ArabAgent

Table Of Contents

'*ArabAgent*' is a web personalization system used to personalize search and filter news articles. ArabAgent mainly specialized for content that in Arabic language. ArabAgent system uses pure content-based approach to provide recommendations to users.

Table of Contents

1. ArabAgent as an Intelligent Agent

ArabAgent system is inherently an intelligent agent. We next examine some of the important definitions and characteristics of intelligent agent introduced by prominent researchers in the field and try to apply the aspects of the definitions to the case of ArabAgent system.

Russell and Norvig[1] defined agent as, "An agent is anything that can be viewed as perceiving its environment through sensors and acting on that environment through effectors." In light of this definition ArabAgent perceives its environment, which is the user web browser and predefined news websites, through its sensors, which is the changing parameters of the browser through event listeners such as loading web page or a user click, and acting in that browser by amending the web page and search results of certain search engines for the user and send the relevant news articles to user's e-mail.

Maes[2] defined the agent as "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed". According to Maes definition, the ArabAgent could be seen as inhabiting the web browser and the personalizing server as the "complex dynamic environment". ArabAgent senses autonomously by implementing event listeners to sense user events as they happen on the web pages and sense news article as they available in the news websites. ArabAgent acts autonomously by modifying user query, modifying search results and recommend news articles without user intervention. ArabAgent (by monitoring user behaviors, and customizing the results of search and filtering news article) realizes the main goal of individualizing the access to the web to its user for which they are designed.

Gilbert has defined agent as [3] "An intelligent agent is software that assists people and acts on their behalf. Intelligent agents work by allowing people to delegate work that they could have done to the agent software. Agents can, just as assistants can, automate repetitive tasks, remember things you forgot, intelligently summarize complex data, learn from you, and even make recommendations to you." ArabAgent could be viewed as assisting people and acting on their behalf in locating and filtering data and saving time by making decisions about what is relevant to the user. Owning and installing the ArabAgent "software" could be implicitly interpreted as a delegation of filtering news articles and picking relevant search results to the ArabAgent system. Furthermore, ArabAgent "repeatedly" filters news articles and ranks search result hits as the user search again and again, "learns" user interests and preferences, and "recommends" news articles.

Hermans[4] defines agent as "a piece of software that performs a given task using information gleaned from its environment to act in a suitable manner so as to complete the task successfully. The software should be able to adapt itself based on changes occurring in its environment, so that a change in circumstances will still yield the intended result" In the case of ArabAgent, the tasks are filtering news articles and tailoring web search results in order to satisfy the user needs using the user feedback of the relevancy of previous web pages and search results as gleaned information from the environment. We could deem the event that happens within the browser such web page loading and user visiting certain web pages gleaned information as well. ArabAgent acts in a suitable manner while filtering news articles and tagging or ranking search results so help user individualize its view of the web. The ArabAgent adapts itself as user's interests change (this change is realized by monitoring the users behaviors and choices while accessing the internet.

To summarize, ArabAgent system is inherently an intelligent agent. One thing helped in that is that information overload problem in the internet is a typical problem to be solved using intelligent Agents. Information overload required the user "*repeatedly*" to spend a lot of time filtering information on the internet, sifting through hundreds or thousands of web search results, and modifying queries submitted to search engines to improve search results. As a result, ArabAgent is used to assist its user and ("*or acts in behave of its user*") to tailor web search results, modify user query, and filter news articles. ArabAgent accepts user's relevance feedback to *adapt* itself to the new interests of the user. See the ArabAgent as a black box in the next figure. Next section describes the ArabAgent framework.

2. ArabAgent Framework

The next figure shows the ArabAgent framework. The framework could be used as a general framework of agent used for web personalization regardless of its application (e.g. personalized search, navigation assistant, content personalization, etc). It contains the main component needed by most web personalization systems. The framework could accommodate more components and sub-components of services, functions, or processes as needed.

Figure 1: ArabAgent Framework

The ArabAgent system consists of seven components; user profile, user interface, query customizer, web wrapper, ranking and filtering component, user modeler and profiler, and text processing component.

3. ArabAgent Application Architecture

The ArabAgent system has been divided into part; the first part is in the client side and the other side exists in a server side. Although, ArabAgent intended to be a single agent that apply a content-based approach to recommend items to the users, the framework and architecture of the ArabAgent system provides dynamic in design. Since the part of the system resides in a separated server, the ArabAgent could serve as multi-agent collaborative recommendation system with a minor modification to the system to the framework.

Whitten[5] state that there are five general layers that can be distributed differently according to the chosen system architecture. These five layers are the presentation layer, presentation logic layer, the application logic layer, data manipulation layer, and finally the data layer. In addition, there are three system architectures that can contain these layers. These three are *file server architecture*, *client/server architecture*, and *Internet-based architecture*.

Since user interface of the ArabAgent system and the user profile are communicating through the internet and sense each user has its own account, the internet-based architecture suits much the ArabAgent System.

The first layer of the ArabAgent system is the presentation layer. It consists of (what is shown to the user in the monitor. It could be a web page) the Web pages that are displayed by the Web browser. These web pages are formatted and presented using the next layer which is the presentation logic layer. The presentation logic layer of the ArabAgent system is the part that monitors the user’s activity and, collects user relevance feedback and useful data that represents his interests. This part also formats and presents the recommended web pages for the user. Here, the presentation layer and the presentation logic layer represent the user interface of the ArabAgent system.

Figure 2: The Application Architecture of ArabAgent system as Internet based Architecture

The user data that has been collected by the presentation logic layer are send to the application logic layer, or shortly the logic layer, the data is then manipulated and used to update the user profile. Also, the logic layer modifies the user query using the data previously saved and collected about the user. The logic layer then sends the modified query to each search engine individually; after adapting the query into the search engine acceptable format. The logic layer then accepts the results from the Web and ranks or categorizes them according to user's interests. The application logic layer is comprised of the following components; Updater component, query customizer component, the web wrapper, and ranking and filtering component.

The data manipulation layer represents the User modeler and profiler. Data manipulation layer used, here, to access the user profile. Other components can't retrieve or restore data into the user profile except through this layer. User profile is the data layer. It contains user preferences such as search engine preference degrees, URLs visited by the user, the user interests and others.

The ArabAgent system implements the Internet-based Architecture; sometimes called multi-tier architecture. See Figure 2. Each rectangle represents a tier. The first tier, that contains the presentation layer and presentation logic layer, is implemented at the client-side. Intuitively, this tier represents the user interface component.

4. ArabAgent Components

In this section we briefly discuss the component of the ArabAgent framework. For more details about any component, please visit its page.

4.1. Web Wrapper

The web wrapper is the only interface with the Web in the system. The web wrapper is component of the system that responsible of monitoring the web while user accessing the internet. The wrapper takes actions according to certain cases. For example, the wrapper of ArabAgent system adds feedback controls as a response of the web returning web search results from Google search engine.

In addition, the wrapper used to form the data submitted to the internet to suit certain websites. For example composing the query submitted for Google search engine so it (the query follow the Google standard format). It also use for extracting links and web pages from certain web pages such as search engines' pages.

4.2. ArabAgent User Interface

To avoid annoying the user with installing a new software application, we opt to implement the user interface as an extension toolbar. The interface of the ArabAgent system consists of a toolbar extension that monitors user activities and logs them in a special file that could be sent to the personalization server to update the user profile. The toolbar is suitable of Firefox and Internet Explorer browsers.

In addition to monitoring and logging, the user interface accepts external feedback from the user for web pages and search results.

The ArabAgent user interface supports user feedback in many ways. The toolbar extension contains a button control that could be pressed if the user is interested in the web page currently viewed by the web browser. After wrapping the search results with feedback controls, the user interface presents the search results in a way the user could provide feedback for these search results.

If the user interface detect that user is submitting a search query to one of the search engines it monitors, the user interface sends the query to the query customizer to modify it before send it to the search engine.

4.3. ArabAgent Profile Updater

The profile updater component accepts feedback from the user interface. Then, it send it to text processing component. The text processing component send back the concepts identified in the document. The updater component updates the profile (returned from user modeler and profiler) with the concepts return from text processing component. After updating the user profile the updater send back the modified profile to the modeler and profiler component to save the profile.

4.4. Text Processing Component

The main goal of the text processing component is identifying concepts from documents; either these documents are web pages from the user relevance feedback or web pages return as a result from search engines. It also used to identify phrases (not concepts) as a part of user query processing task.

Identifying concepts from documents considers two steps. The first step is to detect phrases that seems to be concepts and assign for each of them the corresponding prospective concept(s) id(s) (in case of polysemy, a term can have several concept ids), and second step is to disambiguate between concepts for phrases that have multiple concepts.

Before anything can get into work, data and statistics that are used in the disambiguation process and semantic relatedness task should be elicited and prepared. Arabic Wikipedia dump is used for this purpose many for many of its advantages.

4.5. User Modeler and Profiler

User modeler and profiler is the component that is responsible for creating user profile (if not exist), loading user model, saving user profile. The user modeler and profiler component is the only component to directly deal with user profile. It receives the updated user profile from the updater component and save it.

The user modeler and profiler component also provide user model for both query customizer component and ranking & filtering component so they can personalize their content.

4.6. User Profile

As a personalization system, one of the key aspects is to maintain the user's interests and preferences. Accordingly, a user profile is an integral part of most personalization systems. The data about the user is stored in user profile
In ArabAgent, the user profile is stored in as an

profile.xml

- Details
- Download
- 156 KB

containing data of the user.

4.6.1. User Profile vs. User Model

We here differentiate between two concepts; user profile and user model [6] [7] . User profile stores the raw data of the user. For example, a user profile for a web personalization system could store web pages the user visited, certain domain the user always considers, user queries and other raw data that has not exposed to analysis or exposed to a little. In the other hand, user models are more sophisticated than user profiles. User model is the result of analyzing the data in the user profile; sometimes is called *modeling*. Usually, the user profile is used store the user data in a persistent way for later analysis; for instant the user profile could be a file in a secondary storage. However, user model is essentially used at the application run-time. Therefore, the user model is usually in the memory.

4.6.2. Representing User in ArabAgent

Most of the systems (that are content-based approach) try to represent user interests through representing the keywords of the documents that user showed interesting in, and they claim that they representing user interests. However, in ArabAgent system, the matter is tackled differently. According to Lawrence Page[8] "The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes". Accordingly, ArabAgent attempts to represent the user's interests, knowledge and attitudes to define the importance of a web page to the user as a way to individualize the access to the web.

The user model is represented using two structures; *semantic networks* and *hierarchy of categories*.

4.7. Query Customizer

The query customizer is responsible for modifying user query to suit his/her user need. It could expand user query or alter some keywords or search. The query customizer could narrow the search by limiting it to specific websites (for example websites the user likes to visit).

4.8. Ranking and Filtering Component

The ArabAgent system personalizes user experience using different ways depending on the task the user perform. If user is search the web using one of the monitored search engines (such as Google), the ArabAgent system (using ranking and filtering component) tag the results return by the web search engine by signs to indication the likelihood of the HIT to be relevant to the user.***advantage of trust***

Because the previous method may make the user sift through loads of hits to satisfy all his/her needs, we support the previous method by additional service that ranking the result again. The new rank of the hits is supported in a sidebar (so we make the user feel that we don't interfere in his decision so he feels save).

Furthermore, the ArabAgent system provides filtering techniques for some news websites. The ArabAgent checks for news availability in the websites every five minutes. The ranking and filtering component filters the new news article and if the article is likely to be relevant it send it to the user email.

5. Systems Similar to ArabAgent

The systems INFOrmer[9] [10] , ifWeb[11] , and SiteIF[12] use semantic networks to represent the documents that the user shows interest in them. However, they represent the user differently; while they connect tokens and words exist in the same sentence only (and the order of the words in the sentence determines the direction of the edge between the two words), ArabAgent connect all the phrases that exist in a web page to each other. The semantic networks in the ArabAgent are weighted undirected graph and weights are represented differently.

In all the system that have been mentioned the nodes of the networks represent single token or word (not a phrase). However, in ArabAgent each node is corresponding to a phrase (that may consist of one word or more).

Syskill & Webert[13] and PEA[14] use the same way of personalizing techniques as in ArabAgent. They tag search results to indicate the importance of the link to the user.

1. ^ Russell, J., and P. Norvig. Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ: Prentice Hall, 1995.
2. ^ Maes, P. “Artificial Intelligence Meets Entertainment: Life-Like Autonomous Agents.” Communications of the ACM,November 1995.
3. ^ Gilbert, D. “Intelligent Agents: The Right Information at the RightTime.” IBM white paper, May 1997. citeseer.nj.nec.com/context/1105800/0 (no longer available online.)
4. ^ Hermans, B. “Intelligent Software Agents on the Internet.” 1996 (updated 2000). hermans.org/agents/h22.htm (accessed July 2011).
5. ^ Jeffrey L. Whitten, Kevin C. Dittman, and Lonnie D. Bentley. *Systems Analysis and Design Methods*. 2004. Chapter 13
6. ^ Nora Koch. Software Engineering for Adaptive Hypermedia Systems. PhD thesis, Ludwig-Maximilians-University Munich/Germany, 2000.
7. ^ Fröschl, C. (2005) User Modeling and User Profiling in Adaptive E-learning Systems, Master Thesis, University of Technology, Graz.
8. ^ Page, L., Brin, S., Motwani, R., and Winograd, T., The Pagerank citation ranking: Bringing order to the web, Technical report, Stanford University (1998).
9. ^ O’Riordan A. and Sorensen H., An Intelligent Agent for High-Precision Text Filtering, 1995.
10. ^ Sorensen H., O’Riordan A. and O’Riordan C., Profiling with INFOrmer Text Filtering, 1997.
11. ^ Asnicar, F. A. and Tasso, C. (1997) ifWeb: A prototype of user model-based intelligent agent for document filtering and navigation in the World Wide Web. In: P. Brusilovsky, J. Fink and J. Kay (eds.) Proceedings of Workshop "Adaptive Systems and User Modeling on the World Wide Web" at 6th International Conference on User Modeling, UM97, Chia Laguna, Sardinia, Italy, June 2, 1997, Carnegie Mellon Online, pp. 3-11, http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Tasso.html
12. ^ A. Stefani and C. Strappavara. Personalizing Access to Web Sites: The SiteIF Project. In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT’98, June 1998.
13. ^ PAZZANI, M, MURAMATSU, J, AND BILLSUS, D. "Syskill & Webert; Identifying interesting Web sites", In Proceedings of the National Conference on Artificial Intelligence (AAA I96) (1996),
14. ^ M. Montebello, W. Gray and S. Hurley. A Personable Evolvable Advisor for WWW Knowledge-Based Systems. In Proceedings of the 1998 International Database Engineering and Application Symposium (IDEAS’98), July 1998, pp. 224-233.

The personalization task depends on the application on hand

The personalization techniques used in ArabAgent system related to the design of the user interface.

1. Personalized News Filtering

The ArabAgent system provides filtering techniques for some news websites. The ArabAgent check for news availability in the websites every five minutes. The ranking and filtering component filters the new news article and if the article is likely to be relevant it send it to the user email.

2. Personalized Search

Most of the web personalization systems that deal with Web search as its application have conquer the problem by either treating web search result or the user original query. The systems that deal with search result use varety of methods such rank again, or so-called re-ranking, filter, or tag the result hits of the search engine.

Due to time constrains, it is difficult to ranking again all the result of the search engines. The user can't tolerate the relatively long time the system takes to rank all the results. Furthermore, the results are going to be restricted to the result set which the search engine show to you, is user query which may poorly represent the user needs. In that case, ranking again is also restricted as well to this set of results missing a great opportunity to find other good web pages that may satisfy the user needs.

Two systems treated the problem differently; *Syskill & Webert*[1] and *PEA*[2]. They tagged each HIT of a search engine result by a tag to indicate it relevancy likelihood to a user. Although, the user gradually gains trust on the system as he search again and again, as the system's tags meet the user's expectations, (since he could evaluate the tagged results while he wade across the hits). However, later, the user finds himself wades across hundreds of hits pursuing the ones tagged with high relevancy likelihood.

To resolve this problem in ArabAgent, we have combined both techniques (results tagging and result re-ranking) but with a different way. As user submits his query to the search engine, the system shows the original order and rank of the search engine result. However, the hits are tagged with the relevancy likelihood to user. In addition to tagging, we provide the ArabAgent special rank of the result in a sidebar that the user could show or hide.

In the sidebar, the hits of a result are shown immediately as they retrieved. There is no necessity to wait for all the hits to be ranked to show them all as a bunch. The hits appear to the user in one by one manner, as they weighted by the ranking and filtering component . As new hit given a weighted, the order of the result changes immediately, so the user notices that some hits' positions are changing and others are subsiding (some sink and the others emerge to the top of the result list). This technique of displaying the result prevents the user from feeling the tardy arrival of the result. Furthermore, this technique prevents user wading across hundreds of hits pursuing the ones tagged with high relevancy likelihood.

2.1. Query Personalization

The methods and techniques the deal merely with search result set, has a great disadvantage. Although, it tries to improve the effectiveness of the system by improving its precision at particular rank, it totally ignores the recall measure. The reason behind that is that the system is confined only to the set of result return by a certain query. This could result decline in the overall system performance in case the user submitted bad query keywords (the result hits contains no relevant data at all). Accordingly, some systems opt to personalize the search by modifying the user original query to avoid these drawbacks.

In ArabAgent system, the query customizer is the responsible component for modifying user query to suit his/her user need. It resolves the previous problem by introducing modified query to the search engine. It could expand user query or alter some keywords or search. The query customizer could narrow the search by limiting it to specific websites (for example websites the user likes to visit).

It seems that Modifying user query is good solution to the disadvantages of techniques handle only search result. However, modifying user query is tricky and difficult. You cannot augment new keywords to the user original query unless you are hundred percent sure they are going to improve the result. The keywords could take the query to another direction causing the result to decline in terms of both precision and recall.

1. ^ PAZZANI, M, MURAMATSU, J, AND BILLSUS, D. "Syskill & Webert; Identifying interesting Web sites", In Proceedings of the National Conference on Artificial Intelligence (AAA I96) (1996),
2. ^ M. Montebello, W. Gray and S. Hurley. A Personable Evolvable Advisor for WWW Knowledge-Based Systems. In Proceedings of the 1998 International Database Engineering and Application Symposium (IDEAS'98), July 1998, pp. 224-233.

The **Ranking and Filtering Component** is the other component used in personalizing user access into the web (i.e. the first component is the query customizer component). The ranking and filtering component is mainly responsible in weighting web pages. It is used to filter news articles as they available in their websites as well as tagging search results' hits during search process. It produces a weight representing the relevancy likelihood of the document in respect to user interests and knowledge.

After loading the search result in user interface, the interface communicates with the ranking and filtering Component asynchronously to either rank or tag the result set. The asynchronous property prevent the user from feeling the relatively long time the ranking and filtering Component takes to weight the result hits.

Figure 1: Ranking and Filtering Component

Before ranking and filtering Component calculates the weight of a web page, it sends the web page to the text processing component. The text processing component identifies the concepts of the web page and sends the concepts back to the ranking and filtering component. The ranking and filtering component then build a graph of the concept and their frequency. The graph of the Web page is then compared to user model graph to compute the relevancy likelihood weight of the web page.

In case of filtering news articles, an additional step is performed after computing the relevancy likelihood weight of the news article. The ranking and filtering component has to decide the relevancy of the article, whether relevant or irrelevant. Accordingly, a decision is made whether to send the article to the user email or not.

The main goal of the **text processing component** is identifying concepts from documents; either these documents are web pages from the user relevance feedback or web pages return as a result from search engines. It also used to identify phrases (not concepts) as a part of user query processing task.

The text processing step is fully or partially required for the following: user profile updating Task, filtering and customization Task, and query modifying task.

Identifying concepts from documents considers two steps. The first step is to detect phrases that seems to be concepts and assign for each of them the corresponding prospective concept(s) id(s) (in case of polysemy, a term can have several concept ids), and second step is to disambiguate between concepts for phrases that have multiple concepts.

Preparing Data from Wikipedia

Before anything can get into work, data and statistics that are used in the disambiguation process and semantic relatedness task should be elicited and prepared. Arabic Wikipedia dump is used for this purpose many for many of its advantages. The Wikipedia dump is used to generate the following tables that stored in files:

- concept_inlinkCount:
- concept_outlinkCount
- concept_inConceptList
- concept_outConceptList
- term_conceptsList

For more information of these files generated from Wikipedia please visit the Preparing Data from Wikipedia page.

As Manning[1] mentioned, there are three type of user feedback techniques; (1) explicit user relevance feedback, (2) implicit user relevance feedback and (3) blind relevant feedback. Explicit user relevance feedback, or shortly explicit feedback, is the type of feedback the user provides manually by, for example, choosing between relevant and irrelevant set of documents or by providing rate of relevancy to a document. Explicit feedback is most trusted and reliable source of feedback. However, explicit feedback constitutes a burden to the user and could sidetrack him from his search goals.
Implicit user relevance feedback, or shortly implicit feedback, is type of feedback that is used to collect data and information about the user navigation and search behavior without the intervention of the user. The liger time, opened web pages, etc. Although this type of feedback overcome the problem of explicit feedback, the certainty of its relevancy is not hundred percent guaranteed.

In search context, Blind feedback assumes that top n documents are relevant. According to that, it extracts m keywords with highest frequencies of occurrences from those n documents to expand the user query. Although this assumption is not true, the results of expansion have higher performance than those without blind relevance feedback [reference]. However, the relevancy of such feedback is less certainty even from implicit feedback.
Although, using implicit techniques of the user relevance feedback techniques for web personalization systems are very beneficial since they keep the user focus in his goal without sidetracking or distraction attention from his own goal, ArabAgent system applies explicit techniques to focus mainly on the task of user modeling and expressing user context without being concerned by the efficiency and accuracy of the implicit techniques used to judge on the relevancy of an item or object the user inspect.

ArabAgent provides two ways for explicit feedback; while searching and while browsing the Web. As the user browses the Web he or she could tag a Web page as relevant. In Addition, for each search result of a search engine, he could assign a relevancy status since there is a feedback input associated with every Hit in the result.

1. ^ Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, ISBN: 0521865719

The **user interface** is considered a very important part of any system especially those where directly faced with non-technical non-specialist users of the system such as the majority internet users. One of the main reasons of the success of Google search engine is their simple interface. The interface of Google search engine is very simple consisting merely of text box and a button.

In ArabAgent, we considered this simplicity in the user interface. We also consider not to face the user with new features and properties that he/she is not familiar with and needed to be learned about. As a result, the ArabAgent interface is chosen to be a toolbar extension for most of the prevalent web browsers such as Firefox web browser, Internet Explorer web browser and chrome web browser.

The extension of ArabAgent could appear as toolbar or a button for the web browser. The toolbar has only two buttons; one is labeled options and the other as feedback. Using the "option" button, user can adjust few parameters of ArabAgent system such search engines he or she needs to track, news websites he or she interested in and form of news articles are going sent to his/her email. While user is navigating the web sometimes he/she finds a web page is useful in this case the "feedback" button is used to feedback the page. As user marked a hit as relevant, the user interface sends the user query and the hit's URL to the updater component.

As user writes his query and presses the search button, the user interface sends the user query to query customization component before sending it to the search engine.

Using this toolbar extension allows to automatically append feedback controls to search engine results. After detecting a result page of any tracked search engine, the system add two radio buttons for each hit in the search result; one labeled are "relevant" and the other labeled as "irrelevant". In case the user find the hit relevant he could choose relevant. The extension could be used later to supports implicit user feedback (feedback without user intervention) while user browsing, reading, bookmarking, saving, and printing web pages.

The user interface tags the search result hits with color symbols that indicate the relevancy likelihood of the hits. The symbols take color between red (which means far away from the user interest) and green (which means strongly recommended for the user). Each hit in the search result page is tagged with the symbol during page loading process. Each symbol sends the URL of the hit to the ranking and filtering component asynchronously to load its color using the Ajax web technology.

User model the system view of its user. Koch[1] describes a user model as the representation of the system’s beliefs about the user, see[2] . However, there is ambiguity among some between user model and user profile.

We here differentiate between two concepts; *user profile* and user model [3] [4] . User profile stores the raw data of the user. For example, a user profile for a web personalization system could store web pages the user visited, certain domain the user always considers, user queries and other raw data that has

Table of Contents

not exposed to analysis or exposed to a little. In the other hand, user models are more sophisticated than user profiles. User model is the result of analyzing the data in the user profile; sometimes is called *modeling*. Usually, the user profile is used store the user data in a persistent way for later analysis; for instant the user profile could be a file in a secondary storage. However, user model is essentially used at the application run-time. Therefore, the user model is usually in the memory. This page discusses the user model of the ArabAgent. You could refer to user profile of the ArabAgent in the ArabAgent user profile page.

Most of the systems (that are content-based approach) try to represent user interests through representing the keywords of the documents that user showed interesting in, and they claim that they representing user interests. However, in ArabAgent system, the matter is tackled differently. According to Lawrence Page[5] "The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes". Accordingly, ArabAgent attempts to represent the user's interests, knowledge and attitudes to define the importance of a web page to the user as a way to individualize the access to the web.

To make the user model reflect the above envision, two structures have been combined together to comprise the user model; a *hierarchy of categories* and *semantic networks*. See Figure 1.

Figure 1: Sample User Model

Each time a new interest appears; new node is added to the semantic networks to represent both his knowledge and interest values. While the user interests may change (decay) over time, user knowledge is maintained persistent. This has been represented using the structure itself of the semantic network and two attributes tagging the nodes and the edges in the network. The first attribute is dubbed “lt-interest”, and represents long-term interests and the second attribute named "st-interest" and represents the short-term interests of the user.

Our user modeling provides the following:

1. General knowledge through a hierarchy of categories; the general knowledge provides a source of knowledge domain to help anticipate new topic that may concern the user but he don’t know about its existence. This solve the problem of serendipity through a comparison process between the user knowledge and the general knowledge,
2. User Knowledge through the semantic networks connecting nodes, which represent concepts the user was (and still) interested in, and the edges represent concepts that user like to co-occur in the same document. Weights of both nodes and edges represent the amount of experience the user has in a particular topic and how often two or more concepts co-occur,
3. Representation of user interests through represented by "st-interest" attribute. The user interest is represented as an attribute tagging each node and edge in the semantic network. The attribute value is decay over time as the concept for a corresponding node stop occurring in user’s interested pages or concepts stop co-occur together.
4. The capability of representing short-term (transient or ephemeral information needs) and long-term (persistent information needs) interests through hierarchy of categories. Long-term interests usually are the broad topics of the short-term interests. This does not mean that we consider every broad topic of each instant short-term interest. However, to be a long-term interest its subcategories should be mentioned frequently. A topic of a leaf category could be a long-term interest as well, if it has been mentioned frequently for a long period of time,
5. The capability of representing short-term and long-term interests through weights of edges connecting concepts in the semantic Networks. Long-term interests are represented through the semantic networks by determining the lowest condition probability to be the long-term interests and the high condition probability to be the short-term interest.

1. ArabAgent Hierarchy of Categories

Each category in the hierarchy embraces one or more concepts. The hierarchy constructed once and forever (but the attributes are updated frequently). The Hierarchy of categories is generated using the Arabic Wikipedia project. Before the initial installation of ArabAgent system this hierarchy of categories and their concept nodes need to be generated.

The Arabic Wikipedia project is used to be the source of both hierarchy of categories and their concept nodes for many reasons. Furthermore, The Arabic Wikipedia project is used to elicit data and statistics needed for phrase detection and phrase sense disambiguation tasks.

2. ArabAgent Semantic Networks

The nodes of the semantic networks are the concepts of the categories. The Edges connecting between two nodes are undirected and tagged with two weights; one represents the long-term interests and the other represents the short-term interests. Edges of the semantic network connect between two concepts if they co-occur in the document. Each concept or edge is tagged with two attributes; one represents the short-term interest of the user toward this concept or edge and the other represents the long-term interest.

The semantic networks are built gradually and updated frequently as user provides relevance feedback to update the user profile. As profile is updated, edges in the semantic networks are created to connect between the concepts as they co-occur in the relevant documents and the attributes values of the concepts are (occurrence and interest attributes) updated as well. The calculation of the attributes of both short-term interest and long-term attributes is done through the process of updating the user profile.

3. Building Short-term Interests Calculating Formula

In this section we build two formulas that calculate the weights of short-terms for both concepts and edges. We gradually build these two formulas to help the reader understand the justification of building them.

3.1. Building Short-term Interests Calculating Formula for Concepts

The function that is going to be built computed each time the system update the user profile since new information is available and should be considered in the user model. The function or the formula represent the interest of the user over 30 days. It takes into consideration number of times that each concept updated, the recency of days where updates occur, and the distribution of updating incidence for each concept and connection over time.

As the formula takes into consideration the number of times a concept updated, or for short CUF, stands for *concept updating frequency*, this present in the initial form (1) of the formula. Note that the system performs a single update operation for each document.

$$st\_interest(c) = \sum_{i=1}^{30} NoOfDocumentsContainTheConcept(c,i)$$

Where (c) is the concept we need to compute the user interest for and (i) is an itteration index for the last 30 days of the profile.

However, concept updating frequency is not completely sufficient. Consider the two cases when two users have mentioned a particular concept in 20 documents. However, in the first case each document of the 20 has mentioned the concept only 1 time whereas in the other case each document has mentioned the concept about one fifth of the total occurrences of the concepts in the document. Using the number of documents that present a certain concept is not enough to represent the importance of a concept for the user. Therefore, the formula (1) needs to adapt to:

$$st\_interest(c) = \sum_{i=1}^{30} \sum_{DocCount(c)} \frac{ConceptCount(DOC_c)}{AllConceptCount(DOC_c)}$$

Another reasonable factor to consider is the positions of days in day frequency; day recency. Suppose that the two users have considered relevant the same set of documents for instance over five days, but the first user considered them in the beginning of the 30 days and the other user considered them in the end of the 30 days. It seems that the second user should be more interested in the concept. To overcome this problem, each day of the 30 days has been given a weight. The weight is computed by calculating the difference between the day calculating is held and the day the document considered relevant in, and dividing the result by 30. Combining this weight with formula (2) yields the final formula:

$$dayImportance(date) = \frac{30 - (currentDate - date)}{30}$$

$$st\_interest(c) = \sum_{i=1}^{30} \left(dayImportance(Date(i)) * \sum_{DocCount(c)} \frac{ConceptCount(DOC_c)}{AllConceptCount(DOC_c)} \right)$$

The above equation considers the following:

- Document weight in the calculation.
- Documents count in one day in the calculation.
- Differentiate between the days by considering day recency.
- The number of days

3.2. Building Short-term Interests Calculating Formula for Edges

The edge weight is used mainly to show how closely the concept are related in respect to the user, in other words, the extent the user is interested in documents possess both concepts. A first intuition is to consider the number of the documents mentioned both the concepts against the documents that mentioned at least one of the concepts. This yields the following formula:

$$st\_interest(C_1, C_2) = \frac{|documents_{C1,C2}|}{|documents_{C1}| + |documents_{C2}| - |documents_{C1,C2}|}$$

Where C_1 and C_2 are the two parties of the edge, $|documents_{C1}|$, $|documents_{C2}|$ are the number of documents that mentioned concepts C_1 and concept C_2, respectively, and $|documents_{C1,C2}|$ are the number of documents have both the concepts.

The importance of an edge is related to the importance of both its concepts. If concepts of an edge don't interest the user, the edge itself (which means the connection between these two concepts) will not interest the user. However, if the two concepts highly interest the user, the edge definitely will concern him a lot. As a result, one can not treat all edges (the parties of the edges) as same. The edge is represented by the minimum importance of the concepts participating in the edge in a single document. The new formula is as follows:

$$st\_interest(C_1, C_2) = \frac{\sum_{|docs_{C1,C2}|} MIN(IMP(C_1), IMP(C_2))}{\sum_{|docs_{C1}|} IMP(C_1) + \sum_{|docs_{C2}|} IMP(C_2) - \sum_{|docs_{C1,C2}|} MIN(IMP(C_1), IMP(C_2))}$$

Where IMP(C_1) is the importance of the concept in the document, and MIN(IMP(C_1), IMP(C_2)) is the minimum of the importance of C_1 and C_2 (MIN function choose the lowest importance between these two importance).

The previous formula seams perfect, however, the formula does not considers the decay of user interests over time. Accordingly, the importance of the day is considered as in the concept interest formula before. So, the formula is going to be as follows:

$$st\_interest(C_1, C_2) = \sum_{i=1}^{30} (dayImportance(Date(i)) * \frac{\sum_{|docs_{C1,C2}(i)|} MIN(IMP(C_1), IMP(C_2))}{\sum_{|docs_{C1}(i)|} IMP(C_1) + \sum_{|docs_{C2}(i)|} IMP(C_2) - \sum_{|docs_{C1,C2}(i)|} MIN(IMP(C_1), IMP(C_2))})$$

The previous formulas aren't calculated directly to load the user model. However, part of the above calculation is done while updating user profile profile and the reset of the calculation is done in the in loading from user profile to user model task (just the importance of the concept in the day is calculated for the user profile and assigned as value to the attribute SCISD, also the sum of the minimum participant of edges for that day is calculated and assigned to SMP attribute). For more details about the calculation done while user profile updating refer to the ArabAgent user profile page .

The next section discusses the part of calculations done while modeling the user profile.

4. Loading User Model from the User Profile

1. For each concept (c) in the profile do the following:
    1. make a corresponding node in the semantic network
    2. conceptImportance = 0
    3. for each day (d) for the concept (c) do the following:
        1. conceptImportance = conceptImportance + (importanceOfDay(d) * SCISD)
2. For each connection (e) in the profile do the following:
    1. add two edges; edge(e.to, e.from) and edge(e.from, e.to) to the semantic network because the edges are undirected
    2. edgeImportance = 0
    3. for each day (d) for the connection (e) do the following:
        1. edgeImportance = edgeImportance + (importanceOfDay(d) * (SMP/(conceptImportance(e.to, d) + (conceptImportance(e.from, d) - SMP))

The importance of day is computed as the difference between the day of today and the day of the date in the "date attribute" of the concept or connection.

1. ^ Nora Koch. Software Engineering for Adaptive Hypermedia Systems. PhD thesis,

Ludwig-Maximilians-University Munich/Germany, 2000.
2. ^ Heckmann, D.: Ubiquitous User Modeling. PhD Thesis, Saarland University, Saarbrücken, Germany (2005).
3. ^ Nora Koch. Software Engineering for Adaptive Hypermedia Systems. PhD thesis, Ludwig-Maximilians-University Munich/Germany, 2000.
4. ^ Fröschl, C. (2005) User Modeling and User Profiling in Adaptive E-learning Systems, Master Thesis, University of Technology, Graz.
5. ^ Page, L., Brin, S., Motwani, R., and Winograd, T., The Pagerank citation ranking: Bringing order to the web, Technical report, Stanford University (1998).

As a personalization system, one of the key aspects is to maintain the user's interests and preferences. Accordingly, a **user profile** is an integral part of most personalization systems. The data about the user is stored in user profile and administrated by a user modeling system[1] as stated by Heckmann[2] .

We here differentiate between two concepts; *user profile* and user model [3] [4] . User profile stores the raw data of the user. For example, a user profile for a web personalization system could store web pages the user visited, certain domain the user always considers, user queries and other raw data that has

Table of Contents

not exposed to analysis or exposed to a little. In the other hand, user models are more sophisticated than user profiles. User model is the result of analyzing the data in the user profile; sometimes is called *modeling*. Usually, the user profile is used store the user data in a persistent way for later analysis; for instant the user profile could be a file in a secondary storage. However, user model is essentially used at the application run-time. Therefore, the user model is usually in the memory.

In ArabAgent, the user profile is stored in as an XML file containing partially analyzed data of the user (this level of analysis is just for reducing modeling time). The user model comprises of semantic networks that represents users knowledge, long-term interests and short term-interests. This page discusses user profile of the ArabAgent. You could refer to user model of the ArabAgent in the user model page.

1. User Profile

To obtain such user model, the user profile has to store the sufficient data of the user. One could suggest to store each and every tiny piece of data while user accessing the web so we grantee that we have all the data we need. However, this is not not efficient since the profile will get bigger and bigger and the system will consume a lot of resources in storing such data and processing it. A more efficient way is to store just data with a sufficient degree analysis.

In ArabAgent, we have maintained an profile.xml to store user data. The XML file stores list of concepts and list of connection. Each concept in the list of concept contains the days that the concept mentioned in. Each day has three attributes; the first attribute, dubbed "date", is date of the day. The second attribute is the number of web pages contain that concept, dubbed "docCount". The third attribute is the sum of the importance of the concept in each web page contains the concept in that day and dubbed "SCISD".

For connection list, each connection contains the days the connection appeared in.Each connection has two attributes; "from" and "to", to hold the concept ids of the two parties of the connection. Each day in the connection has three attributes as well. The first attribute is the date as in the concept and called "date". the second attribute is the number of documents the two parties co-occur in and dubbed "docCount". The third attribute is the sum of the connection importance for the documents and dubbed "SMP".

These attribute are updated frequently as we will see soon in updating subsection.

2.1. Calculating the Importance of the Concepts and Connections in A Web Document

Calculating the importance of a concept in a web page is simple. After identifying concepts of the web page, we count the occurrences of the concepts in the web page. The importance of a concept is the concept occurrence in that page divided on the all occurrences. This importance is similar to term frequency that used in information retrieval to compute the weight of the term. However, instead of considering all terms, we consider only concepts of the page.

$$conceptImportance(c,p) = \frac{occurrenceOfConceptInThePage(c,p)}{occurrencesOfAllConceptInThePage(p)}$$

Where c is the concept we need to compute the importance for, and p is the web page.

The importance of a connection between two concepts in a web page depends on the concept Importance of both concepts in that page.

$$connectionImportance(c_1,c_2,p) = MIN\left(PageImportance(c1,p), PageImportance(c1,p)\right)$$

$$\begin{aligned} &connectionImportance(c_1,c_2,p) = \\ &\quad MIN\left(conceptImportance(c_1,p), conceptImportance(c_2,p)\right) \end{aligned}$$

Where c1 is the first part of the connection and c2 is the second part of the connection, and p is the web page.

2.2. Updating User Profile

The user profile is updated frequently to adapt to the frequently changing user interests and needs. The calculation of the attributes of both short-term interest and long-term attributes is done through the process of updating the user profile. The new concepts are added to the user profile while updating task as well.

1. ^ Wahlster, W. and Kobsa, A. (1989). User models in dialog systems. In Kobsa, A. andWahlster,W., editors, *User Models in Dialog Systems*, pages 4–34. Springer, Berlin. (Symbolic Computation Series).
2. ^ Heckmann, D.: Ubiquitous User Modeling. PhD Thesis, Saarland University, Saarbrücken, Germany (2005).
3. ^ Nora Koch. Software Engineering for Adaptive Hypermedia Systems. PhD thesis, Ludwig-Maximilians-University Munich/Germany, 2000.
4. ^ Fröschl, C. (2005) User Modeling and User Profiling in Adaptive E-learning Systems, Master Thesis, University of Technology, Graz.

This Page is still under Construction

The categorization of Wikipedia pages and their link structure are available as SQL tables, so that they can be exploited without parsing the actual Wikipedia articles. The category and categorylinks tables are the two files of the Wikipedia dump that contains that knowledge about the categories, their structures and which articles belong to which categories. The files are downloaded to extract the hierarchy of categories.

Each category represented by the concepts that belong to it. The process of concept extraction is discussed. After the extraction of concepts, the concepts are associated to categories of their corresponding Articles.

As we mentioned above, the use model consists of hierarchy of categories and semantic networks. The categories comprise some concepts that corresponding to different topics that may interest user. The semantic networks are built gradually and updated frequently as user provides relevancy feedback. However, hierarchy of categories is constructed for one time i.e. at the deployment of the system.

The Arabic Wikipedia project has been chosen to be a source to provide such hierarch of categories and concepts that used to build the user model. Furthermore, Arabic Wikipedia project has facilitated building a word sense disambiguation technique based on Wikipedia link structure to detect and disambiguate between concepts to use in the text processing component.

Sub-component of Comparing User Model Graph with Web Page Graph

Context terms are terms that have only one sense each and therefore define the context of the document they are in.

During phrase sense disambiguation step, the phrases with only one sense are determined. These phrases help to disambiguate other phrases with more than one sense. The the disambiguation technique computes the semantic relatedness between the senses of the context terms and each
prospective sense of the phrases needs disambiguation.

Evaluating cc

This Wikispace is about ArabAgent System. ArabAgent system is a web personalization system used to individualize web activities and experience to over Arabic content (Web pages and links) on the internet. It is used to personalize web search results (Google so far) and filter news articles (Ahram Gate so far).

The ArabAgent is in a form of a toolbar extension that could be installed in either Firefox browser or Internet Explorer browser. The toolbar connects with a personalization server (where you user profile is) either to update your user profile or filter out and customize web search results and news articles. You Could download the Firefox toolbar or Internet Explorer toolbar and try it.

The ArabAgent system is a part of a master thesis named " An Intelligent Agent for Arabic Web Information Retrieval" by Mohamed Ibrahim El-Desouki, a student at "Institute of Statistical Studies and Research" at "Cairo University".

For testing ArabAgent System please refer to one of the following:
(Note: since the application is uploaded to the free web hosting service "Google App Engine", it has only a limited quota to work with. So you may find one of the links is not working)

- http://elwakeel-elaraby.appspot.com/
- http://disooqi.appspot.com/

To use ArabAgent system you have to have an account on google to hold you user profile.

The thesis is supervised by:

1. Dr. Mervat Gheith
2. Dr. Waleed Arafa
3. Dr. Kareem Darwish

Contact me at:
Email: disooqi@gmail.com
Mobile phone: (+20) 01116500535

The **personalization server** comprises the web server tier, application server tier and data manipulation tier, and data tier (or the user profile). In the other hand, the ArabAgent client is the part that runs in the user machine (which here is the toolbar extension).

The **term detection step** goes as follows: after tokenizing the document, the tokens are normalized; using the unified normalization[1] . The document then is processed to generate word n-grams. The n-gram generation process differs from the usual way of producing n-gram; the concept ids are assigned during the n-gram generation process. See the algorithm in the box below. While the system generates n-grams, it tries to match the n-gram to the synonyms of each different concept and assign the concept id(s) to the term in case a match occurs. Phrase or n-grams of several concept ids is saved for latter disambiguation in the second step. The size of the n-gram, n, is equal to length of synonym with maximum length. Although, there is little likelihood to produce wrong phrases, the customized method for generating n-gram has the advantage of reducing ambiguity by trying to produce longer phrases first.

```
Input: TokensQ (queue of all document tokens), nList (a list of n-gram
 size), synDic (synonym-concept_ids dictionary),
n (size of n-gram)
Output: list of phrases, each with its prospective concept(s)
Algorithm:
1)If TokensQ size = 0, then return;
2)Else If TokensQ size >= n, Choose first n tokens from the TokensQ in
to nList.
3)Else, choose all tokens from the TokensQ into nList.
4)Constitute a phrase by concatenating all the tokens in nList.
5)Try to find a corresponding synonym(s) for the phrase.
6)If (synonym(s) found in synDic)
  a)Assign the concept_id(s) to the phrase.
  b)Empty nList.
  c)Go to step 1.
7)Else (the phrase has no corresponding synonym)
  a)Then remove one token from the end of nList.
  b)Check the size of nList after removal
    i)If number of tokens that exist in nList > 0, then go to step 4.
    ii)Else, go to step 1.
```

The stopwords removal process begins after the detection process and the reason for that is some phrases may contain stopwords, which will not be matched if we remove the stopwords before the n-gram process. For example, the phrase “كان مهرجان” if we remove stopwords before n-grams process the word “كان” is going to be remove, but if we removed stopwords after n-gram process nothing going to be removed. (In Wikipedia, stopwords don’t have corresponding articles, so stopwords don’t exist as concepts in the synonym-concept_ids dictionary).

Many phrases may refer to the same concept, so the appearance of one of them in the document makes it subject to replace by the equivalent concept id; this solves the problem of synonyms. In case of polysemy problem, phrases might lie under several concepts. The second step arise here as a technique to disambiguate between the several concepts and choose the right concept id. Techniques are illustrated in the next section. As a result unwilling match is prevented with the same spelling but with different in meanings phrase.

After replacing all the phrases and terms by their right concept id, we treat the document as if it is “bag of words”; however, it is actually a page of concepts. The rest of the information retrieval steps remain as it is. An extra step is to go through the previous steps manually for substituting the query’s phrases by the intending concepts ids.

1. ^ El-Disooqi M., Arafa W. and Darwish K. Stemming techniques of Arabic Language: Comparative Study from the Information Retrieval Perspective. The Egyptian Computer Journal , Vol. 36 No. 1, June 2009.

Table of Contents

If an n-gram has multiple concepts, then a concept disambiguation is going to happen. The disambiguation process starts after the phrase detecting process is completed for the document since the disambiguation process depends on all the phrases which posses a single concept of document; context terms. Multiple techniques have been used for evaluation and comparison reasons. All techniques used in experiments depend on the Arabic Wikipedia statistics such as in-links and out-links of an article, number of out-links of an article and others. The disambiguation techniques that have been tried are inspired from Milne and Witten[1] [2] and Cilibrasi and Vitanyi[3] ; however, neither machine learning techniques nor link probability are used.

The techniques used to disambiguate could be categorized in to two sets, the first set contains one technique which choose the most common sense among the senses of a phrase, the other category or set (which includes three techniques) depends on so-called semantic relatedness. the semantic relatedness could be computed based on in-links, out-links, or both the in-links and out-links of a Wikipedia article.

1. Using 'Most Common Sense' to Disambiguate Senses

Different techniques have been examined to disambiguate between concepts and each has been evaluated. First technique is choosing the most common concept among all concepts. The commonness measure depends on number of articles refereeing to the article that representing the concept and is calculated by dividing the number of in-coming links to the page representing that concepts divided by the sum of numbers in-coming links of all concepts being disambiguated.

2. Using 'Semantic Relatedness' to Disambiguate Senses

The disambiguation techniques depends on so-called semantic relatedness between concepts. Different techniques have been examined to calculate the semantic relatedness between concepts. to disambiguate between different senses of a phrase, we used the following algorithm:

1. After detecting all the phrases of the document, the all the context terms are identified
2. Then, the weights of the context terms are identified (the weight of the context term show its contribution in the context of the document)
3. Then, for each non-context term disambiguate it as follows
    1. set the AWR_Max = 0 and sense = null
    2. for each candidate sense do the following:
        1. calculate its 'average of weighted relatedness' (AWR) with all the context terms
        2. if the AWR > AWR_Max, then set AWR_Max = AWR and set sense = s, else continue to next sense of the phrase
    3. return sense (the sense with greatest relatedness value)

This algorithm is the same for all techniques that use semantic relatedness to disambiguate senses of the phrases. the different between them is in the way of calculating the semantic relatedness for each technique.

2.1. Computing the context term weight

Context terms are not same some of the are more representative than other. Accordingly, it is unfair to make them contribute using the same power of effect. As a result, to maintain the degree of consistency with the central thread of text, a weight is calculated for each context term.

The weight is calculated by dividing the sum of the semantic relatedness with other context terms on the number of context terms - 1.

$$contextTermWeight(ct) = \frac{\left(\sum_{i=0}^{|contextTerm|} semanticRelatedness(ct, contextTerm(i))\right) - 1}{|contextTerm| - 1}$$

2.2. Calculating the 'Average of Weighted Relatedness' for a Prospective sense

To decide which sense is the right sense to choose, the system calculate the so-called "Average of Weighted Relatedness" for each sense of the phrase and then choose the sense with the greatest value. see the algorithm.

This average is calculated by aggregating the product of the semantic relatedness between the sense and the context term and weight of this context term for each context term. As stated by the following formula.

$$AverageOfWeightedRelatedness(S) = \frac{\sum_{i=0}^{|contextTerm|} contextTermWeight(i) * semanticRelatedness(S, contextTerm(i))}{|contextTerm|}$$

Where S is a sense of several senses for the phrase.

There are three other techniques used for disambiguation depends on the semantic relatedness between the candidate concept and the surrounding context terms. A general theme is to choose the concept with highest average of semantic relatedness with context terms. Since the context terms are not the same in their representation of the context of a document, the semantic relatedness of the context terms are weighted. The weight expresses the importance of the context term to the document by averaging the semantic relatedness between the desired context term and all other context terms; for more details and examples about weighting the semantic relatedness with context terms please refer to (Milne and Witten, 2008b).

The later three techniques have the same calculation theme. However, they differ on the way of calculating the semantic relatedness. The first depends on the in-links counts, the second depends on out-links count and the third depends on the average between both the first and the second ways; review (Milne and Witten, 2008a).

1. ^ Milne, D. and Witten, I.H. (2008a) An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proc. of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08), Chicago, I.L.
2. ^ Milne, D. and Witten, I.H. (2008b) Learning to link with Wikipedia. In Proc. of the ACM Conference on Information and Knowledge Management (CIKM’08), Napa Valley, California.
3. ^ Cilibrasi, R.L. and Vitanyi, P.M.B. (2007) The Google Similarity Distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370-383.

Table of Contents

Before phrase detection and phrase sense disambiguation tasks could be supported and user profile can be built, the data and statistics that used in both tasks should be elicited and prepared from Arabic Wikipedia dump. The aim of this process is to prepare the files used by the text processing component in order to perform its tasks.

The data and statistics that the this page shows how to extract are used in both text processing task and for building the user model. The user model consists of two structures a graph that connects between connects between concepts that user interested in and a hierarchy of categories that embraces the concepts interested in by the user. We first define what Arabic Wikipedia database dump is. Then, we demonstrate the output of the task of preparation. The output of the preparation task is a set of files that have the needed data and statistics for both text processing and modeling user. Section 3, explains how we extract these data and statistics from the database dump. The section first explain how we extract the concepts then the relatedness statistics and finally how to build the hierarchy of categories and bind the concepts with their categories.

1. The Arabic Wikipedia Dump

Wikipedia offers free copies of all available content to interested users. These databases can be used for mirroring, personal use, informal backups, offline use or database queries. All text content is multi-licensed under the Creative Commons Attribution-ShareAlike 3.0 License (CC-BY-SA) and the GNU Free Documentation License (GFDL). Images and other files are available under different terms, as detailed on their description pages. For the advice about complying with these licenses, see Wikipedia:Copyrights.

A database dump contains a record of the table structure and/or the data from a database and is usually in the form of a list of SQL statements. A database dump is most often used for backing up a database so that its contents can be restored in the event of data loss. Corrupted databases can often be recovered by analysis of the dump. Database dumps are often published by free software and free content projects, to allow reuse or forking of the database. Dumps from any Wikimedia Foundation project found here; http://download.wikimedia.org/

The Arabic Wikipedia project dump has several files; about 30 files. Most pf these files are SQL command that could be used later to generate table structure and populate them with data. Other files contains statistical data about the project. Example of such files is "pages-articles.xml.bz2" compressed file which contains current versions of article content wrapped in some XML. Another example is "page.sql.gz" compressed file which consists of a list of SQL statements that could be used to generate both table schema and table data. However, in our preparation process we don't use all the 30 files of the dump; we only use the following files in the preparation:

- pages-articles.xml.bz2
- page.sql.gz
- pagelinks.sql.gz
- redirect.sql.gz
- category.sql.gz
- categorylinks.sql.gz
- site_stats.sql.gz

All the file are significantly compressed. The first file, as mentioned above, contains current versions of article content wrapped in some XML. Each of the remaining files is representing a database table of the MediaWiki database schema(the software that runs the Wikipedia projects). See the database schema in Figure 1.

Figure 1: Database schema diagram for MediaWiki as of 1.17 (Click to Enlarge)

The "page.sql.gz" compressed file contains all the pages that exist in Wikipedia. The pages are not all articles; there are other kinds of pages such as user pages, discussion pages, Wikipedia help pages and others. Each page type is called a namespace and given an id. The namespace id of the articles in the Wikipedia is 0. The table has the following fields (only the important fields are described):

- page_id: Uniquely identifying primary key.
- page_namespace: This field contains the number of the page's namespace.
- page_title: The sanitized page title, without the title of its namespace. It is stored as text, with spaces replaced by underscores. The real title showed in articles is just this title with underscores (_) converted to spaces ( ).
- page_restrictions
- page_counter

- page_is_redirect: A value of 1 here indicates the article is a redirect; it is 0 in all other cases.
- page_is_new
- page_random
- page_touched
- page_latest
- page_len
- page_no_title_convert

The compressed file "pagelinks.sql.gz" contains all internal links in Wikipedia. Each entry contains the source page's ID; "pl_from" field, and the namespace (number); "pl_namespace" field, and article name (in text); "pl_to" field, that is being linked to within that source page. There may be many instances of the source page's ID, as many as the internal links within it, but there can be only one entry per internal link for any page ID. Note that the target page may or may not exist, and due to renames and deletions may refer to different page records as time goes by.

In Wikipedia project, A redirect page is a page that automatically redirects the reader's browser to a specified target page. Redirects are used to help users locate information and keep wikis organized, so that multiple names, abbreviations, misspellings, or related topics can all point to the same page. The "redirect.sql.gz" compressed file contains for each page that is currently a redirect. The table generated from this file contains the fields "rd_from" which the id of the source page, "rd_namespace" which is the target namespace number, and "rd_title" which is the target page title without namespace. Note that the target page may or may not exist.

All the categories of Wikipedia project is contained in the compressed file "category.sql.gz". The table produced from the file contains the following fields:

- cat_id: Uniquely identifying primary key.
- cat_title: Name of the category, in the same form as page_title (with underscores).
- cat_pages: Number of pages in the category
- cat_subcats: Number of sub-categories in the category
- cat_files
- cat_hidden

The table that generated from the compressed file "categorylinks.sql.gz" links the pages to their categories. The table also connect sub-categories to their categories. The table has the following fields:

cl_from: Stores the page_id of the article where the link was placed.
cl_to: Stores the name (excluding namespace prefix) of the desired category. Spaces are replaced by underscores (_)
cl_sortkey
cl_timestamp
cl_sortkey_prefix
cl_collation
cl_type: identify the type of page (page, subcategory, or a file) associated with the category; takes the values 'page', 'subcat' or 'file'.

Finally the file "site_stats.sql.gz" contains some statistics about the dump itself such as number of articles, number of views, number of edits, number of page total pages number of users, total pages number of admins, total pages number of images, and finally total pages number of active users. In our preparing process we only need the number of articles.

2. The Output Files

The process of preparing data and statistics from Wikipedia database dump results in the following files as its output. These files are used directly by the text processing component. The files attached to each link is just a sample to realize the format and syntax of each output file:

- **concept_inlinkCount.txt;** this file maintains the concepts extracted from Wikipedia and the number of articles back-linking their corresponding articles. (The concepts were essentially articles in the Wikipedia so normally they have back-links from other Articles)
- **concept_outlinkCount.txt;** this file contains the concepts and number of hyperlinks in their corresponding article in Wikipedia.
- **concept_inConceptList.txt;** this file contains the concept ids given to their corresponding Wikipedia articles and the list of concepts corresponding to the list of articles connecting to these articles.
- **concept_outConceptList.txt;** this file contains concept ids given to the articles and the live wikilinks of the corresponding article
- **term_conceptsList.txt;** this file contains all the terms of the Wikipedia and their potential concepts.
- **concepts.xml;** this file contains all the concept ids and their alternative synonyms.

Two sets of files

- The first set consists of the concepts XML file and terms file; used in the phrase detection process.
- The second set of files are the ones used in the calculation of the phrase sense disambiguation process.

3. Preparing Process

This rest of this page illustrate the algorithm of eliciting and preparing these data and statistics from Wikipedia dump database files. This Article page will show you how the mentioned files above is prepared.

3.1. Eliciting Concepts form Wikipedia

The idea behind extracting concepts from Wikipedia is based on the fact that each *Wikipedia article* is usually describes a single topic or entity. The article titles in Wikipedia are unique. Here, the Wikipedia's articles are used as concepts; each article in the Arabic Wikipedia project is corresponding to one concept that represents it.
Each *Wikipedia article* is usually describes a single topic or entity. The article titles in Wikipedia are unique. Here, the Wikipedia's articles are used as concepts; each article in the Arabic Wikipedia project is corresponding to one concept that represents it. Each concept takes an identifier, **concept id**, which is used later in the text processing task. Another kind of pages in Wikipedia project is called *redirect pages*. These redirect pages represent other names (synonyms) that the article could take; each redirect page represent another different name of the article. Each single article page could have many redirect pages and each redirect page point to only one article page. The redirect pages have been used to form part of the synonyms of the concepts. Throughout article text, the text may mention many other articles' names throughout the text. And an article could be referred to by many articles and since that the context that may mentioned in could vary from one article to another, the article's name could take different morphological forms and/or have different affixes. These forms are considered the rest of the synonyms to represent the concepts.

After uncompressing the files, the first step is to prepare the files by extracting the useful data (from our system point of view) from these files which has the form of SQL commands. The first file to prepare is the "page.sql.gz". The data is usually extracted form the "INSERT INTO" SQL statement (this is true also for all other files in the form of SQL commands such as pagelinks, redirect, category, categorylinks, and sitestats). The preparation step involves removing all the commands of the SQL and keeping only the data.

After preparing the data to be used, the system creates the concepts (senses). The creation of the concepts first involves removing namespaces other than articles namespace which has the id of 0. The namespace 0 does not only has the article pages but also has redirect pages and disambiguation pages (these are pages that help user disambiguate his term and direct him to the right sense of his term, during search on Wikipedia). Therefore, the system removes the disambiguation pages as well since they don't represent any entity or idea; they merely used for clearing the meaning of user query. The redirect pages are removed too for the same reason, so the remaining pages are just the article pages.

Since redirect page used to provides multiple names, abbreviations, acronyms,misspellings, and synonyms to be pointing to the same article, we uses this redirect pages to be the various synonyms for the concept. The redirect pages are exist in the "page.sql.gz" file and the links between the redirect pages and Article exist in the "redirect.sql.gz". Before useing them directly as synonyms, the file (redirect.sql) needs more preparation. For example, there are some redirect pages are pointing to other redirect pages, also there are some redirect pages that pointing to themselves and Some redirect pages redirect to pages belongs to other namespaces than 0.

The preparation steps of the redirect pages starts with removing all redirects that pointing to pages of namespaces other than namespace 0. Then the system removes redirect pages that pointing to themselves. For redirect pages that pointing to other redirect pages, we follow these links, if they lead us to an article we keep them all by replacing the target of each link by the article name in the page file; else (lead us to page not in namespace 0 for example) we remove them.

Next, after preparing the redirect links to a page, we consider the names of the redirect pages as synonyms for the article names they point to.

Sometimes when you editing an article in Wikipedia and you want to refer to another article, you want to refer to them by another name or by the same name but more suitable to the context of sentence you are writing. For example, the article named "مصر" -Egypt in Arabic- in the Arabic Wikipedia has the phrase the to pointing paragraph first the in (articles Wikipedia between link inner as) wikilink a as "للبحر الأحمر" article "البحر الأحمر". The context in the article "مصر" has forced the article name "البحر الأحمر" to changed to by is one first the ;Wikipedia in that do to ways Two .sentence the of context the suit to "للبحر الأحمر" creating a redirect page as we mentioned above, and the second way is to just by adding the pipe "|" divider followed by the alternative name in order to link to an article, but display some other text for the link to the reader.

We have taken the advantage of such properties to gather more morphological forms of the articles' names. This property provides us with different morphological forms of articles'names, more synonyms, acronyms, and more different forms agglutinated with prefixes and suffixes. Accordingly, we have parsed the "pages-articles.xml" file to collect theses forms and consider them as second set of synonyms.

After collecting the synonyms using the above two methods, we normalize all the synonyms using the unified normalization technique . The unified normalization technique removes all short vowels, Kashida, punctuation and non-letters. It replaces the TAA MARBOUTA "ة" with HAA "ه" and ALEF MAKSOURA as stored are synonyms their and concepts the ,Finally ."ا" with "أ,آ,إ" replaces and ,"ي" YAA with "ى" XML file named "concepts.xml" (as one of the output file).

3.2. Extracting the Relatedness Statistics between Concepts

The analysis of Wikipedia link structure starts by building a table that show the links between two concepts. This table is built from the file "pagelinks.sql" (the file that hold the links between all articles). First, we remove all the links that have their targets not belong to namespace 0. Then we remove the targets that have their page title not exist as synonyms. After that we replace the page titles of targets by their page ids. Finally, we replace the page ids of both sources and targets of links by the concept ids from the "concepts.xml" file.

After replacing the page ids of both sources and targets of links by the concept ids, the remove any duplicates of links. Then, we assure that each concept has a link to itself. For each distinct target concept of a link in the concepts' links table, we generate a list of its source concepts and save two output files; the "concept_inConceptList.txt" file which contains all the distinct concepts that appear as target concepts in concept links table each with its source concepts, and the second file is "concept_inlinkCount.txt" which contains each concept appears as a target concept in the concept links table associated with the number of times it appears as target.

The files "concept_outConceptList.txt" and "concept_outlinkCount.txt" are similar to the previous two files "concept_inConceptList.txt" and "concept_inlinkCount.txt", but instead considering the target concepts we consider the source concepts. For each concept appears as a source side of the links in concept links table, consider all the concepts appear as target for them. The file "concept_outConceptList.txt" consists of all distinct source concepts with their list of all concepts appear as targets, and the file "concept_outlinkCount.txt" consists of all distinct source concepts with their number of concepts appear as targets for each.

After generating the statistical data of the internal link structure, that used later in text processing component for disambiguation task, build the create the inverse file of the "concept.xml" file. The "concept.xml" file consists of concept ids each with its different synonyms. However, while phrase detection process we need to determine the senses or concepts for each detected phrase. Accordingly, we built the "term_conceptsList.txt" file which is each phrase with its list of possible senses found in Wikipedia.

3.3. Building User Hierarchy of Categories

The Wikipedia project allows each page (whether it is article or not) to has one or more a categories. Each categories could embrace one or more pages. Some categories may exist for organization reasons (i.e. this make some categories don't have pages). The categories of Wikipedia projects form an acyclic directed graph. Each categories could have one or more super categories (or parent), and may has one or more sub-categories. We use the the file "category.sql" to extract the categories. The file "categorylinks.sql" is used to link concepts to their categories.

Since the graph is "acyclic directed graph", We use the term hierarchy for sake of simplicity.
Since we need only categories that classifying article pages and their super categories until the root of the hierarchy. We need to eliminate other categories that used for other pages. Accordingly, since we know the pages that are used as concepts, we maintain only the categories that classify these pages only. Using the "categorylinks.sql" file we keep only those links that their "cl_from" attributes belong to the set of page ids of articles that used as concepts. We extract the categories names from the field "cl_to" of these links. To determine the parent categories of categories in hand a new iteration is conducted to determine their parent using the page id of these categories. We conduct more iterations until there is no more categories found.

Here are the steps in a compact form:

- prepare_wiki_files();
    - prepare_pages();
    - prepare_pagelinks();
    - prepare_redirects();
    - prepare_categorylinks();
- create_concepts();
    - remove_ns_other_than_ns0();
    - remove_disambiguation_pages();

    - remove_redirect_pages();
    - prepares_in_redirects();
    - add_synonyms_to_concepts_file(out pageid_syns);
        - add_synonyms_from_redirects(ref syn_pageid_dic);
        - add_synonyms_from_article(ref syn_pageid_dic);
    - Normalize synonyms
    - create_xml_file(ref pageid_syns);
- create_links_statistics();
    - Building Concept Concept Link Table
    - remove duplicate same concept link
    - create cc inlink count
    - create concept inconcept list index
    - create cc outlink count
    - create concept outconcept list index
- generate_term_concepts_list_index(concepts_path);

The **query customizer** component is responsible for individualize query submitted to search engines monitored by the ArabAgent system. The query customizer is another component used for personalizing user access to the web. The other component is the ranking and filtering component.

Personalizing the user experience using query processing techniques has several advantages over other techniques that handle (rank or tag) search results. Although, techniques the deal merely with search result set try to improve the effectiveness of the system by improving its precision at particular rank, it totally ignores the recall measure. The reason behind that is that the system is confined only to the set of result return by a certain query. This could declines the overall system performance in case where user submitted bad query keywords (the result hits contains no relevant data at all). Accordingly, some systems opt to personalize the search by modifying the user original query to avoid these drawbacks. Furthermore, it is difficult to ranking again all the result of the search engines. The user can't tolerate the relatively long time the system takes to rank all the results.

In the other hand, one of the main disadvantages of the keyword search, which has been adopted by the majority of search engines such Google, is that the result is very sensitive for keywords used in web search. Different keywords yield different result for most search engines; even if they have the same meaning. Accordingly, modifying user query is tricky and difficult. You cannot augment new keywords to the user original query unless you are hundred percent sure they are going to improve the result. The keywords could take the query to another direction causing the result to decline in terms of both precision and recall.

Therefore, in ArabAgent, we introduced a novel technique to modify the user query. Instead of adding just the synonyms for the query, a technique to reduce the ambiguity of the query is used. This technique is suitable only for the informational kind of queries [1] -queries that cover a broad topic (e.g., neural networks, ancient Greece) for which there may be thousands of relevant results and usually used for learning about the thing you are searching for. Using this technique with other kinds of queries such as transactional and navigational queries would harm them.

Figure 1: Query Customizer Component

Figure 1 depicts the sub-component of the query customizer. The technique is summarized in the following algorithm:

1. Identify just the phrases (not the concept) in the query using the phrase detection sub-component of the text processing component, then
2. Use the user model to disambiguate the phrases in query with most interesting concept to user.
3. Replace each phrase in the query by another phrase for the same concept, but with minimum senses or concepts. (this could only done for informational queries, however other queries such

navigational and transactional queries is not suitable)

4. The component could (optional) limit the search of the search engine to certain websites.
5. The component could (optional) add synonyms and acronyms for each phrase in the query (this is only provided for informational queries)

---

1. ^ Andrei Broder. 2002. A taxonomy of web search. *SIGIR Forum* 36, 2 (September 2002), 3-10.

Semantic Relatedness is a technique to compute the relatedness or similarity between two concepts or senses. The semantic relatedness used as a part of the phrase sense disambiguation process in the ArabAgent system. In ArabAgent, the semantic relatedness applied using several techniques. All the techniques discussed here are taken from Milne and Witten[1] [2] . The ArabAgent computes the semantic relatedness using *"most closely related pair using in-links*", and *"most closely related pair using out-links"* techniques. The ArabAgent could also combine between these two techniques to leverage both in-links and out-links of the Wikipedia articles.

Table of Contents

1. Preparing Wikipedia Data

Semantic relatedness techniques use the Arabic Wikipediato elicit data and statistics that are needed during the computation of semantic relatedness.

Before we go further in explaining the methods and techniques used in semantic relatedness, the data and statistics should be elicited from the Arabic Wikipedia and prepared for use. The elicitation process result in the following files which are essentially required for the task of semantic relatedness:

- **concept_inlinkCount.txt;** this file maintains the concepts extracted from Wikipedia and the number of articles back-linking their corresponding articles. (The concepts were essentially articles in the Wikipedia so normally they have back-links from other Articles)
- **concept_outlinkCount.txt;** this file contains the concepts and number of hyperlinks in their corresponding article in Wikipedia.
- **concept_inConceptList.txt;** this file contains the concept ids given to their corresponding Wikipedia articles and the list of concepts corresponding to the list of articles connecting to these articles.
- **concept_outConceptList.txt;** this file contains concept ids given to the articles and the live wikilinks of the corresponding article
- **term_conceptsList.txt;** this file contains all the terms of the Wikipedia and their potential concepts.

2. Semantic Relatedness, What does it Mean?

Since concepts are Wikipedia articles in the first place, the semantic relatedness between two concepts is essentially a task of finding relation between their two Wikipedia articles. Two techniques used to find this relation between the two articles; the first depends on the hyperlinks (called the wikilink) found within the two article of Wikipedia, or so-called *out-links*. This technique compares between the two set pf wikilinks found within both articles and as the percentage of the shared number of wikilinks increases as the relatedness of the two articles (and the two concepts in turn) increase.

The other technique to compute the semantic relatedness is similar to the one above but instead of using the wikilinks we use the back links, or so-called *in-links*. Also, here the technique compares between the in-links and as the percentage of the shared number of in-links increases as the relatedness of the two articles (and the two concepts in turn) increase.

Another difference between the technique uses out-links and the technique uses the in-links is the way to compare between the lists of the out-links and in-links of the both articles to compute their relatedness. following subsections describes this difference in detail.

3. Semantic Relatedness using Out-links

This measure is defined by the cosine similarity between the two out-links vectors of both articles that their relatedness is needed. The weight scheme is almost similar to tf-idf weight scheme. The only difference is that weight is computed as the link occurrence counts weighted by the *probability of each link occurring* instead of weighted by the probability of term occurring in the Wikipedia articles.

Calculating semantic relatedness using out-links requires the following data:

- The list of distinct links inside an article to represent out-links,
- The occurrence count of each distinct wikilink inside the article,
- count of articles that contain each link in the distinct wikilinks inside both articles, and
- The number of all Wikipedia articles, |W|.

Suppose that s and t are the source and target articles. T is the set of all articles that link to t, then the probability of the occurring of this link is computed by the following formula:

$$\omega(s \longrightarrow t) = log\left(\frac{|W|}{|T|}\right) \quad \text{if s} \in T,\ 0\ otherwise$$

The set of features of the two vectors is the union of all distinct links made from both of the two articles.

4. Semantic Relatedness using In-links

To compute the *semantic relatedness using in-links* technique, the system must maintain the following data:

- The set of in-links of each article in Wikipedia, and
- The number of all Wikipedia articles, |W|.

This technique is originally inspired by distance measure of Cilibrasi and Vitanyi [3] ; named Normalized Google Distance. Since they used Google search engine to measure the distance between two terms. Later, Milne and Witten[4] used the same measure but based on Wikipedia structure and dubbed *Wikipedia Link-based Measure* or *WLM*. The distance measure is as follows:

$$sr(a,b) = \frac{log\,(max(|A|,|B|)) - log(|A \cap B|)}{log(|W|) - log\,(min(|A|,|B|))}$$

Where a and b are the two articles of interest, A and B are the sets of all articles that link to a and b respectively.

Furthermore, semantic relatedness could be computed using both the in-links and out-links. This technique showed great performance over either using in-links or out-links alone to compute semantic relatedness [5] . The technique is computed by simply calculate the average between the two measure previous measures.

Computing the semantic relatedness is not the only way in ArabAgent system to disambiguate between senses. ArabAgent system uses *most common sense* technique as another technique for disambiguation.

1. ^ Milne, D. and Witten, I.H. (2008a) An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proc. of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08), Chicago, I.L.
2. ^ Milne, D. and Witten, I.H. (2008b) Learning to link with Wikipedia. In Proc. of the ACM Conference on Information and Knowledge Management (CIKM’08), Napa Valley, California.
3. ^ Cilibrasi, R.L. and Vitanyi, P.M.B. (2007) The Google Similarity Distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370-383.
4. ^ Milne, D. and Witten, I.H. (2008a) An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proc. of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08), Chicago, I.L.
5. ^ Milne, D. and Witten, I.H. (2008a) An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proc. of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08), Chicago, I.L.

aaaa

unified normalization technique

Table of Contents

As user provides his relevant set of documents, the user profile is updated accordingly. More importantly, as certain concepts did not get mentioned any more in the set of relevant documents, the system assumes that the user interests of these concepts decline. As a result, a time window technique of size 30 days is used to adapt the user interest attributes automatically to mimic user interest behavior. The interest attribute value is get decay over time until it reaches value zero i.e. the user has no interest in a topic anymore; although he may has a great knowledge about the topic.

1. Preparing Documents for Updating

To create the graph of to the set of the feedback documents, several pre-processing steps to documents need to be accomplished. After downloading and eliciting text form Web page, the concepts should be identified from the documents. The concept identification process isn't direct. Phrases that may compose concepts in the future are detected first. Then, a disambiguation process is needed to reveal ambiguous phrase i.e. phrases with multiple meaning. Following, the pre-process step and the process of updating user profile are explained in great detail.

2. Generating Feedback Document Graph

After identifying the concepts of each document, each document in the set is then represented as an undirected unweighted graph where just frequency of occurrence attributes values of only concepts is computed. For each document, the occurrence attribute values for the concepts are the numbers of occurrence of the concepts in the document. Each identified concept is connected with all other concepts within particular document. The system uses the graph of the documents to update the user profile.

3. Updating the Profile

The updating process begins after preparing the document that considered relevant and generating its graph. The updating process starts with updating the concepts first. In case a concept is not exist it adds a new concept to the profile. After done from the concept the system turns to the edge. In the manner, the system updates the existing edges first, if the edge is not exist in the profile, it adds a new edge with profile. The following discusses the steps of updating the profile in detail.

3.1. Updating Existing Concept

For each document in the feedback set, update the user profile with the graph of that document. Each concept in the document graph is checked for its existence in the user profile. In case the concept exists in the profile, the existing concept in the profile is check if it has been visited today. If the concept has been visited today we increase the docCount attribute by 1 and add the concept importance value of the current concept to the old value of SCISD attribute plus concept importance of the graph (the concept importance value equals the the concept occurrence frequency in the document divided be the sum of concept occurrence frequency of all concepts). In case the concept hasn't been visited today, a new day element is added for the concept with date value equals today date, docCount value equals 1 and SCISD value equals the value of concept importance.

3.2. Adding New Concept

In case the concept is not exist in the user profile, (the concept is totally new), a new concept element is add to the profile with a new day representing today where the value of date attribute equals today's date, the value of docCount attribute equals 1 and SCISD value equals the value of concept importance.

3.3. Updating Existing Edges

After adding all the concepts to the profile we add the edges of the graph into profile. Each edge in the document graph is checked for its existence in the user profile. In case the edge exists in the profile, the existing edge in the profile is check if it has been visited today. If the edge has been visited today we increase the docCount attribute of last day element (which is today) by 1 and add the minimum concept importance value of concepts participating in the edge to the value of SMP attribute in the profile. In case the edge hasn't been visited today, a new day element is added for the edge with date value equals today date, docCount value equals 1 and SMP value equals to the minimum concept importance value of concepts participating in the edge.

3.4. Adding New Edges

In case the edge is not exist in the user profile, (the concepts could exist but the edge is totally new), a new edge element is add to the profile. The attributes "to" and "from" are equal to the concepts participating in the edge. a A new day representing today is added to the edge element where the value of date attribute equals today's date, the value of docCount attribute equals 1 and SMP value equals to the minimum concept importance value of concepts participating in the edge.

Modeler and profiler component is the only component that directly manipulates the user profile. The modeler and profiler component is used to update user profile, generate the user model from the user profile, provides the user model for all other component for personalization, and save the new user profile after updating.

The data extracted from the user feedback is stored in a user profile. After the user interface provides the relevancy feedback of the user and sends it to the modeler and profiler component, the modeler and profiler component send the user relevance feedback to the text processing component to identify the concepts in the web pages. After identifying concepts, the modeler and profiler component loads the user profile into memory and updates it using the concepts. The modeler and profiler component save the profile after updating.

Although user profile stores the user data, ArabAgent system don't use the user profile directly in personalization tasks (i.e. tagging, ranking, or modifying query). The system has to generate the more convenient form of user model for personalization tasks. The user model is the system view of the user. The modeler and profiler component generates the user model and provides it to query customizer component and ranking and filtering component.

This component do the following:

1. Download the feedback document
2. Remove tags and unwanted scripts from the text of Web page
3. Identify the concepts of the text using the text processing component.

see the following figure.

Text Processing Component of ArabAgent System

Downloading Web Page

One of the main problem is to identify the encoding of document that is feed to the system. Several approaches have been used. however, the most successful one is the method that depends on machine learning to identify the encoding[1] . In ArabAgent we have used a simple but expensive method (in term of time because we download the page twice). We first download the web page encoded in utf-8 encoding. Then, we check its "charset" attribute in the "meta" tag, if exist we download again the text with new encoding, if not exist that means that the page is encoded in utf-8 and no need to download it again.

```
 private string getHTMLofWebpage(Uri uri)
        {
                string url = uri.AbsolutePath;
                WebClient client = new WebClient();


//this method always download pages encoded in windows-1256
                string html = client.DownloadString(url);
                string charSet = get_page_charset_value(html);


//the page iencoding is always utf-8 of no encoding is specified
                if (string.IsNullOrEmpty(charSet))
                {
                    client.Encoding = Encoding.UTF8;
                    html = client.DownloadString(url);
                }
                else if (Encoding.GetEncoding(charSet) != client.
Encoding)
                {
                    client.Encoding = Encoding.GetEncoding(charSet);
                    html = client.DownloadString(url);
                }

                return html;
        }

        private string get_page_charset_value(string html)
        {
            MatchCollection mc = Regex.Matches(html, @"<meta.*?>"
, RegexOptions.IgnoreCase | RegexOptions.Singleline);

            for (int i = 0; i < mc.Count; i++)
            {
                if (mc[i].Value.Contains("charset="))
                {
                    string str = mc[i].Value;

                    return str.Substring(str.IndexOf("charset=") + 8
, str.LastIndexOf("\"") - str.IndexOf("charset=") - 8);
                }
            }
            return null;
        }
```

Removing Tags and Scripts

Once the html string is available, we remove the all tags from the string as well as the Javascript string and CSS string. The keywords inside the "meta" tag are added to the text again. See the following code:

```
void processing_web_page(string html, out List<string> terms, bool
 stopword_removal)
        {

//Extracting certain Informaion from tags (such as meta and title tags
)
            html = add_meta_keywords_and_description(html);


//remove tags arbitrary no information are kept about tags (no title i
nforamtion such as information
            //that kept by <h1> tags and <p> tags.
//you may check "knoweldege extraction from web pages" topic it could
help.
            html = remove_unwanted_tags(html);

            List<string> tokens = new List<string>();
            tokens.AddRange(Tokenize(html));

            for (int i = 0; i < tokens.Count; i++)
                tokens[i] = text_processing(tokens[i]
, stopword_removal);

            terms = tokens;
        }

        string add_meta_keywords_and_description(string html)
        {
            MatchCollection mc = Regex.Matches(html, @"<meta.*?>"
, RegexOptions.IgnoreCase | RegexOptions.Singleline);
            char[] sep = { '\"' };
            //char c = '\u0022';
            for (int i = 0; i < mc.Count; i++)
            {
                if (mc[i].Value.Contains("name=\"keywords\""))
                {
                    string str = mc[i].Value;
                    Match m = Regex.Match(str, "content=\"[^\"]*\"");
                    if (!string.IsNullOrEmpty(m.Value))
                        html = html.Replace(str, " " + m.Value.Split(
sep)[1] + " ");

//return str.Substring(str.IndexOf("charset=") + 8, str.LastIndexOf("\
"") - str.IndexOf("charset=") - 8);
                }
                else if (mc[i].Value.Contains("name=\"description\""))
                {
                    string str = mc[i].Value;
                    Match m = Regex.Match(str, "content=\"[^\"]*\"");
                    if (!string.IsNullOrEmpty(m.Value))
                        html = html.Replace(str, " " + m.Value.Split(
sep)[1] + " ");
                }
            }
            return html;
        }

        string remove_unwanted_tags(string html)
        {

//removing script tags including the text between start and end tags
            html = Regex.Replace(html, @"<script.*?</script>", " "
, RegexOptions.IgnoreCase | RegexOptions.Singleline);


//removing tyle tags including the text between start and end tags
            html = Regex.Replace(html, @"<style.*?</style>", " "
, RegexOptions.IgnoreCase | RegexOptions.Singleline);


//removing <noscript> including the text between start and end tags
            html = Regex.Replace(html, @"<noscript>.*?</noscript>",
" ", RegexOptions.IgnoreCase | RegexOptions.Singleline);


//code = Regex.Replace(code, @"<!--.*?-->", " ", RegexOptions.IgnoreCa
se | RegexOptions.Singleline);

            html = Regex.Replace(html, "<[^<>]+>", " ", RegexOptions.
IgnoreCase | RegexOptions.Singleline);

            //removing character entities
            html = Regex.Replace(html, @"&\S+?;", " ", RegexOptions.
IgnoreCase | RegexOptions.Singleline);
            return html;
        }
```

After the preparing the text of the web page, the text is sent to text processing component to identify its concepts for later either update user profile or personalize the user experience.

1. ^ Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, ISBN: 0521865719

The **Web Wrapper** of ArabAgent system is the component that deals directly with the Web matters such as submitting queries for the search engine, receiving result from the search engine, extracting information from results, extracting news article from web pages etc.

After Query customizer component modify the user query, it sends the query to the web wrapper. The web wrapper adapts the query to suit the search engine sending to. The web wrapper receives the result from the search engine and sends it to the user interface to show it. In case a new news article available in the news website the web wrapper send it to ranking and filtering component before send it to user email.

Although, most of the search engines accept free text search queries, each search engine has its own set of operators. For example, in addition to the well-known widely used operators such as OR, AND, and NOT, there are a lot of specific operators to Google search engine, for example, such as allinanchor:, allintext:, allintitle:, allinurl:, cache:, define:, filetype:, id:, inanchor:, info:, intext:, intitle:, inurl:, link:, phonebook:, related:, site:, and more (for more information about Google operators check this Website). The Web wrapper forms each query to suit the search engine submitted to.

In addition to search engines' operators, there are search result URL parameters. The web wrapper extracts the values of these parameters or changes them in order to obtain the results of a search. These parameters differ from search engine to another search engine. Example of such parameters is Google search URL parameters such as "q=", "as_epq=", "as_oq=", "as_eq=".

The continuous growth of the free Wikipedia project makes it a good source of a controlled vocabulary. Due to collaboration work of volunteers, the Wikipedia grows constantly and rabidly. This gives it more advantage than other resources which is fixed in size such as Arabic WordNet. The Wikipedia produces a database dump every 15 days this makes the Wikipedia reflects the reality and makes it up-to-date. For example, you could not find a latest Arabic film or new football player in knowledge resources other than Wikipedia.

Another reason is that, while Arabic WordNet version 2 consists of 9228 synsets (6252 nominal, 2260 verbal, 606 adjectival, and 106 adverbial) and about 18,957 Arabic expressions (Rodriguez, 2008), the Arabic Wikipedia project dump, which has been pulled on 31 May 2010, contains about 127,273 good articles and 709,234 total pages; both synsets and articles are used to represent the concepts. In addition, one of the basic criteria for selecting synsets to be covered in Arabic WordNet is the generality which means it is not thorough enough (Black et al, 2006).

The majority of Wikipedia pages have been manually assigned to one or multiple categories. The Arabic Wikipedia Project consists of about 27,000 categories. They represented as acyclic directed graph. The top most category is dubbed “fundamental” category which is divided into 16 subcategories; art, science, sports, …, etc. Each category is represented by the

As Wikipedia considered being a good resource, there are some drawbacks which concern more the Arabic Wikipedia. Most of the drawbacks are because Arabic Wikipedia is still under development. For example, there could be lacking in some forms or affixes for some phrases or terms since they haven’t been yet in a context forcing them come in specific morphological forms or with certain affixes. Also, there exist some wrong internal links in the Arabic Wikipedia. Fortunately, although these drawbacks have an impact on using Wikipedia as a source of language processing, these drawbacks disappear as Wikipedia continuously evolves. [Immaturity of the Arabic Wikipedia project]

In order to extract our knowledge, we use the XML dump of Arabic Wikipedia. It is approximately 887 Megabytes large and stores the articles in the original wiki markup language. The database dump of the Arabic Wikipedia project has been used as a resource to help retrieving documents based on concepts. The database dump of Arabic Wikipedia project is pulled around twice a month. The dump contains an SQL commands that run against different database management system to create the schema and its extension of the database. The main tables that have been used to extract the knowledge needed to our system are pages table, pagelinks table, redirects table, category table, categorylinks table, and articles XML file.