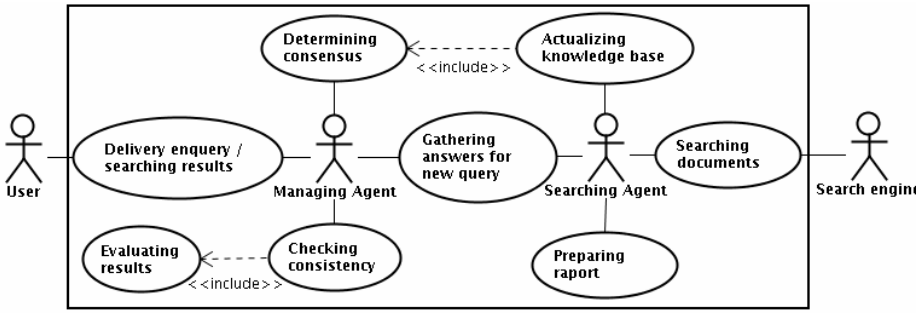

Fig. 1. System Use Case diagram

Thus far we have omitted a number of administrative functions that are involved in, among others, system start-up and system shut-down. These functions are: (1) Upon system start-up all necessary agents are created (first the MA, that creates a specific number of SAs) and the SAs load information about search engines (e.g. location, interface, etc.) from a shared database. (2) Each SA loads, from the KDB, individual information that is used in search engine selection. (3) Upon system shut-down each SA stores content of its KDB for use during the next system run.

Let us now assume that the system is running (all agents have been initialized and information about at least some searches is stored in each SA's KDB). In Figure 2 we present a complete UML Action Diagram of interactions between the MA and an SA during servicing a user query. Let us note that the selection of the search engine involves both the information stored in the KDB and interactions with the MA. To prevent all SAs from converging on a single search engine we force each SA to use a different one to process each query. This process is completely asynchronous; e.g. the

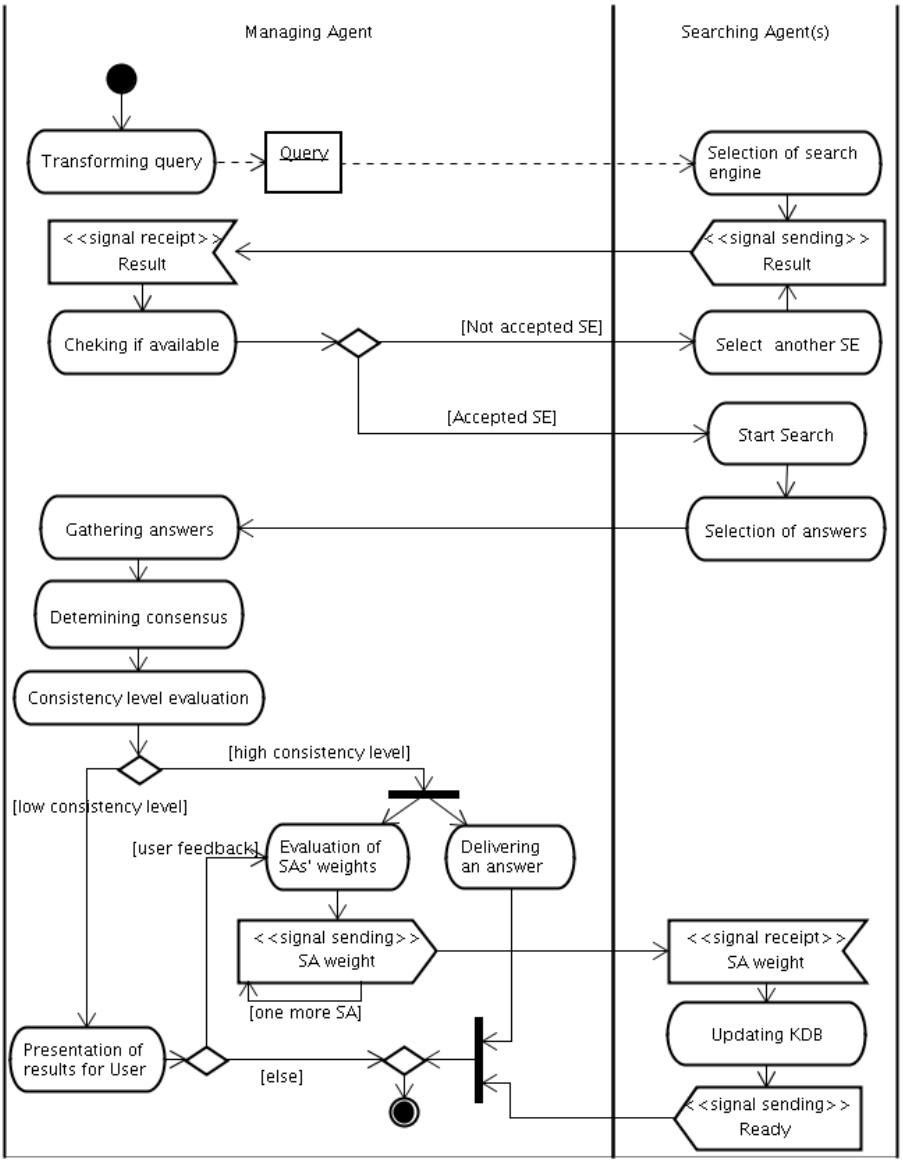

Fig. 2. Servicing user query – UML activity diagram

first SA to select *Google* and contact the MA will be able to use it, while the remaining SAs interested in *Googe* will have to select a different search engine. The MA oversees this process until all SAs have selected a different search engine. To simplify this process we have decided to utilize smaller number of agents than the number of available search engines and allow them to select randomly the next search engine (in the case when they have to use one that is not their favorite).

2.3 Main Algorithms Used in the System

Let us now describe key computational processes that take place in the system. First the MA finds the *consensus answer* using a version of the *branch-and-bound* algorithm [2] (we assume that there are *m* SAs and they report *n* results each):

Input: *results* provided by *m* SAs – $r^{(1)}$, $r^{(2)}$, ..., $r^{(m)}$ each in the form $r^{(i)} = <d^{(i)}_1, d^{(i)}_2, ..., d^{(i)}_n>$ where $d^{(i)}_1, d^{(i)}_2, ..., d^{(i)}_n$ are URLs representing documents for i = 1, 2, ..., *n*.
Output: *consensus answer* $C = <d_1, d_2, ..., d_n>$

BEGIN

1. create set *D* of all documents from all *results* (without repetitions)
2. for each $d \in D$
   - create vector $<t_1, t_2, ..., t_n>$ where t_j is the position on which *d* appears in $r^{(j)}$; if *d* does not appear in $r^{(j)}$ then set t_i as the length of the longest ranking increased by 1
   - calculate average t(d) of values $t_1, t_2, ..., t_n$
3. *consensus answer C* is obtained by ordering elements of *D* according to values t(d)
4. since size of *D* can be large one can utilize only a few top results.

END.

The resulting *consensus answer* represents a combined view on what the search engines believe the final answer should be. Unfortunately such a final answer is not always reliable. We can say that it is reliable only when all search engines have a very similar view as to what the final answer should be. To be able to establish how similar obtained answers are we calculate the *consistency of the consensus.* This is achieved by applying the following algorithm (see also [11]):

Input: Set *X* consisting of *n* binary matrices $A^{(k)}$ (for $k=1,...,n$) of size $m \times m$ representing individual rankings (result sets)
Output: YES if consensus is consistent; NO otherwise

BEGIN

1. For all rankings calculate consensus *C*
2. Calculate $\hat{d}(X) = \dfrac{\sum_{x,y \in X} d(x,y)}{m(m+1)}$;
3. Calculate $\hat{d}_{\min}(X) = \dfrac{\sum_{y \in X} d(C,y)}{m}$;
4. If $\hat{d}_x(X) \geq \hat{d}_{\min}(X)$ then YES; NO otherwise

END.

In the case when the *consistency of the consensus* is low (below a threshold; and the above algorithm responds with a NO) results are shown to the user and she is requested to provide explicit feedback by selecting responses that she finds to be

relevant to the query. These responses are then used to rant the search engines. This is done in the similar way to what happens when consistency is high enough. Then the following operations take place (C is the *consensus answer* calculated as above):

+ distance $d(C,A^{(j)})$ for $j = 1, \ldots, m$ is calculated
+ such k that $d(C,A^{(k)})$ is minimal is found
+ weight for agent k is then assumed to be $W[k] = 100\%$;
+ for each remaining agents SA its weight is equal to:

$$W[j]:= \frac{d\bar{l}(p_i \cap q_j)}{\max\{dl(p_i), dl(q_j)\}} 100\%$$

where $d\bar{l}(p_i \cap q_j)$ is the length of common part of strings p_i and q_j starting from their beginning, and $j = 1,\ldots, m, j \neq k$.

Weights $W[j]$ are then sent back to each SA and will be used to update its KDB and thus utilized when the search engine for the next query is to be selected.

The following other algorithms are utilized in the system (their detailed descriptions are given in report [3]):

- Algorithm for eliminating repeating URLs,
- Algorithm for transferring a binary matrix into a ranking,
- Algorithm for calculating distances between rankings,
- Three algorithms for calculating three types of distances between queries.

Let us now describe in some more detail finding distances between queries. User can express queries using logical forms with conjunctions "$\wedge$" and "$\neg$", in the form $(a \wedge b) \wedge (\neg c)$, where a, b and c are terms. For example,

$$(consensus \wedge conflict) \wedge (\neg\, voting).$$

This query is extracted form the interface by the MA and passed to the SA which reformulates it for the selected search engine (e.g. interia.pl) as an HTTP statement:

http://szukaj.interia.pl/id/query?mss=search&q=*consensus*+*conflict*~+*voting*&eng=dc.

Before this step takes place, the SA has to select which search engine it would like to use. In its KDB it stores past queries and success achieved in responding to them using various search engines. New queries are compared to the old ones, among others, using the following algorithm (see also [10, 11]):

Input: queries p and q in the form $p=(p_1 \vee p_2) \wedge (\neg p_3)$ and $q=(q_1 \vee q_2) \wedge (\neg q_3)$,
Output: distance $d(p,q)$.
BEGIN

1. Calculate $d(p_i,q_j)= \frac{d\bar{l}(p_i \cap q_j)}{\max\{dl(p_i), dl(q_j)\}}$ for $i,j=1,2$, where $dl(x)$ is the length of string x;
2. Calculate $d=\max\{d(p_i,q_j)$ for $i,j=1,2\}$;
3. Calculate $d'=d(p_3,q_3)$ using the pattern in step 1;
4. Calculate $d(p,q)=(1/2)(d+d')$

END

3 Initial Experimental Results

The proposed system has been implemented using Aglets agent environment [7, 8]. Specifically, we used JDK 1.1.6, Swing 1.0.1, JBCL 2.0 and IBM Aglets 1.1b3. For response sets generated by each search engine we have removed URL's that were more than 80% similar, assuming that they point to the same answer, and left only one. Then we have used 30 best responses. In experiments described below 20 search engines and 6 SAs were used. Finally, an SA was allowed to try to assign a search engine to the query only if its accuracy in the KDB was above 50%. Otherwise it was forced to select a random search engine.

We performed two tests. At the beginning of each test KDBs of all SAs were empty. The first test involved query „głosowanie ∧ wybory" („voting ∧ election", in Polish). Table 1 contains the information about the search engines used by SAs and resulting weights informing about the similarity degree between the URLs' rankings provided by SA agents and the *consensus answer*. In all samples the consistency level of response sets was high enough to not to involve the user.

Table 1. Results of 6 samples for query: „głosowanie ∧ wybory"

Sample	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6
Sample I.	Onet 84%	Ahoj 100%	ICM 65%	Hoga 51%	Interia 56%	Google 43%
Sample II	Onet 51%	Ahoj 100%	ICM 21%	Hoga 25%	Interia 27%	WP 36%
Sample III	Onet 100%	Ahoj 52%	Arena 21%	WP 28%	Google 49%	Poland 45%
Sample IV	Onet 100%	Ahoj 52%	Hoga 21%	ICM 18%	Poland 19%	Google 33%
Sample V	Onet 100%	Ahoj 52%	ICM 21%	Hoga 18%	Interia 19%	WP 44%
Sample VI	Onet 100%	Ahoj 72%	Poland 56%	ICM 25%	Arena 32%	Hoga 14%

From Table 1 follows that the best search engines turned out to be Onet and Ahoj (both search engines focused on Polish web sites). We can also notice that in the first retrieval majority of weights for search engines are high enough (more than 50%). This means that rankings generated by SAs are consistent to the large degree. Therefore, in the second sample agents 1-5 used the same search engines while agent 6 had to draw a new one. This was the reason of change of weights to the disadvantage of agents 3, 4 and 5. In the next samples only agents 1 and 2 have good enough accuracy and can use the same search engines (Onet and Ahoj) while other agents had to draw. It is worth noting that because the query was in Polish, therefore agents draw only from Polish search engines. After checking by an expert it turned out that all responses were relevant.

In the second test the system performed retrieval for 50 different queries related to "holidays." On the basis of results obtained in these tests we can draw the following conclusions:

• Utilization of KDBs caused SAs to less and less often draw search engines, rather these selections have been determined on the basis of previous queries.

• Growth of information stored in KDBs resulted in results being more and more relevant (e.g. number of „pushy" URLs has been smaller and smaller). In the final rankings for the last query „pushy" URLs have not occurred.

4 Conclusions

While utilization of several search engines for the same query is not novel, method for reconciling the results presented here is. Its advantage is that it does not need the information about the user (her preferences, profiles etc.). It works on the basis of an assumption that if several experts (search engines) solve the same problem, then the reconciled solution should be more credible than those proposed by individual experts. Future work should concern using advanced inconsistency measures to evaluating conflict situations and making decision for consensus determination [11]. We also plan to utilize the proposed approach for content collection within the agent-based travel support system described in [13].

References

1. Barthelemy, J.P., Janowitz M.F.: A Formal Theory of Consensus. SIAM J. Discrete Math. **4** (1991) 305-322
2. Barthelemy, J.P., Guenoche, A., Hudry, O.: Median linear orders: Heuristics and a branch and bound algorithm. European Journal of Operational Research **42** (1989) 313-325
3. Błażowski, A, Nguyen, N.T.: AGWI- Multi-agent System Aiding Information Retrieval in Internet. In: Proceedings of SOFSEM 2005. Lecture Notes in Computer Science **3381** (2005) 399-403.
4. Day, W.H.E.: Consensus Methods as Tools for Data Analysis. In: Bock, H.H. (ed.): Classification and Related Methods for Data Analysis. North-Holland (1988) 312-324
5. Ferber, J.: Multi-Agent Systems. Addison Wesley, New York (1999)
6. Katarzyniak R.P., Pieczynska-Kuchtiak A.: A consensus based algorithm for grounding belief formulas in internally stored perceptions. Neural Network World **5** (2002) 461-472
7. Lange, D., Oshima, M.: Programming and Developing Java™ Mobile Agents with Aglets, Longman (1998)
8. Lange, D., Oshima, M.: Java agent API: Programming and deploying aglets with Java. Aglets Web page: http://www.ibm.co.jp/trl/projects/aglets/
9. Menczer, F.: Complementing Search Engines with Online Web Mining Agents. Decision Support Systems **35** (2003) 195-212
10. Nguyen, N.T.: Consensus System for Solving Conflicts in Distributed Systems. Journal of Information Sciences **147** (2002) 91-122
11. Nguyen, N.T., Malowiecki, M.: Consistency Measures for Conflict Profiles. LNCS Transactions on Rough Sets **1** (2004) 169-186.
12. Maes P.: Agents that Reduce Work and Information Overload. Communications of the ACM, 37, 7 (1994) 31-40
13. Ganzha M., Gawinecki M., Paprzycki M., Gąsiorowski R., Hyska W., Pisarek S.: Utilizing Semantic Web and Software Agents in a Travel Support System, Idea Publishing (to appear)