
Taroub Issa
Al-Quds Open University
Palestine
tessa@qou.edu

ABSTRACT

The Web is a major source of accurate information for users. It is a dynamic environment that is changing all the time. It consists of three dynamic components: Content, Link and Usage. These components are the goal of web search engine researchers. For search engines to do their job properly, they must pass through many steps: crawling, indexing, retrieving and ranking. Unfortunately, recent search engines suffer from a serious problem, that is: they can't keep up with this dynamic environment of the Web. Their indexes are not updated frequently. This paper presents the way in which search engines work and describes in detail the basic tasks of a search engine. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been conducted on them.

An overview of how the whole system of a search engine works is provided, and makes an overview of some models that been used to improve the way in which search engines work like: Waco, InfoSpider, ContentUsageAnts and Webcomb model. We will introduce the way in which each model works. Filippo proposed an implementation of the Infospider model called MySpider, which is an evolutionary multi-agent system in which agent could adapt with their environment. This model was not an alternative for recent search engines, on the contrary, it is used to complement the recent search engines by improving their performance. The three other models also introduced to improve the search engines work.

Keywords: Complex Adaptive System, Multi Agent System, Web Content, Usage Content, Self Organize, Search Engine Crawler.

I. INTRODUCTION

The Web is a major source of accurate information for users. It is a dynamic environment that changes all the time. Finding relevant and recent information is therefore a hard task. This dynamic huge amount of data needs organizing, filtering and maintaining all the time. For search engines to do their job properly, they must pass through many steps: crawling, indexing, retrieving and ranking. Unfortunately, recent search engines suffer from a serious problem, that is: they can't keep up with this dynamic environment of the Web. Their indexes are not updated frequently.

Search engines use Content and Link cues when searching for relevant pages that meet with the user query. They depend on clustering hypothesis and clustering link hypothesis as a strategy for search. There is a serious need for designing models that could reorganize the content of the web dynamically to meet users' needs. This work represents models that tried to reorganize the web content and links to make the work of search engines easier like InfoSpider, ContentUsageAnts, Waco and WebComp model.

The paper will be as follows: Section 2 describes how search engines work. Section 3 gives a detailed definition of the four models, in which we describe the dynamic three web components: Content, Link and Usage that the models deals with. The content of the web is usually a combination of textual and non textual components of the website. Web content is the stuff in your web that includes but not limited to database, utility applications and multimedia [1]. As the HTML pages are connected to each other by links, [2] sees the WWW as a complex network whose vertices are HTML pages and links connecting pages with each other. Web pages are diverse in their structure, like hubs and authorities pages [3], and web pages were divided into five categories: Strongly Connected Components SCC, IN, OUT, tendrils and tubes, and disconnected. In [4] users were classified into random users, rational users and recurrent users. Web page authors come from different backgrounds, creating a vast variety of topics. Also is section 4 we focus on the ContentUsageAnts model illustrating how we can achieve a self organized model by MAS and complex adaptive system. Also we illustrate the results obtained for this model. Finally conclusion and the references.
II. HOW DOES SEARCH ENGINE WORK?

Engineering a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been conducted on them.

Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is a very hard task. There are differences in the ways various search engines work, but they all perform three basic tasks:
1. They search the Internet or select pieces of the Internet based on important words.
2. They keep an index of the words they find, and where they find them.
3. They allow users to look for words or combinations of words found in that index.

A search engine finds information for its database by accepting listings sent in by authors who want exposure, or by getting the information from their "web crawlers," "spiders," or "robots," programs that roam the Internet storing links to and information about each page they visit. A web crawler is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, S0, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Collected pages are later used for other applications, such as a Web search engine or a Web cache.

The most important measure for a search engine is the search performance, quality of the results and ability to crawl, and index the web efficiently. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Some of the efficient and recommended search engines are Google, Yahoo and Teoma, which share some common features and are standardized to some extent.

Commercial developers noticed the potential of the web as a communications and marketing tool when graphical Web browsers broke onto the Internet scene (Mosaic, the precursor to Netscape Navigator, was the first popular web browser) making the Internet, and specifically the Web, "user friendly." As more sites were developed, the more popular the browser became as an interface for the Web, which spurred more Web use, more Web development etc. Now graphical web browsers are powerful, easy and fun to use and incorporate many "extra" features such as news and mail readers. The nature of the Web itself invites user interaction; web sites are composed of hypertext documents, which mean they are linked to one another. The user can choose his/her own path by selecting predefined "links". Since hypertext documents are not organized in an arrangement which requires the user to access the pages sequentially, users really like the ability to choose what they will see next and the chance to interact with the site contents [8].

A. SEARCH ENGINE SYSTEM ARCHITECTURE: Before a search engine can tell you where a file or document is, it must be found. To find information on the hundreds of millions of Web pages that exist, a typical search engine employs special software robots, called spiders, to build lists of the words found on Web sites. When a spider is building its lists, the process is called Web crawling. A Web crawler is a program, which automatically traverses the web by downloading documents and following links from page to page. They are mainly used by web search engines to gather data for indexing. Other possible applications include page validation, structural analysis and visualization, update notification, mirroring and personal web assistants/agents etc. Web crawlers are also known as spiders, robots, worms etc.

Crawlers are automated programs that follow the links found on the web pages. There is a URL Server that sends lists of URLs to be fetched to the crawlers. The web pages that are fetched are then sent to the store server. The store server then compresses and stores the web pages into a repository. Every web page has an associated ID number called a doc ID, which is assigned whenever a new URL is parsed out of a web page. The indexer and the sorter perform the indexing function. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every web page and stores important information about them in an anchors file. This file contains enough information to determine where each link points from and to, and the text of the link [6].

The URL Resolver reads the anchors file and converts relative URLs into absolute URLs and in turn into doc IDs. It puts the anchor text into the forward index, associated with the doc ID that the anchor points to. It also generates a database of links, which are pairs of doc IDs. The links database is used to compute Page Ranks for all the documents.

The sorter takes the barrels, which are sorted by doc ID and resorts them by word ID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of word IDs and offsets into the inverted index. A program
called Dump Lexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. A lexicon lists all the terms occurring in the index along with some term-level statistics (e.g., total number of documents in which a term occurs) that are used by the ranking algorithms. The searcher is run by a web server and uses the lexicon built by Dump Lexicon together with the inverted index and the Page Ranks to answer queries[6].

B. HOW A WEB CRAWLER WORKS: Web crawlers are an essential component to search engines; running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers, which are all beyond the control of the system. Web crawling speed is governed not only by the speed of one’s own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel. Despite the numerous applications for Web crawlers, at the core they are all fundamentally the same. Following is the process by which Web crawlers work:
1. Download the Web page.
2. Parse through the downloaded page and retrieve all the links.
3. For each link retrieved, repeat the process.
   The Web crawler can be used for crawling through a whole site on the Inter-/Intranet. You specify a start-URL and the Crawler follows all links found in that HTML page. This usually leads to more links, which will be followed again, and so on. A site can be seen as a tree-structure, the root is the start-URL; all links in that root-HTML-page are direct sons of the root. Subsequent links are then sons of the previous sons.

A single URL Server serves lists of URLs to a number of crawlers. Web crawler starts by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and so on, recursively. Webcrawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. Each crawler keeps roughly 300 connections open at once. This is necessary to retrieve web pages at a fast enough pace. A crawler resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the crawler really does is to automate the process of following links.

Web crawling can be regarded as processing items in a queue. When the crawler visits a web page, it extracts links to other web pages. So the crawler uses these URLs at the end of a queue, and continues crawling to a URL that it removes from the front of the queue[7].

C. IMPROVEMENTS OF WEB SEARCH ENGINES:
There are different ways to improve the performance of web search engines. Generally speaking, there are three main directions:
1. Improving user interface on query input
2. Using Filtering towards the query results
3. Solving algorithms in web page spying and collecting, indexing, and output
Method 3 is the fundamental solution for any direct search engines to deal with the problems of unequal accessing, out-of-date information, and low metadata using, as well as information coverage. It is also very important to look at the user interface issues that methods 1 and 2 are dealing with — How to handle user queries effectively and present the results efficiently[5].

III. MODELS

A. INFOSPIDER:
F. Menczer, in his work 'Complementing search engines with online web mining agents, [2002]' *, noticed that the retrieved documents by the recent search engines suffer from low recall, low recency and low precision. The search engine indexes don't update frequently, so the user don't get the correct result for searching. Filippo proposed an implementation of the Infospider model called MySpider, which is an evolutionary multi-agent system in which agent could adapt with their environment. This model was not an alternative for recent search engines, on the contrary, it is used to complement the recent search engines by improving their performance. Infospider representation consists of keywords and feed-forward neural net. This system needs a seed-URLs pointing to pages relevant to the user query. Agents start crawling given an amount of energy. The agents used the
neural net to decide which links to visit next, rewarded by updating energy for each relevant search. InfoSpider can adapt by both neural net and keywords representation and by learning neural net weights. Agents have reproduction and destroying mechanisms. Selection mechanism manages the agents to visit the most promising sites that have the appropriate information and to execute concurrently. This leads to the multi-threaded version of InfoSpider, MySpider. It has a simple interface in which the user specifies a query and the maximum number of pages to be crawled by MySpider. This model needs some design decisions about security, using cache memory (only one agent can use the cache memory), the estimate function which decides the energy the agent receives the first time it visits a page, and the cost function which decides the amount of energy the agents spends the first time visits a page. Three metrics had been used to evaluate the model; recency, precision and recall. MySpider model was evaluated using precision-recall-recency plots, and showed a significant improve on using traditional search engines only [15].

B. WACO:

Hassas has proposed a framework for developing CAS for complex networks such as the internet and the web [10]. She combined Holland's basic properties to the stigmergy mechanism. She used situated multi-agents paradigm and behavioral intelligence for identifying the building blocks (agents), their internal models and their roles (tags), also used the Web as the physical environment to facilitate the emergence of aggregates and the flows of information. For communication, she used a mechanism based on a spatial representation and mediated by the environment, such like the stigmergy mechanism. This favours the aggregation of control information and its spreading through the distributed environment, also Hassas maintained equilibrium between exploration and exploitation in the behavior to allow aggregation (reinforcement) of the building blocks and diversity (randomness).

In [11, 12], Rattrout & el. at. illustrated how the CAS principles are applied in the context of web content organization through WACO (Web ants content organization) model. WACO is an approach, inspired by social insects, to organize dynamically the web content [11]. In this approach, the web is considered as a complex environment, inhabited by artificial creatures called WebAnts. These creatures, implemented by mobile agents [16], are organized in a colony and mimic some behaviours of natural ants, namely the collective sorting behaviour and the food foraging behaviour. The content of the web is viewed by WebAnts as a potential source of food that needs to be organized and collected in an efficient way. Documents contained by the websites are considered as objects to be sorted following their semantic contents, so as to construct semantic clusters, where a cluster is a set of semantically close documents with respect to a predefined similarity measure. We consider two levels in this application: a higher level corresponding to the extraction of information from documents using any algorithm of text mining and a lower level which uses a synthetic pheromone coding of the extracted information to organize and search the web content.

The social organization of the agents in WACO is achieved through dividing the agents' population into four groups, and assigning different roles to each group. The building blocks are the populations of WebAnts agents, which mimic the collective sorting behaviour and the foraging behaviour observed in natural ants. Four types of WebAnts were created, each assigned a different task (tags associated with each agent): Explorers WebAnts, Collectors WebAnts, Searchers WebAnts, Requests Satisfying WebAnts [22].

These different types interact with each other through feedback. No one is in charge. Each semantic topic is identified by a kind of pheromone. WebAnts communicate through a stigmergic mechanism, using a multi-structured electronic pheromone.

Synthetic pheromone is coded by a structure with these different fields:

- Label (Wij): characterizes the kind of information coded by the pheromone, which is in our context the semantic value of a document (weighted keyword).
  \[
  W_{ij} = L_c, H_c, T_f, \text{IDF} \tag{1}
  \]

Tf is the frequency of the keyword in the current document, the Hc is a Header constant (Hc > 1 if the word appears in a title, = 1 otherwise), which increases the weight of the keyword if it appears in the title of the document, and idf is the inverse of document frequency. The linkage constant LC (Lc > 1 if the word appears in a link, = 1 otherwise)

- Intensity (tij): expresses the pertinence of information; the higher its value, the greater its attractive power. This value is computed at each site i, for each topic j, using the number of documents addressing the same topic, each time (t+1) a new document is added, as:
  \[
  t_{ij} (t + 1) = r_j t_{ij} (t) + \sum_{k=i[D_{ij}]} D_{ij} t_{ij} (t) \tag{2}
  \]
rj represents the persistence rate ((1-rj) the evaporation rate). Djik (t) the intensity of pheromone emitted by a document k, on the site i for a topic j at time t, and Dij is the set of documents addressing topic j on the site i.

- Evaporation rate: expresses the persistence rate of information in the environment. The lower its value, the longer is the influence of the spread information in the environment. This value is computed as the ratio of documents addressing the same topic with respect to all documents of site i.
\[ r_j = \frac{|D_j|}{|D_i|} \]  \tag{3} 

\( D_j \) is the set of documents addressing the topic \( j \) on the site \( i \), and \( D_i \) is the set of all documents on the site \( i \). The idea here to make the clustering of documents of a same topic more attractive than isolated documents. If a site contains heterogeneous semantic content, this information is considered as not sufficiently pertinent. So the associated pheromone will evaporate more quickly than that emitted by homogenous content.

- Diffusion rate: expresses the distance to which information is spread in the environment, the higher its value the greater the scope of the information in the environment. We express this distance using the linkage topology information. Indeed, when browsing the web looking for a topic, one explores pages following links addressing the topic of interest. We associate to each site \( i \), a distance \( d_{ij} \) for each topic \( j \) addressed by \( i \), which is computed as the longest path from the site \( i \) to the last site addressing the topic \( j \), following a depth first search.

\[ d_{ij} = \max_k (d_{ik}) \]  \tag{4} 

\( k \) is the number of links addressing topic \( j \), from a site \( i \). The idea here is to make sites that are a good entrance point for a search, have a wider diffusion scope than the other ones. Doing so, we could guide the search process to handle queries like “Give me all documents \( k \) links away this one. The WebAnts in WACO are created in a dynamic environment. We express this distance using the linkage topology information. Indeed, when browsing the web looking for a topic, one explores pages following links addressing the topic of interest. We associate to each site \( i \), a distance \( d_{ij} \) for each topic \( j \) addressed by \( i \), which is computed as the longest path from the site \( i \) to the last site addressing the topic \( j \), following a depth first search.

C. WebComp Model:

Rattrout and Hassas proposed two models, WACO and Webcomp models to achieve selforganize Web. In [13], Rattrout proposed the WebComp, which is an advanced model (model to combine web components). It tried to study the web as a complex system and as a multi scale space with three components (content, link and usage), and tried to extract the semantic from these components which helps in search process.

This model viewed the web from four components: content, link, usage and semantic, tries to aggregate the information from these spaces according to their tags, to discover the hidden pages and to make the results of search is better. Agent’s communication used to enable the interaction between the three spaces.

It is based on two levels: virtual level and real one (figure 3.1). The interaction between two pages is done through different agents having the same head tags and different values of similarities. In this model, the problem is presented from three concepts as in figure 3.17.

A MODEL TO COMBINE WEB COMPONENTS:

In [17] Rattrout et al. suggest a multi-scale space model called Webcomp. Pages connect to each other whereby hyperlink. Web Pages have Content, Usage, Structure, and values of semantic. Finding the effective approach is the main idea which makes us able to do aggregation between various spaces without loosing the semantic [18] value of pages. We can enable the interaction within a single space of different spaces by using the agent’s communication. Figure.3.2 represents the interactions between these components.

There are two levels of space of this model: the real main space, and the virtual sub space. The virtual spaces enrich the real ones by adding information that are related to their dynamic Structure. Different agents can form the interaction between two pages specially the agents that have the same Head Tags and different values of semantic. We can also combine previous agents that are based on identical positive tags by using the resulted similarities. We can also combine a Multi Agent System by using further agents. We can know about interactions between different space levels by following the virtual dimension and according to the degree of similarity.

THE MULTI-SCALE SPACE DEFINITION:

The Total Documentary Space (TDS) is group of pages tagging from the Web by searching engines according to one (or a set of) keyword(s) \( (kj) \). We can use algorithms of Page rank, Best First Search, InfoSpider [19-21] to make a variety of Web documents \( (W) \). These pages \( (pi) \) are downloaded locally with the goal of applying all the needed process.

The page (information) is defined as follows:

\[ TDS = \bigcup\{p_i \in W, \forall k_j \in p_i, \forall p_i \in W \& p_i \in TD\} \]

\[ pi=<\text{information} \ Content(\text{IC}), \ \text{information} \ Link(\text{II}), \ \text{information} \ Usage(\text{IU})> \]

Where \( f(p, k) \) is the frequency of Keywords \( K \) in the page. \( i(k) \) is the opposite of the logarithmic frequency with regard to the page.

\[ I_C = \{k \mid k \in p \& w_{p,k} = f(p,k) i(k)\} \]

Where \( I_l \) is the set of links that exist in TDS. Two verities of agents are known for the Usage of a page \( pi \), a user agent \( (AgUj) \) and an abstract agent. A user agent communicates with the agent of the page \( pi \) \( (AgPi) \) to register the last page visited by the user from \( pi \). So, a history is shared between different agents for the same page. The last information is the major type head tag connected directly to the following level of document
spaces. IC denotes the head tag for the agent representing the Content space. IS denotes the head tag for the agent representing the semantic space. It denotes the head tag for the agent representing the space of links. Finally the IU denotes the head tag for the agent representing the Usage space. An abstract agent is created for each space according to the type of its main tag.

**MAIN LEVEL SPACES:**

1. **Link Space (EL):** is a graph space resulting from link similarity where,
   \[ E_L = \{ \forall p_i, p_j \in TDS / p_i \sigma_L p_j \}. \]

2. **Content Space (EC):** is a graph space resulting from similarity computation based on Content similarity where,
   \[ E_C = \{ \forall p_i, p_j \in TDS / p_i \sigma_C p_j \}. \]

3. **Usage Space (EU):** is a graph space resulting from Usage similarity computation based on the Usage information similarity where
   \[ EU = \{ p_k \in (Ep_i \cup Ep_j) \& p \notin TDS \} \] (Epi) is the space of visited pages starting from pi whoever the user is (ui). The similarity is calculated between the two spaces of the two pages (pi, pj).

4. **Semantic space:** is applied on the three different spaces in order to have a semantic significance for spaces.
   \[ E_S = \{ \forall p_i, p_j \in TDS / p_i \sigma_S p_j \}. \]

**CONTENT SIMILARITY \( \sigma_C \):**

Every relation between two pages in the Web is based on their relation for n terms. The similarity measures \( \sigma \) can be defined from distance measures \( \delta \) using the relationship [20-21]:

\[ \sigma_C(p, q) = \frac{\delta}{1+\delta} \]

Where \( \delta \) and \( \sigma \) are representations of the pages in word vector space after removing stop words and stemming. This is actually the “cosine similarity” function, traditionally used in information retrieval IR.

**SEMANTIC SIMILARITY**

A semantic similarity between two documents is defined in [18] using the entropy of the documents’ respective topics:

\[ \sigma_s(d_1, d_2) = \frac{2 \log \Pr[t_0(d_1, d_2)]}{\log \Pr[t(d_1)] + \log \Pr[t(d_2)]} \]

Where \( t(d) \) is the topic node containing d in the ontology, \( t_0 \) is the lowest common ancestor topic for \( d_1 \) and \( d_2 \) in the tree, and \( \Pr[t] \) represents the prior probability that any document is classified under topic t.

**LINK SIMILARITY**

Link similarity is defined with

\[ \sigma_l(p, q) = \frac{\|U_p \cap U_q\|}{\|U_p \cup U_q\|} \]

Where, Up is the set containing the URLs of page p’s outlinks, in links, and of p itself. Out-links are obtained from the pages themselves, while a set of in-links to each page in the sample is obtained from the list of the table of the out links that point to the pages exists. This Jaccard coefficient measures in [18] the degree of clustering between the two pages, with a high value indicating that the two pages belong to a clique.

**D. ContentUsageAnts model:**

ContentUsageAnts model proposed in [13]. It’s approach is to reorganize the content of the web pages to achieve a self-organize system as formal step in WebComp model [14]. It consists of two parts: physical environment and social environment. The physical environment is the dynamic web, where the web is considered as a complex environment. It has three basic components: content, usage and structure as shown in figure 3.2.

It focuses on comparing between the content information exists in the content space and the content information exists in the user space using CAS (complex adaptive system) and MAS (Multi Agent System) concepts. The dynamic complex web self-structures through the processes of self-organization and stigmergy. The content of the web pages is viewed by Agents(ANTS) as a potential source of food that needs to be organized so as to be collected in an efficient way.

Here the similarity between the usage document space and the content document space is done to remove all the bad pages from the resulting search. The usage space is built from many visited pages by the user in term of the search and exploitation of the information available over the web.

The URLs for these pages is constructed of context and links. Links have two types: inlinks and outlinks. In this model, the work is done on the context of pages. The pages are given a weight key(Label), giving importance to the occurrence of words in the header and links of the context of pages. Then the agents start to estimate the content similarity for these pages in the usage space in order to classify them and rank them from the higher to lower interests for the user. So, in the content analyzing, we look at the content of any page as vector words pace with additional weights, and treat it as normal page content similarity.

The content similarity is calculated for all the pages in the user space as well as in the content space. The results are used to cluster the URLs in the usage space. This process must be repeated every time the user visits new pages. A new pages are added to the usage space every
time the user visits pages according to his experience and knowledge. He goes to them from the outlinks that exist in the relevant pages.

In the usage space the resources are the collection of documents that could be collected in time $t_i$ before the user visit another pages. Adding pages to the index is treated in real time by agents which give us the opportunity to update the information rapidly as possible. The results of clustering are used for making comparisons between the content similarity in the content space and the content similarity in the usage space. Doing similarity comparisons between the two spaces let us reorganize the web pages between the relevant pages left in the path of accurate pages. The agent in order to affect its environment. In most domains of reasonable complexity, an agent will not have complete control over its environment. It will have at best partial control, in that it can influence it.

The social environment is represented by the MAS, in which the MAS are divided into groups and roles are defined for each group, then agents are assigned to groups. Roles can be formed and assigned to agents by emergent self-organization in the system depending on the environment state and based on the correlation with the agents’ spatial organization.

Figure 3.4 gives an abstract, top-level view of an agent. In this diagram, we can see the action output generated by the agent in order to affect its environment. In most domains of reasonable complexity, an agent will not have complete control over its environment. It will have at best partial control, in that it can influence it.

In this model we have: a dynamic Web that needs to be reorganized through the CAS and MAS concepts.

To support the work of the model to achieve a self-organize web, the agents work together in the usage space as well as in the content space, to give feedback and filter the results of search and remove all the bad pages.

Definitions of spaces in ContentUsageAnts Model:
- Content space (environment) $<$agent (Ac), page (Pc), opération (Oc), tag (Tc)$>$:
  It represents the real data in web pages, the data web pages was designed to convey the users ( not limited to text). The content space is the environment that contains the textual pages, where the agents navigate and explore searching for similar information to their tag. The agent is defined as a set of group of agents that navigate searching for information according to their specialty. The page is the set of target pages that the agent attack to get information. The operation is the role that the agent will do in it’s life cycle. Tag is the kind of information that illustrate each agent.
- Usage space (environment) $<$agent (Au), page (Pu), opération (Ou), tag (Tu)$>$:
  The usage space represents the information extracted from the session’s history for each user invited or authorized. It contains all the traces of URLs that is registered as the user visited while he still opens the session. These pages are mined for content and link information and also registered as usage information.

The content and the link represents two types of information. Web pages here represent the target where agents attack for information. The resources are the words extracted from the documents and links belong to the usage URLs[18].

AGENT, GROUP, ROLE:
This is the architecture that defines groups and roles for agents as in [23]:
- **Agent** $<$operation (O), tag (t), life(l), number (n)$>$ $<$type$>$: that is we need to know for each agent it's operation, number of the group, life cycle, and it's type.
- **Group:** are defined as set of agent aggregation, in conjunction with the role definition, group may be any usual multi-agent system.
- **Role:** (operation (O)): is an abstract representation of the agent function, service or identification within a group. Agent can handle a role. Handling a role in a group must be requested by the candidate group and must be awarded.
- **Life** $<$time(t), energy (e)$>$ $<$page (P)$>$: is identified by milliseconds, and the energy for each agent depends on the force of the value of the information he carries from the resources.
- **Page** $<$word (w), frequency (f), id$>$: we identify the page as a set of words in the cosine vector space, where for each word exists frequency and from which page.

For each page in the usage space has outlinks and inlinks and the content itself.

**Tag** (t) $<$agent(A), value(v), page(p)$>$ $<$type$>$: identifies the agents by one kind of information. Where the tagging mechanism identifies the process of the aggregation between the agents who have the same identifiers.
- **Resources** (R): $<$page (p), time (t)$>$
  The collection of documents that exist in the content environment that can be collected in time $t$ before the document is reloaded again.
- **Similarity** (oc): $<$page (p), q$>$ $<$thresholds (λ)$>$: Is the clustering process in the content space according to the similarity values between pages with respect to the threshold given by the user himself.

SimilarityAgent calculates the contents similarity between pages $Pi, Pj$, where $P$ is the set of the pages of the Total document space as follows:

**AgentSimilarity** (Agent/Group/Role)

Input: Page $P= \{P\}$

$$\sigmac (p,q) =$$
For each page \( P_i \) we calculate the summation of all the similarities between this page and the rest of the pages in the space, divided by \( n-1 \) as follows:\(^{18}\):

\[
\sigma (c, P_i) = \frac{\sum c \cdot P_i}{n-1}
\] (6)

In this model (ContentUsageAnts), there are many artificial agents called Ants that interact with each other to organize the content of Web pages inside the content and usage spaces. They mimic the behavior of natural ants.

Using the stigmergy mechanism, the Agents mimic the ants natural behaviors by:

- Foraging / Information search, Collective Sorting / Clustering of documents

The different types of Ants interact with each other and communicate through the stigmergyic mechanism using a multi-structured electronic pheromone. Synthetic pheromone is coded by a structure with different fields: The documents in the content space are indentified by a pheromone (ContentKey)[23]:

\[
\text{ContentKey (C}k\text{j)}: \text{characterizes the kind of information coded by the pheromone, which is in our context the semantic value of a document in the content space (weighted keyword)}
\]

\[
\text{C}k\text{j} = L_c. H_c. T_f. IDF. Uc
\] (7)

\( T_f \) is the frequency of the keyword in the current document, the \( H_c \) is a Header constant (\( H_c >1 \) if the word appears in a title, =1 otherwise), which increases the weight of the keyword if it appears in the title of the document, and idf\(_k\) is the inverse of document frequency. The linkage constant \( L_c \) (\( L_c >1 \) if the word appears in a link, =1 otherwise). Uc calculate the weight of documents in the usage space. As we describe next.

The documents in the usage space are indentified by a pheromone (UsageKey):

\[
\text{UK}\_j : \text{characterizes the kind of information coded by the pheromone, which is in our context the semantic value of a document in the usage space (weighted keyword)}
\]

\[
\text{UK}\_j = L_h. H_h. T_f. IDF
\] (8)

\( T_f \) is the frequency of the keyword in the current user document, the \( H_h \) is a Header constant (\( H_h >1 \) if the word appears in a title, =1 otherwise), which increases the weight of the keyword if it appears in the title of the document, and idf\(_k\) is the inverse of document frequency. The linkage constant \( L_h \) (\( L_h >1 \) if the word appears in a link, =1 otherwise).

How the Agents work insight the spatial organization of the complex web (in which websites consists of documents and relevant documents gathered in clusters)?

- The agents jumps from one site to another in the content space and usage space with the label given at the start of search.
- Analyzes the content of documents inside the website in the content space, and analyzing the content of history pages inside the usage space.
- Giving each document in both spaces a semantic value using any content mining algorithm (synthetic pheromone), we add here to the weight of the documents (TFIDF), the frequency of the keywords in the header of the documents (\( H_c \) or \( H_h \)) and the frequency of the keywords in the links of the document (\( L_c \) or \( L_h \)).
- Calculating the content similarity for these pages.
- If the document considered interesting, adding it to the appropriate cluster in both spaces.
- Through the search process, the agents (ants) drop the synthetic pheromone, as a trace for the other agents to follow the source of relevant pages.

IV. THE RESULTS OF CONTENTUSAGEANTS MODEL

After representing how the ContentUsageAnts model works, we must illustrate how our approach can achieve a self-organize web using stigmergy and MAS concepts. This passes through three main steps. The first main step represents collecting process and creating TDS. The URLs are stored in MySgl database table. They are ready now for the mining and similarity agents to work on them.

In step 2, the two sub-spaces are created from the TDS. Agents start working on the contents and links of pages to create the content and link spaces according to its information type. The four types of agents now work in creating the content space. The agents calculate the weights of pages in the content space according to weight formula:

\[
\text{C}k\text{j} = L_c. H_c. T_f. IDF. Uc
\]

The content similarity is found by applying the similarity rules using the \( Ckij \) among all possible pair of pages.

The same measurements are repeated for the link space using the link similarity rules. Pearson correlation coefficient is applied over the results to find the correlation between the two majors for TDS basic. Fig 3.7 represents the distribution for the pages with respective similarity measures. From figure 3.8, it is obvious that there is no correlation between them[18].

At this stage the user did not use the system. In other words he did not open a session. The agents start filling the tables with the information extracted from the pages and start to formalize each space. Content space different tables start to be filled with terms, their frequencies in
link, title and content, and their page references. Agents use the pheromone which leads the other agents to follow the source of relevant pages.

At t=0, user starts using the system. As a result, usage space is created and usage agents starts working. Pages added to the usage space are compared whether they exist in the TDS or not. Pages which are not in the TDS are added to it and recalculation and re-ranking is followed by it as shown in figure 3.9. Ants calculate the weight of pages visited by users according to the formula:

$$UK_{ij} = Lh \cdot Hh \cdot T_f \cdot IDF$$

In usage space, content and the link similarity is calculated in the same fashion as it is calculated in TDS. Table 7.4 illustrates the case.

As a result we can, find that when a page is added from the usage space to the TDS, another is brought from the edge of TDS to the center as can be seen in the figure 3.10. so using the user interest can re-rank the results of searching.

At t=4, another pages are visited by the user, the agents add this page to the link and content similarity tables. We found that the system gets transformed to a new form of aggregation and correlation, which becomes higher and higher as shown in figure 3.11. We can see that using ants in the usage space changes the auto organization for the system and gives us more dynamic values where we can observe that the pages start to aggregate together strongly.

At t=6, it is obvious that the system starts to do self organization with addition of every new page as shown in figure 3.12. Agents accept Tags only from those pages whose link similarity is above a given threshold which may be zero or any positive value. From tag space, it can be observed that page aggregation classify pages together according to their tags.

Dead points existing in the X-axis in figure 3.14 represent those pairs of pages (p,q) that have content similarity equal to zero in the UDS.

V. CONCLUSION

Despite the fast technological developments of computational tools for searching the internet, the rapidly growing and dynamic nature of the Web is making static search engines too incomplete and out of date for Web intelligence. Searchers suggested complementing search engines with query-driven online Web mining, to build Web intelligence tools that scale better with the dynamic nature of the Web, allowing for the location of pages that are both relevant and recent. This paper made a review of four models which are summarized as follows:

1. MySpiders: a public Web mining tool deployed as a threaded Java applet. This tool implements the InfoSpiders algorithm, a query-driven algorithm based on an adaptive population of online agents with an intelligent text mining behavior emulating the browsing performed by human users.

2. WACO is an approach, inspired by social insects, to organize dynamically the web content [11]. In this approach, the web is considered as a complex environment, inhabited by artificial creatures called WebAnts. These creatures, implemented by mobile agents [16], are organized in a colony and mimic some behaviours of natural ants, namely the collective sorting behaviour and the food foraging behaviour. The content of the web is viewed by WebAnts as a potential source of food that needs to be organized and collected in an efficient way.

3. WebComp model: In this model, Rattrout tried to study the web as a complex system and as a multi scale space with three components (content, link and usage), and tried to extract the semantic from these components which helps in search process.

4. ContentUsageAnts model: which has the ability to reorganize the content of the Web pages in the content and usage spaces of the Web using CAS and MAS paradigms. ContentUsageAnts model proposed a better way to get results for a user query using the history of and the retrieved pages by the user. Ants in our model crawled the content space and the user space looking for relevant pages according to tags given to the ants. Ants follow the pheromone traces to reach the most related pages to our query. Comb the content and usage space results of similarity measurements give a better results and removes the bad pages from the list of retrieved pages returned to the user.

The researchers keep developing models, improving the search engines work and giving better search results for users.

VI. APPENDIX

![Figure 3.1: looping circle between real space and virtual spaces. In virtual spaces, we can see the interactions exist between agents in one space level and in multi level spaces[13].](image)
Figure 3.2: The Dynamic Web Three Components [13]

Figure 3.3: The domains of ContentUsageAnts model [13]

Figure 3.4: An agent in its environment [13]

Figure 3.5: Building the frequency tables in ContentUsageAnts model [13]

Figure 3.6: The Data Flow In ContentUsageAnts Model [13]

Figure 3.7: The general link and Content similarity distribution for the TDS at t=0 [23].

Figure 3.8: The $\sigma_c$, $\sigma_L$, distribution for TDS at t=0, no correlation exist[23]

Figure 3.9: The general link and Content similarity distribution for the TDS at t=2, we can see that emergence of new changes in the distribution take place [23].

Figure 3.10: The $\sigma_c$, $\sigma_L$, distribution for new TDS at t=2, good correlation emerges, and aggregation can be happens[23]

Figure 3.11: The $\sigma_c$, and $\sigma_L$, distribution for new TDS at t=4, good correlation emerges, and aggregation can be happens between three or more agents[23]
Figure 3.12: The $\sigma_c$ and $\sigma_L$, distribution for new TDS at $t=6$, good correlation emerges, and aggregation can happen between four groups of agents[23].

Fig. 3.13 Disorder decreases while new documents are created, and sorting occurs in WACO model[17].

Fig. 3.14 Clusters mean size and its standard deviation evolution[17].

Fig. 3.15 Mean values and standard deviation values of Searchers WebAnts energy evolution[17].

Fig. 3.16 Population evolution: proportion of active agents/total population of agents[17].

Fig. 3.17: Interrelated relation between the three major items that represent the problematic and the propo [22]

Fig 3.18: the interaction between the components of the Web[22]

Fig 3.19: the architecture of info spider model [14]

REFERENCES

http://oak.cs.ucla.edu/~cho/papers/cho-toit01.pdf
[15] The web from a complex adaptive systems perspective†Maya Ruperta,b,* Amjad Rattrout, Salima Hassasb.