Making Topic-Specific Report and Multimodal Presentation Automatically by Mining the Web Resources

Shaikh Mostafa Al Masum, Mitsuru Ishizuka
Department of Information and Communication Engineering, University of Tokyo, Japan
mostafa_masum@ieee.org, ishizuka@i.u-tokyo.ac.jp

Abstract

Due to availability and accessibility of enormous internet-based resources and dynamic nature of web pages, the task of information retrieval is becoming more challenging and gradually tricky. This paper describes about agent based autonomous system, Automatic Report to Presentation (ARP), with the notion of autonomous information service emerging as the result of integration among natural language processing, web intelligence, and character-based agent interaction. The system, ARP, automatically builds a report on a topic or search phrase(s) given by a user by fetching a set of web-pages; and then parsing; summarizing, affect-sensing and correlating information extracted from those. The system also makes a concise presentation automatically and, a group of character based software-agents autonomously present the topic in a story-telling manner employing text-to-speech engine with accompanied content-rich slides, different gestures and affects.

1. Introduction

Internet is the biggest online multi-disciplinary information repository in the world. Due to availability and accessibility of enormous internet-resources and dynamic nature of web pages, the task of information retrieval is becoming more challenging and tricky. User interested in a certain topic can utilize numerous sources of information resources of various nature, content and characteristics. Hence the idea of automatically creating report(s) on a particular topic or query string given by a user and building automatic multimodal presentation(s) utilizing online resources might be thought as an addition towards autonomous information service as well as a special kind of web-search service providing content-rich report and story-like presentation for grasping an idea about a topic quickly and with fun. Such system will not suffice serious web-searching but may evolve as a service to deliver story-like result(s) for a web-search query. Specially, children and naïve users of computers might find such a system useful and as a source of fun. The research is encircling different techniques like html parsing web pages; extraction and summarization; question answering system; information retrieval; sensing affective information from text for multimodal presentation and markup language to dynamically script avatar based software-agents with affects and gestures to incorporate anthropomorphism for improved human-computer interaction.

Web pages are often very "noisy" in the sense that they might contain many unrelated information. So, many unrelated text segments may be identified by an HTML-parser. There are different techniques and heuristics for parsing HTML pages as discussed in [10][13][19], but the limitation of those parsers for our purpose is, as web pages may emphasize phrases or long text segments unrelated to the key information, further parsing is required to extract the important text and concept from a document. Hence for the system, *ARP*, we have implemented several heuristics to parse HTML coded Web-pages and extract data in the form of tuples of salient heading(s) and associated text-chunk.

Web searching, extracting and finally summarizing useful information are active research areas since last decade. In brief, the techniques include Keyword-based search (e.g. [4][10]), Web queries, Wrapper Induction for Information, Effective Resource Discovery, User Preference-based search and Content or Context based summarization. For the system keyword-based searching using web-search engines, e.g.[9][24], are taken as the initial step to collect the links of potential information relevant to searching topic. Web query languages allow the user to retrieve data from web pages by using extended database query languages. This is not required for our system. Wrapper Induction for Information approaches (e.g., [4]) is not also suitable because a wrapper is a procedure for extracting tuple from a particular information source. Hence, they are not designed for finding significant concepts and exploratory texts associated with the different concepts of user-specified topics. Effective (Web) Resource Discovery aims to find Web-pages relevant to users' requests or interests (e.g., [19]). This approach uses techniques such as link analysis, link topologies, and text classification methods to find relevant pages. In the user preference approach, information is presented to the users according to their preference specifications and this is not helpful for our problem too. Content based summary utilizes textual content of the web documents in question. The disadvantage of this method becomes evident when a particular web page contains little textual content and relies mostly on visual language communication. Context-based method, e.g., [1][6], are making use of the hypertext structure of the web, exploit the paragraphs or

other text units that are close to the links pointing to the particular document being used to create the content. For our approach we used the mixed approach of content and context based retrieval. We first searched the topic in *Wikipedia*[23] and if the topic is ambiguous and found in the *Wikipedia* source, we extract several context(s) and relevant content(s) of the topic and then subsequently other web-search engines are invoked. We also utilize ConceptNet 2.0 [14] for sensing affective information and topic gisting from the chucks of text considered for presentation slides. In the system we employed ConceptNet to utilize two functions of ConceptNet, *GuessMood()* and *Summarize()*. *GuessMood()* returns a tuple of six emotions (happy, sad, angry, fearful, disgusted, and surprise) with their respective percentage value indicating the mood of the input text and *Summarize()* function is useful for getting a summary or gist of the topic from a chunk of text. In future we plan to employ other functions of ConceptNet for further analysis of Context.

Related work to ours is question-answering (e.g., [8], [12],[11]). A question-answering system is used to answer user questions by consulting a repository of documents. [8] utilizes the snippet returned from a search engine to help find answers to a question. We have incorporated some of the heuristics from question-answering research to finding such informative pages and also utilize some of the concepts of [12] which explained about mining topic-specific concepts and definitions collected from web pages. However, the total task is different in terms of building the outline of a report and presentation dynamically and associating summarized text-chunks to the related heading and finally presenting the topic gist by some character-agents with accompanying affect and gesture to mimic human-like behavior.

Some of the prominent applications, e.g. embodied characters are now used as virtual tutors in interactive learning environments [22], virtual sales agents [2] and presenters [3] in a pre-scripted manner. So, a scripting language, Multimodal Presentation Markup Language (MPML)[18][21] has been developed to script and generate human-like behavior for the character-agents. We have extended the *"Auto-Presentation"* system discussed in [20] in terms of retrieving information for ambiguous topic and making multiple presentations; building content-rich report; adding affective qualities to the character-agents and encoding into MPML script dynamically; improving topic gisting and increasing user interaction.

2. Overview of the System

The objective of the system is to provide fun and knowledge especially to the kids, students and naïve computer users by building a presentation and a content-rich document of an input topic. In the system, we do not require extensive linguistic analysis or machine learning than shallow language processing, but we employ conventional search engines, web encyclopedia and exploit the structure of the web pages to identify candidate phrases for information retrieval. To create the contents web encyclopedia (e.g. [23]) and multiple but unique web pages returned by search engines ([9][24]) are consulted. The features added or improved to the previous system discussed in [20] are:

- Multiple report and presentation for ambiguous topic.
- Improved Web Search, Filter, Extract, Rank, Summarize and Report Outline.
- Mood / Affect incorporation to the character-agents

The system, *ARP*, is consisting of multiple agents performing specific tasks. Figure 1 shows the architecture of the system in terms of agent interaction. The names of the agents are self-explanatory. Briefly, the Query Analyzer (QA) agent does key-phrase extraction for topic identification from input query and disambiguation processing for the topic. The Web Crawler (WC) and Web-page Extractor (WE) agents interact with each other to fetch web pages based on search-keys employing search-engines ([9][24]). The WE agent also reads each web-page and considers potential HTML-tags to produce simple text-files by extracting data from those. The Report Builder (RB) agent creates summarized text-chunks from the extracted text-files and prepares a report with salient headings. The Presentation Builder (PB) agent creates different slides from the report, retrieves images using search engine to add them to the slides, does mood analysis using ConceptNet and creates MPML script accordingly. Finally the Presentation Avatar (PA) agent creates JavaScript from the MPML script and performs the presentation using several character-agents (e.g. Microsoft-Agent character Genie, Marlin etc [17]).

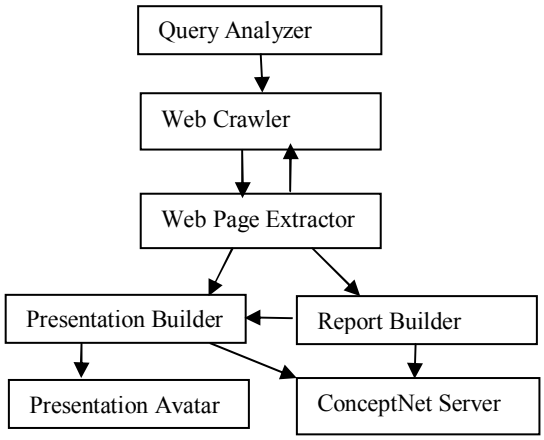

Figure 1. Multi-Agent Architecture of the System

3. Implementation of *ARP*

The ARP system consists of six software-agents working in a pipelining manner. ConceptNet Server is the implementation of ConceptNet 2.0 [14]. In the following subsections each agent and its functionalities are been discussed with necessary algorithms.

3.1. Query Analyzer

The query analyzer first validates linguistically if a proper question has been asked. For a valid question, *Q*, using a shallow language parser, it extracts the Topic, *T* from *Q*. *T* is depicted as a concatenated string of words (W_i). $T = W_1 + W_2 + ... + W_C$

Where, W_i indicates words

C = count of words in *T* and at present C ≤ 10 and

$W_i \neq$ adverb, pronoun, preposition, determiner, aux-verb

For example if someone types *"What do you know about the arsenic poisoning in the asian countries?"*; Query analyzer signals it as a valid question and outputs Topic, *T* as: *"arsenic poisoning asian countries"*. But sometimes *T* may have ambiguous meaning or context. For example if someone asks, *"Can you tell me about Jaguar? or What is a virus?",* the topic *"Jaguar"* or *"virus"* may have different contextual senses. In order to deal with this issue query analyzer first searches for *T* in Wikipedia, the online encyclopedia, and does a data-mining in the web-page returned by Wikipedia for the key "*For other uses"* which essentially gives the web-link to decipher the possible contextual senses for topic *T*. If such data-mining is successful, *T* is set ambiguous and the agent further does data-mining in the web-page returned by that particular disambiguation link of Wikipedia for the possible senses. In the case of multiple contexts we consider 5 maximum senses and hence the system can generate maximum of 5 automatic reports and presentations of the topic. For example *"Jaguar"* returns 16 such senses and we considered the first 5 senses. So for an ambiguous *T* we get a set of senses which we name as, Topic-Sense, *TS* and represented as,

$TS = \{TS_1, TS_2, ..., TS_N\}$

N= Number of Senses found for the Topic, N ≤ 5

Query Analyzer then makes a set of search topic for the web-crawler agent and hence, the set of Search-Topic, $ST = \{ST_1, ST_2, ..., ST_N\}$ where each Search-Topic, ST_i also contain the main Topic, *T*.

Hence a Search-Topic, $ST_i = T + TS_i$; 1≤ i ≤ N;

3.2. Web Crawler

The Web Crawler agent employs web search engines, i.e. [9][24], to fetch a set of relevant web-pages based on each search topic. For each search-topic, ST_i, in set *ST* the agent does the following. Consider a Search-key, $K = ST_i$ and then prepares a search string to search for *K* using the online encyclopedia, i.e. [23]. The search-link is represented by, $W_i = \{W_iL_1\}$;

Web crawler then invokes Google search engine and fetches the search-result page for *K* and receives a set of links from Web-page extractor agent. The set of links for *K* returned by [9] is,

$G1_i = \{G1L_1, G1L_2, ...G1L_n\}$ and we limit n≤10.

In the same way the agent invokes Google again to receive the set of links for different prefix added search-key, *K* as indicated below.

$G2_i = \{G2L_1, G2L_2, ...G2L_n\}$ n≤10

$G3_i = \{G3L_1, G3L_2, ...G3L_n\}$ n≤10

$G4_i = \{G4L_1, G4L_2, ...G4L_n\}$ n≤10

The above sets of links are obtained from Google for *K*, where *K*= "Who What When" + ST_i ; *K*= "History of" + ST_i and *K*= "About" + ST_i respectively. The search-string used to invoke Google search engine as an http request is *http://www.google.com/search?num=10&hl=en&q="*+K +*"&lr=lang_en&as_ft=i*. The Web-page extractor agent can also return the links of PDF files that could be viewed as HTML pages as suggested by Google search result.

The search string used to invoke yahoo search engine is *http://search.yahoo.com/search?p="*+K and yahoo search engine gives the following sets of links as indicated;

$Y1_i = \{Y1L_1, Y1L_2, ...Y1L_n\}$ n≤10

$Y2_i = \{Y2L_1, Y2L_2, ...Y2L_n\}$ n≤10

The above sets of links are obtained from [24] for *K*, where *K* = ST_i and *K*= "About" + ST_i respectively.

The agent then makes the set of unique links obtained from [9] and [24] and finally makes a set of unique links, U_i, which serves as the sources of knowledge for the search-topic, ST_i .

$G_i = G1_i \cup G2_i \cup G3_i \cup G4_i$ //all links of Google

$Y_i = Y1_i \cup Y2_i$ //all links of Yahoo

URL-set, $U_i = W_i \cup G_i \cup Y_i$ //The list of unique URLs

The web crawler agent finds definition of the topic using the following search string. For example, http://www.google.com/search?num=10&hl=en&q=define:"+ *K*; returns the definition of *K*. If the agent fails to find any definition for a given Topic, ST_i, it forms sub-topics by taking the portion of the search topic. Algorithm to find a definition using [9] is given below:

```
Begin
   Search-Topic, ST = ST_i
   W_i is the list of words in ST
   Search-Key, SK=NULL
   Set j = C ; where is the number of words in ST
 Set Definition, d=NULL
 While (d=NULL)
   Begin
     Search-key, SK_j = sum_{i=1}^{j} W_i where 1 ≤ j ≤ C
     d = getDefinitionFromGoogleFor(SK_j)
     If (d = NULL)
       then j= j-1
     Else exit the loop
      Loop While
   End
End
```

3.3. Web-page Extractor

After receiving the URL-set, U_i from the web-crawler agent, it retrieves a set of web-pages, WP_i represented as, $WP_i = \{P_1, P_2, ..., P_M\}$

Where, M= number of web-pages for the search-topic ST_i

The agent then reads the content of each page, P_i, and retrieves text between <body> and </body> tag. While reading the content from the web-pages the following heuristics are followed.

a. Emphasizing tags like <h1>, <h2>, <h3>, <h4>, <b>, , <big>, <i>, , <u> are considered for heading or salient text.

b. Ignore the heading text if longer than 125 characters.

c. Omit the texts inside <script> and <style> tags.

d. Collect the text chunk which appears between the other types of tags not mentioned above.

e. Ignore the text that contains an URL or an email address.

f. Ignore the text-chunk which is too long (e.g. more than 600 words) or too short (e.g. less than 10 words).

g. We assume that the heading text represents as the title for the text-chunk(s) found immediately after the heading(s). Several headings may be retrieved in a row and then we need to summarize the headings too.

h. Some unwanted markup text and character (e.g.   etc.) may be present in the retrieved text, so we stripped out all the text between '<' and '>' markup character.

The output from each Extracted Page (EP_i) is a list of tuples of potential headings and text-chunk, which can be represented as following (i.e. l many heading, p many text-chunks),

Heading-Text Tuple, $HT_k = [[h_1, h_2, ...h_l], [T_1, T_2, ...T_p]]$

So, the list of Extracted Page, $EP_i = \{HT_1, HT_2, ... HT_R\}$

Thus a page may result R ($0 \leq R \leq 50$) many HT tuples. Finally the agent produces a list of Extracted Pages *(EPL)* which is further analyzed to prepare automatic content. The output of the agent can be represented as following, (M many extracted pages, each having R many HT tuples) where the value of R may not be equal for each page.

$EPL = \{EP_1, EP_2, ... EP_M\}$

3.4. Report Builder

The Report Builder agent employs ConceptNet 2.0 as a server application to receive summarized text for a chunk of input text using *Gist()* function. The pre-processing algorithm for report building is given below:

Begin
  For each item, EP_i in EPL
    For each tuple, HT_k in EP_i
      Begin
        $H_k = [h_1, h_2, ...h_p]$; get list of Headings
        $T_k = [T_1, T_2, ...T_q]$; get list of text-chunks
        $h_k = Gist(H_k)$; get the summary of title(s)
        $t_k = Gist(T_k)$; get the summary of text(s)
        $HTG_k = [h_k, t_k]$; tuple of Heading-Text Gist
        Add HTG_k to Page-level Heading-Text list, PHT_i
      End
End

Since we get the following list called Page-level Head-Text list, from each EP_i,

$PHT_i = \{HTG_1 HTG_2, ..., HTG_R\}$

For a search-topic, ST_i, having M many documents we get a set of documents, D_i, containing M numbers of PHT,

$D_i = \{PHT_1, PHT_2, ... PHT_M\}$

The report builder agent creates a report, R_i, using the contents of D_i according to the following algorithm:

Begin
  Set Report, $R_i = PHT_1$
  For j=2 to M
    For each tuple, HTG in PHT_j
      Score= *GetCloseness(HTG, R_i)*
      If Score <40 Then
        $R_i = R_i \cup \{HTG\}$ //add the content to report
      If R_i contains more then 40 elements then
        Exit the loop
End

That means, we initialize the report object with the contents retrieved from the first link and then consider each head and text tuple of the other pages to be inserted into the report if the content is not similar than that of previously entered contents.

3.3.1. Summarization method. The function we used in the system to retrieve the summary of an input text is *Gist(txt)*. This function is implemented in the ConceptNet server which employed a language parser MontyLingua [16] to produce a sequence of verb-subject-object-object (VSOO) frames. Summary has been produced by unifying common VSOO and selecting the other well formed VSOO frames.

3.3.2. Measuring Closeness method. *GetCloseness* function utilizes a method named *MeasureCloseness* which takes two *HTG* as input and returns a percentage value indicating the similarity-distance between them. The function is a variant implementation of traditional *TF-IDF* [5][7] scoring for text. Instead of simple *TF-IDF* scoring the function first makes two sets of words to represent the text-chunks by considering the n many word having top TF scores. Then a vector-like distance has been calculated between the text-chunks. The algorithm is given below.

$$V_1 = \{t_1 : n_1, t_2 : n_2, ...t_p : n_p\}; p = 15, n_i \neq 0$$

$$V_2 = \{s_1 : m_1, s_2 : m_2, ...s_p : m_p\}; p = 15, m_i \neq 0$$

$$maxdis(V_1, V_2) = \sqrt{\sum_{i=1}^{p} n_i^{\,2} + \sum_{i=1}^{p} m_i^{\,2}}$$

$$dis(V_1, V_2) = \sqrt{\sum_{\substack{i=1 \\ 1 \le j \le p, t_i = s_j}}^{p} (n_i - m_j)^2 + \sum_{k=1, k \neq i}^{p} n_k^{\,2} + \sum_{l=1, l \neq i}^{p} m_l^{\,2}}$$

$$closeness(V_1, V_2) = (100 - \frac{dis}{maxdis} * 100)$$

It means that each head-text tuple *(HTG)* is represented by a vector. A vector is represented by a set of tuples of p many most frequent words and their corresponding frequencies. Above shows such vectors having p many tuples where t_i and s_i indicate frequent words omitting stop-words (e.g. articles, prepositions etc.); n_i and m_i indicate their corresponding frequencies. The function *maxdis()* calculates the maximum distance if the vectors do not have any common term and *dis()* calculates the distance of the two vectors (if there is some common terms). The equal value of *maxdis()* and *dis()* indicates 100% closeness. The value given by *closeness()* function indicates the measure of content similarity between the tow text-chunks. *GetCloseness()* function keeps a list of such vector representation of each *HTG* inserted into the Report and update the list when a tuple is inserted. Hence the output 55 means, the function *GetCloseness()* has found the maximum value of 55% match of the frequent words and their frequencies of the input tuple with that of any of the existing tuples. At present we insert a new tuple if the score is less than 40. This value has been fixed by running the system with several parameters and we have noticed that for score 40 we get less repetitive but content-rich information. We also limited the maximum number of tuples to be inserted into presentation object to 40.

3.5. Presentation Builder

The Presentation Builder agent considers the report object, R_i, and takes some of the contents (e.g. maximum 5 lines of text) from each of the element to prepare presentation slides. At present the system makes maximum of 40 slides for each presentation topic. The algorithm to prepare a presentation is given below;

Input: Report Object, $R_i = \{HTG_1, HTG_2, ... HTG_R\}$

where R<=40

Begin

For each element in R_i

Initialize Scene, $S_i = \{A_1 A_2, PS_i, Seq_i\}$

$T_i = TextRandomlyTaken(HTG_i)$

$mood[i] = GuessMood\ (T_i)$

$A_1 = \{[T_1, mood_1], [T_3, mood_2]...[T_{m-1}, mood_{m-1}]\}$

$A_2 = \{[T_2, mood_2], [T_4, mood_4]...[T_m, mood_m]\}$

$PS_i = [H_i, T_i, image_i]$

$Seq_i = \{Order(A_1, A_2)\}$

End

A presentation object creates a set of scenes. A scene is created from each element of the report object. A scene, S_i, is a tuple of two character-agent objects, A_1 and A_2; a Presentation Slide Object, PS_i, and a Sequence Object Seq_i. The character agent object A_i contains m many tuples consisting of a text, T_i and $mood_i$. Text T_i is taken randomly from HTG_i and T_i contains m many lines. At present we fixed the value of m to 5. For each line we obtained the mood of the sentence using the function *GuessMood()* of ConceptNet 2.0. Odd-sequenced line of text is assigned to agent A_1 and even-sequenced line of text is assigned to A_2 agent to speak and act accordingly. A presentation-slide object is created using the heading taken from HTG_i, text T_i and an image link, $image_i$, retrieved by Google image-retrieval service. The image is retrieved by web-crawler agent using the Search-Topic, ST_i, and stores the links of images in a list. A sequence object is formed by ordering the sequence of two agents to deliver their contents verbally.

3.5.1. Selecting Text to be spoken by TTS Engine. The current heuristic of the function *TextRandomlyTaken()* is to select two lines from the top, one line from the middle and two lines from the bottom of the input text-chunk, *HTG*. These five lines of text are considered to be spoken out by the presenter agent(s) employing text-to-speech engine. If the input HTG contains less than five lines, all the texts are selected to be spoken out.

3.5.2. Affect Sensing from the Text. The function *GuessMood()* is implemented in the ConceptNet server. The system invokes this function and it performs textual affect sensing of the input text chunk. The function returns a tuple of six emotions (happy, sad, angry, fearful, disgusted, and surprise) with their respective percentage value indicating the mood of the input text-chunk after affectively classified into six affect categories using common-sense approach. The detail implementation is described in [14][15].

3.6. Presentation Avatar

The presentation avatar agent maintains a list of presenters which are, in this case, Microsoft Agent based characters. The agent first converts the scenes produced by presentation builder agent to an MPML script and web-pages (as slides). Finally the auto-generated MPML script is converted to JavaScript code in order to run the presentation on any web-browser. The Presentation file, basically an HTML file, is opened in a web-browser by

the agent and the JavaScript controls to present the slides automatically one after another by the character agents. The system makes the presentation as a hyper-linked document and hence a user can switch among the different presentations (in the case of ambiguous topic) and slides any time by interrupting the presenters.

3.6.1. The group of presenters. Currently the system employs Microsoft Agent based characters as the presenters. They are namely, Merlin, Genie, Peedy, AI, Robby and James.

3.6.2. Dynamic Generation of MPML Script. An example of automatically generated MPML script is given below. It is an XML-like language supporting to script action, affect, etc. For details about tags and MPML scripting see [18].

4. Test and Evaluation

At present we did not find such system similar to ours to make direct comparison and evaluation. But we optimistically claim that the system can successfully create report(s) and presentation(s) of whatever topic is given as input. For example we asked the system to prepare a report on *"emotion sensing from text"* and it retrieves meaningful contents from several *pdf* files and content-rich report can be created. One heading and text tuple is also given as an example.

The First Head-Text tuple obtained from the link:
http://www.networkworld.com/news/2005/0421emotipcsc.html
H: [Emotion-sensing PCs could feel your stress]
T: [Computers that can read and respond to human emotions can be more effective and reliable than computers that do not, according to Rosalind Picard, professor and founder of the Affective Computing Research Group at the Massachusetts Institute of Technology.]

Figure 2. One example of Head-Text tuple obtained from a web-page.

Te output of *GuessMood()* function is given below for a input sentence; output tuple indicates the affective affinity of the sentence.

GuessMood *(*Arsenic has been used as a cure for diseases such as syphilis and has been shown to assist in curing some leukemias.*)*
Returned Values:
disgusted (88%), surprised (40%),sad (39%), fearful (31%), happy (15%), angry (15%). So we set the mood for this text to *disgust* to the agent who speaks-out this sentence.

Figure 3. *GuessMood()* used to set mood of a sentence

As an example of ambiguous topic, when the query *"What do you know about Jaguar?"* was asked to the system, the system gave these five senses for the topic *"Jaguar"*: *Jaguar as Panthera onca; Jaguar as Car; Jaguar as Brazilian Cartoonist; Jaguar as Mac OS; and Jaguar as Rocket*. Similarly, when someone asked *"Can you tell me about virus?"* the system created the following five story-lines for the topic *"virus"* as indicated below.

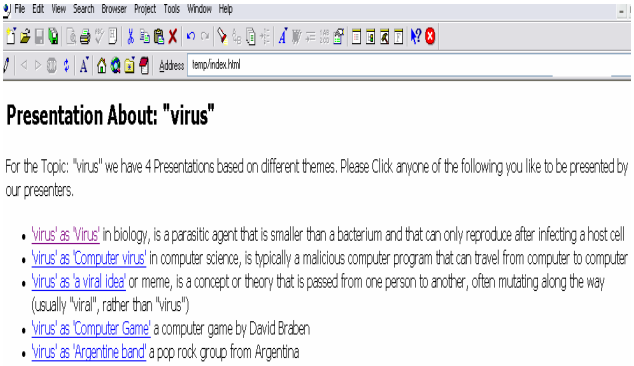

Figure 4. Five story-lines for an ambiguous topic *"virus"* are created automatically.

User can choose any of the presentation according to personal interest. By default the system automatically starts to present from the first topic and so on. While the automatic presentation is being performed, a user can navigate among the slides too.

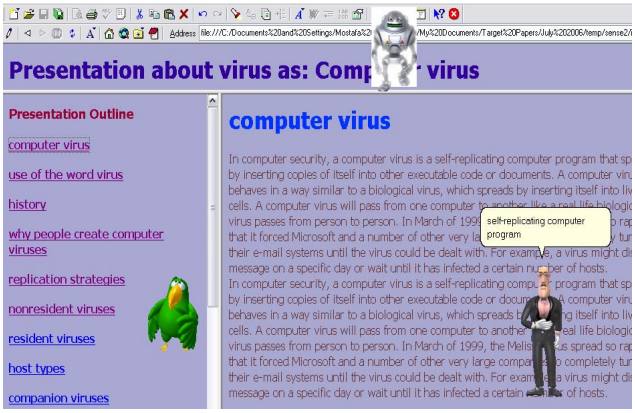

Figure 5. Auto-Presentation on *Virus* as *Computer Virus*

5. Conclusion

The purposes and functions of the discussed system are different from that of conventional systems of information retrieval in several aspects. For example, the task-oriented, semi-autonomous and collaborative multi-agent architecture emphasizes on emotion support by scripting affects by MPML-tags to make the presentation more human-like and finally a quick concept building approach around the topic has been implemented by considering several functionalities of ConceptNet 2.0. For developing the system we used MS Visual C++, Microsoft Speech API and Microsoft Agent. We admit that additional work is necessary to optimize the system so that it can support multi-user for higher loads with fast response. Further refinements in the algorithms are necessary to improve

information retrieval, extraction and association. Hence we are concentrating on the structure of web documents to develop more practical heuristics to perform data mining from the web-pages more efficiently. In future we plan to implement the system as a web-service so that any user can be able to make query and receive the result of the query as a summarized text-report and accompanied multi-modal presentation along with the list of sources of information. We also plan to perform usability study of the system in future.

10. References

[1] Amitay, E., and Paris, C., "Automatically summarizing web sites: is there any way around it?", In *Proceeding of the 9th International Conference on Information and Knowledge Management*, McLean, Virginia, November 2000, pp. 173-179.

[2] André, E., Rist, T., Mulken, S. V., Klesen, M., and Baldes, S., "The automated design of believable dialogue for animated presentation teams", In *J. Cassell, S. Prevost, J. Sullivan, and E. Churchill, editors, Embodied Conversational Agents*, The MIT Press, 2000, pp. 220-255.

[3] Ashish, N., and Knoblock, C., "Wrapper generation for semi-structured Internet sources", *ACM SIGMOD Record*, 26(4):8-15, 1997.

[4] Brin, S., and Page, L., "The anatomy of a large-scale hypertextual Web search engine", *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] Chen, L. and Sycara, K., "WebMate: A personal Agent for Browsing and Searching", In *the Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, Carnegia Mellon University, September 30, 1997, pp. 132-139

[6] Glover, E. J., Tsioutsiouliklis, K., Lawrence, S., Pennock, D. M., and Flake, G. W., "Using web structure for classifying and describing web pages", In *Proceeding of 11th International WWW Conference*, Honolulu, Hawaii, May 2002, pp 562-569

[7] Gong, Y., Liu, X., "Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis", NEC USA, C & C Research Laboratories, USA, SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA, pp. 19-25.

[8] Guan, T., and Wong, K. F., "KPS - a Web information mining algorithm", *In Proceedings of WWW8*, Toronto, Canada, May 1999, pp 1495-1507

[9] Google Web-Search Engine, http://www.google.com

[10] Henzinger, M. R., "Algorithmic Challenges in Web Search Engines", *Internet Math*, 1(1): 115–123, 2003.

[11] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A., "Extracting large-scale knowledge bases from the Web", In *Proceeding of the Int'l Conf. on Very Large Data Bases*, Edinburgh, Scotland, 1999, pp. 639-650.

[12] Kushmerick, N., "Wrapper induction for information extraction", In *Proceedings of International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997, pp.729-737.

[13] Liu, B., Chin, C. W., and Ng, H. T., "Mining Topic-Specific Concepts and Definitions on the Web", In *Proceeding of the Twelfth International World Wide Web Conference*, Budapest, Hungary, 2003, pp. 251-260.

[14] Liu, H. and Singh, P., "ConceptNet: A Practical Commonsense Reasoning Toolkit", *BT Technology Journal*, 22(4):211-226, October 2004. Kluwer Academic Publishers

[15] Liu, H., Lieberman, H., and Selker, T., "A Model of Textual Affect Sensing using Real-World Knowledge", In *Proceedings of the Seventh International Conference on Intelligent User Interfaces*, (*IUI 2003)*, Miami, pp. 125-132

[16] Liu, H., "MontyLingua v1.3.1, Toolkit and API (2003)- http://web.media.mit.edu/~hugo/montylingua

[17] Official home page of Microsoft® Agent http://www.microsoft.com/msagent

[18] Prendinger, H., Descamps, S., Ishizuka, M., "Scripting Affective Communication with Life-like Characters in Web-based Interaction Systems", *Applied Artificial Intelligence*, 16(7-8):519-553, 2002.

[19] Rakhshan, A., Holder, L. B., and Cook, D. J., "Structural Web Search Engine", *International Journal of Artificial Intelligence Tools*, 13(1):27-33, 2004.

[20] Shaikh, M. A. M., Ishizuka, M., and Islam, T., "Auto-Presentation: A Multi-Agent System for Building Automatic Multi-Modal Presentation of a Topic from World Wide Web Information", In *Proceeding of IEEE/WIC/ACM Int'l Conf. on Intelligent Agent Technology*, Compiegne, France, September, 2005, pp. 246-249.

[21] Tsutsui, T., Saeyor, S., Ishizuka, M., "MPML: A multimodal presentation markup language with character control functions", In *Proceeding of Agents'2000 Workshop on Achieving Human-like Behavior in Interactive Animated Agents*, Barcelona, Spain, June 2000, pp. 50-54.

[22] Weng, D. S. and Wu, N. X., "SiteHelper: A localized agent that helps incremental exploration of the World Wide Web", In *Proceeding of WWW6*, California,USA, April 1997, pp.691-700.

[23] Wikipedia, http://en.wikipedia.org

[24] Yahoo! Search Engine, http://www.yahoo.com