Knowledge Engineering the Web

Ben Choi

***Abstract*—This paper focuses on the largest source of human knowledge: The Web. It presents the state of the art and patented technologies on search engine, automatic organization of webpages, and knowledge-based automatic webpage summarization. For the patented search engine technology, it describes new methods to present search results to the users and through browsers to allow the users to customize and organize webpages. For the patented classification technology, it describes new methods to automatically organize webpages into categories. For the knowledge-based summarization technology, it presents new technics for computers to "read" webpages and then to "write" a summary by creating new sentences to describe the contents of the webpages. These search engine, classification, and summarization technologies build a strong framework for knowledge engineering the Web.**

***Index Terms*—Search engine, classification, web mining, document summarization, ontology, knowledge engineering.**

I. Introduction

The Web is the largest sources of human knowledge. It consists of webpages, various text documents, and multimedia files in the Internet. From a web user point of view, this vast amount information is unorganized and unmanageable. From a computer point of view, this vast amount of data is unorganized and incomprehensible. Currently most computer systems, including most search engines, process the vast amount of data simply as data without any meaning. They have no sense of knowledge as human user might have.

The future advancement is moving from data processing to knowledge processing. The advancement is to build computer systems that can, for example, “read” a document, “comprehend” the contents of the document, and “write” a summary of the document.

This paper presents three advancements that build a strong framework for knowledge engineering the Web. Firstly, it presents a patented technology for search engines and Web browsers that allows Web users to search and customization their views of the Internet. Secondly, it presents a patented technology for computers to automatically organizing webpage and documents into categories. And thirdly, it describes a knowledge-based system that can “read” webpages and text documents, “comprehend” the contents, and then “write” new sentences to summarize the document.

Manuscript received September 12, 2019; revised July 10, 2020.

The author is with Louisiana Tech University, USA (e-mail: pro@BenChoi.org).

II. Patented Search Engine Technolgy

This section is based on the author’s United State patent # 7134082 and related papers [1]-[3]. It describes a new type of search engine that allows the users to have more interaction and control of their Internet. Using the new search engine, we can treat webpages and files on the Internet like files on our own disk directory. We can browse, organize, search, and even edit Internet files the same way as we do on our files. The proposed approach is to present the search results in a hierarchical structure much like a directory tree structure. This new approach not only allows the users to click on links, but also allows them to move a link from one folder to another. The proposed approach has four advantages: providing new ways for people to use the Internet, giving people more control of their search engine and their Internet, providing new ways to organize and classify webpages found on the Internet, and providing new ways to capture the preferences of the Internet communities. The proposed approach to build a new type of search engine has been implemented and tested.

We are facing information overloaded. Finding information relevant to what we are seeking is becoming more important as the Web is growing in explosive speed. Nowadays, most people try to find whatever information on the Web by using search engines. Given a few search keywords, most search engines today will retrieve more than a few thousand webpages. The problem now is that we need to scan pages after pages, manually and time consumingly, to find what we need or often give up without getting the needed information.

We need to address the problem of helping Web users to find the information that they need. There are several approaches to address the problem. The currently most popular method to address the problem is by ordering the search results and presenting to the users the most relevant pages first. This method is called page ranking [4], [5], which is one of the important factors that makes Google currently the most successful search engine. Google uses over 100 factors in their methods to rank the search results [6]. Their methods seem to help Web users find the needed information quicker than their competitors. Even with the help of page ranking, we are facing the problem of manually performing sequential search through webpages after webpages.

Another approach to help Web users to find the information that they need is by presenting the search results in a hierarchical structure much like a directory tree structure. Using the tree structure, the Web users can browse from one group of webpages to another group, much like browsing the computer files on a directory tree. For example, if a Web user searches the word “apple”. He can focus his search on the computer group if he is looking for Apple computers. She can browse the fruit group if she is looking for healthy foods. Using this approach could largely reduce the manual search time for the Web users. To make this approach possible, Web pages need to be first classified or clustered into groups forming hierarchical structures [7], [8]. We will present another patent technology for doing the clustering and classification in the next section. We addressed some other approaches in relating to building better search engines in other publications [5], [9], [10]. In this section, we will focus on the approaches to present search results in hierarchical structures.

When we use the proposed new search engine for the first time, we get an Internet directory that contains folders and subfolders of categorized webpages and files found on the Internet. We use the Internet directory the same way as we use our own disk directory. We can move files from a folder to another, in doing so we reclassify webpages or Internet files. Whatever we do to the Internet directory is saved for us. We have more control of our Internet. Our changes to our Internet directory and other people's changes to their Internet directories are gathered and analyzed to produce the next generation of Internet directory that is given to the first-time users. The next generation of Internet directory also contains new webpages found on the Internet, which is also used to update our Internet directory. We are a member of the Internet community and we help reshape our Internet.

The new search engine described in this paper provides the following four advantages:

- Providing new ways for people to use the Internet,
- Giving people more control of their search engine and their Internet,
- Providing news way to organize and classify webpages found on the Internet, and
- Providing a new way to capture the preferences of the Internet communities.

Unlike a regular search engine that focuses on searching, the proposed new search engine also focuses on providing more control to the Web users. In addition to searching, our new search engine provides methods for allowing multiple users to arrange and modify the classification of the Web. Ontology is the first step toward knowledge engineering the Web.

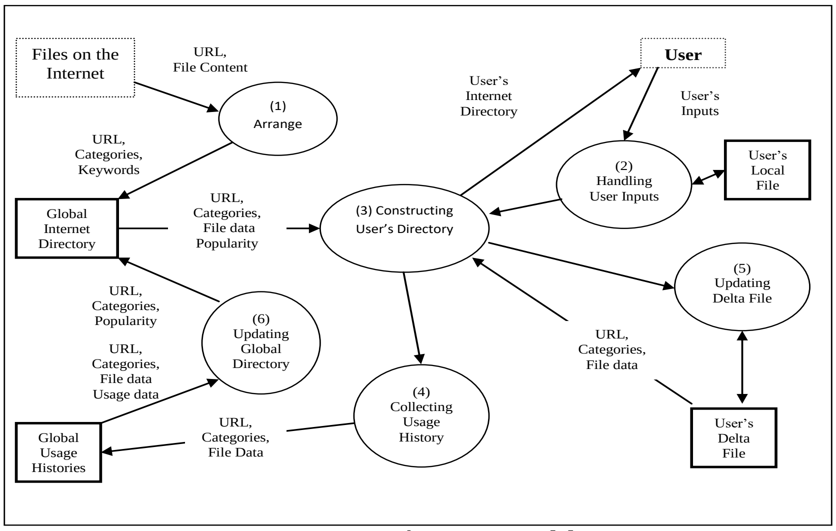

Files on the Internet
URL, File Content
(1) Arrange
URL, Categories, Keywords
Global Internet Directory
URL, Categories, File data Popularity
(3) Constructing User's Directory
User's Internet Directory
User
User's Inputs
(2) Handling User Inputs
User's Local File
(5) Updating Delta File
URL, Categories, Popularity
(6) Updating Global Directory
URL, Categories, File data Usage data
URL, Categories, File data
(4) Collecting Usage History
User's Delta File
Global Usage Histories
URL, Categories, File Data

Fig. 1. New search engine system [7].

Initially a global Internet directory is created (existing classification or directory structure can also be used). The global Internet directory is presented to a user in a directory structure that has similar looks and feels as user's own directory. In addition to searching, the users can browse the Internet directory, and can even manipulate the files in the Internet directory in the same way as manipulating their own files. For example, they can move files from one sub-directory to another, rename sub-directory or files, open sub-directory or files, and even edit files.

The results of each user's changes to the global Internet directory are stored in a file called delta file. The delta file records the user's preferred arrangement of the Internet files. For a new user a default global Internet directory is provided. The delta file is used to update the global Internet directory to create a customized Internet directory for each user.

All the users' changes to the Internet directory are collected and analyzed. The collection of the usage histories of all users records the overall preferred arrangement of the Internet files. It is used to update the global Internet directory to reflect the needs and preferences of the entire user community.

The overall working of the new search engine system is highlighted as shown in Fig. 1. The system requires six major processes as shown in six circles in the figure. For clarity, the figure describes the interaction of one Web user and the search engine, which is designed to handle millions of users. The functions of each of the six processes are described below.

(1) Arranging files in the Internet to form a global Internet directory: Webpages (or files) stored in the Internet (networked computers located all over the world) are collected, analyzed, and classified to build a global Internet directory. The directory contains links to the files, and it groups the files into meaningful categories (folders). This process also periodically updates the global Internet directory to add new files and delete obsolete links. The process further consists of Web crawler that automatically locates and retrieves webpages. It also consists of automatic webpage classification engines that automatically analyzes webpages and puts them into the appropriate categories (folders). We will describe the automatic webpage classification engine in the next section.

(2) Handling a user's inputs to arrange and modify

files in the user's Internet directory: The user can treat the user's Internet directory as user's own directory. Functions such as Move, Copy, Rename, Delete, and Create can be applied to a folder or a file. This process receives user's inputs and sends requests to the servers. The server will handle the requests and send back an updated directory.

(3) Constructing a user's Internet directory based on the global Internet directory and a user's delta file: A user's Internet directory will be constructed if the user's request is not a Search, otherwise the search request will be processed and the search results will be sent back to the client. The user's delta file contains the user's preference settings and customization data. Based on the delta file, the global Internet directory is customized to produce the user's Internet directory. The delta file contains a history of users' actions preformed on the global Internet directory. The history records the difference between the global Internet directory and the user's preferred Internet directory. The data stored in the user's delta file is sent to the server. The delta file may also be stored in the server under the user's account. Based on the data the server amends the global Internet directory and creates a customized Internet directory for the user.

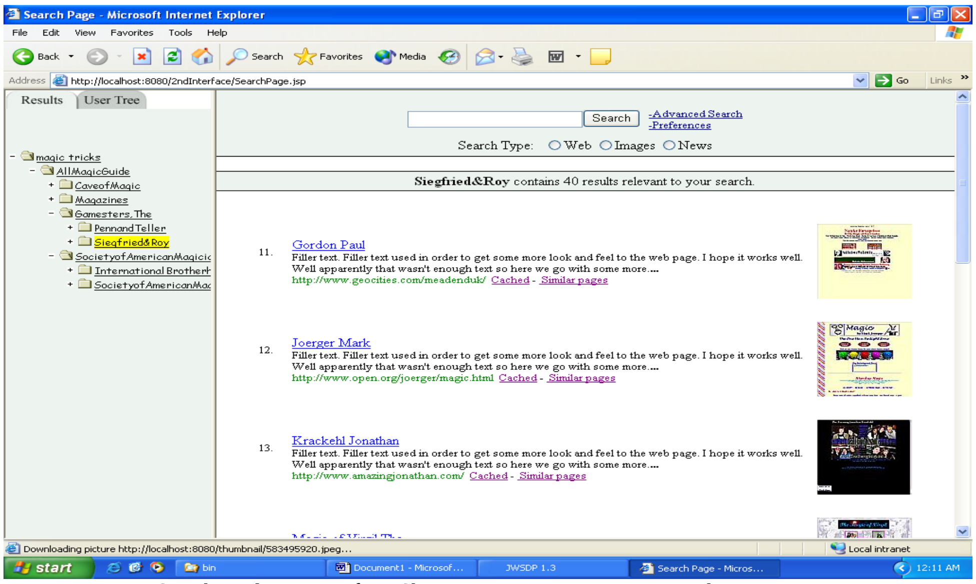
(4) Collecting user's arrangement and usage histories: The data on all the requests received in the server are collected and summarized. These data are stored in the global Usage Histories database. The data include the attributes such as the number of open functions preformed on a particular file, the number of times a file has been move from one folder to another folder, the number of times a file has been renamed, and other statistical data. These attributes are updated based on each user request.

(5) Updating the user's delta file based on the user's inputs: The user's arrangement and modification of the user's Internet directory are recorded in the delta file. The delta file will be updated each time the user makes any change to the Internet directory. The delta file will also be updated after the global Internet directory has been updated. In this case, those changes that have been incorporated into the global Internet directory will be removed from the user's delta file.

(6) Updating the global Internet directory based on the collected histories: Periodically the collected global usage histories are analyzed. The global Internet directory is updated to reflect the common preference of all users. The update includes changing the popularity attribute of the file, moving a file from one folder to another, renaming a file or a folder, and deleting a file from a folder. All these updates are based on the statistical data stored in the global usage histories database. If majority of the users prefers a new arrangement, then this new arrangement will be incorporated into the global Internet directory.

For being a search engine, the proposed new search engine also handles search requests. Fig. 2 shows one of our designs to represent the search results. In additional to present the search results in a list format ordered based on relevancy, this search engine also presents the search results in categories ordered based on relevancy, and provides the page previews, as shown in the figure. The Web users can focus and narrow down their search by selecting the folders (categories) that they are interested in. Some folders also contain sub-folders that allow users to further homing into what they are looking for.

The proposed approach to build a new type of search engine has been patented [1], implemented, and tested. Fig. 2 shows search results retrieved from our working prototype new search engine. By using Java, we were able to implement drag-and-drop functionality that allows Web users to move a webpage from one folder to another. Our search engine utilizes object database, called ObjectStore, which allows our search engine to effectively organize Web pages into categories and to effectively retrieve search results through Java interfaces.


Search Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/2ndInterface/SearchPage.jsp Go Links
Results User Tree
magic tricks
AllMagicGuide
CaveofMagic
Magazines
Gamesters, The
PennandTeller
Siegfried&Roy
SocietyofAmericanMagicic
International Brotherh
SocietyofAmericanMac
Search -Advanced Search -Preferences
Search Type: Web Images News
start bin Document1 - Microsof... JWSDP 1.3 Search Page - Micros... 12:11 AM
Local intranet

Siegfried&Roy contains 40 results relevant to your search.

11. Gordon Paul
Filler text. Filler text used in order to get some more look and feel to the web page. I hope it works well. Well apparently that wasn't enough text so here we go with some more....
http://www.geocities.com/meadenduk/ Cached - Similar pages

12. Joerger Mark
Filler text. Filler text used in order to get some more look and feel to the web page. I hope it works well. Well apparently that wasn't enough text so here we go with some more....
http://www.open.org/joerger/magic.html Cached - Similar pages

13. Krackehl Jonathan
Filler text. Filler text used in order to get some more look and feel to the web page. I hope it works well. Well apparently that wasn't enough text so here we go with some more....
http://www.amazingjonathan.com/ Cached - Similar pages

Downloading picture http://localhost:8080/thumbnail/583495920.jpeg...

Fig. 2. Search result user interface: Showing categories, summaries, and page previews.

Over twenty students have been working to develop various aspects of building new and more advanced search engine. Some students work on creating new indexing methods to speed up keyword search [11], and some work on ranking Web pages relevant to search keywords [5]. Some students work on classification and clustering of webpages [7], [8], [12] (which will be described in detail in Section II), and some work on create new parallel and distributed computer systems to support millions of search users [9], [13]. Some students work on even more advanced feature, including automatic webpage summarization [10], [14], [15], which will be described in detail in Section III. Readers are encouraged to see the related publications for details on the various aspects of building an advanced search engine that forms a foundation for knowledge engineering the Web.

III. Patented Classification Technology

Ontology is the first step toward knowledge engineering the Web. The Web contains vast numbers of webpages or documents stored in computer files located all over the world. More and more files are constantly being created and placed on the internet. The vast numbers of internet files and the speed in which the internet is growing make it impossible to use human labor to classify and organize those files into meaningful categories.

This section presents new technics to automatically organize computer files or webpages into meaning categories, to add new computer files or webpages, and to maintain the resulting organization in hierarchical directory tree structure. This section is based on the author's United State patent # 8473532 and related papers [3], [7], [12], [16]-[22].

The new technic allows a user to provide a large number of unorganized files. It partitions the unorganized files into hierarchically arranged categories that form an initial directory. It creates a description to summarize the contents of each of the categories. Then, it uses the descriptions in a classification step that assigns a newly given computer file to one of the categories. When the number of files in a category exceeds a user predefined limit, the method partitions some of the files into additional categories. Then, it updates the descriptions of a category and all its parent categories whenever additional files or categories are added into or removed from the category.

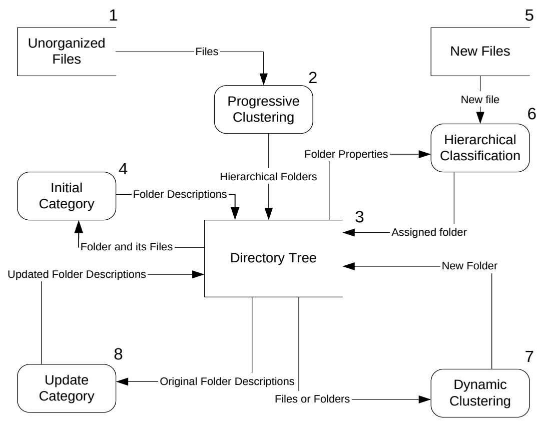


Fig. 3. Organization by clustering and classification.

Fig. 3 shows the data flow diagram for organization of computer files by clustering and classification. The technic consists of five processes (as shown in Fig. 3): Progressive Clustering (2), Initial Category (4), Hierarchical Classification (6), Dynamic Clustering (7), and Update Category (8). The technic allows a user to provide a large number of initially unorganized files or an existing initial Directory Tree. As an overview, when Unorganized Files are given, Progressive Clustering process partitions a group of unorganized files into hierarchically arranged categories that form an initial Directory Tree. This process is skipped when user provides the initial Directory Tree. Initial Category process will then take the initial Directory Tree and create a Folder Description that is an encoding to summarize the contents of each of the categories or folders in the directory. Hierarchical Classification process takes a new file and searches the descriptions of certain categories to find the most appropriate category for placing the file. When the number of files or folders in a category exceeds a user predefined limit, Dynamic Clustering process partitions some of the files or folders into additional categories that is stored as folders in the Directory Tree. Update Category process will then update the descriptions of a category and all its parent categories whenever files or folders are added into or removed from a category.

```
Main Process
     If a large number Unorganized Files is given
       Call Progressive Clustering process
       // to organize the files into Hierarchical Folders
       // to create a new Directory Tree
       Call Initial Category process
       // to generate a description for each folder
     If an existing Directory Tree is given
       Call Initial Category process
       // to generate a description for each folder
     If a New Files is given
       Call Hierarchical Classification process
     // to assign a folder for the new file
   If a file or a folder is added into or deleted from a folder
       Call Update Category process
       // to update the description of the folders from current folder to the root folder
   If the number of files or folders in a folder exceed a given limit
     Call Dynamic Clustering process
     // to create a new folder for grouping some of the files or folders
```

Fig. 4. Organizing files.

The main process is outlined in Fig. 4. If a large number of Unorganized Files is given, it will "Call Progressive Clustering process". Progressive Clustering process takes Unorganized Files and partitions the files into hierarchically arranged categories (folders) that form an initial Directory Tree. The process of operation is outlined in the following pseudo-code.

```
While the number of files or folders in
the current folder exceeds a limit:
      Call Dynamic Clustering process
      (to group some of the files or
      folders into a new folder)
      Call Progressive Clustering process
      with remaining files or folders
      (to recursively cluster the new folder)
```

Progressive Clustering process can be used to cluster files or folders. It will continue recursively clustering files or folders into new folders while the number of files or folders in the current folder exceeds a user defined limit. The clustering step is performed by calling Dynamic Clustering process, which is explained later. A new folder will contain some of the files or folders from the current folder which have a requisite degree of similarity and the remaining files or folders in the current folder are passed again recursively to the Progressive Clustering process. The recursive steps will stop when the number of files or folders in the current folder does not exceed the limit. When the Progress

Clustering process is used to cluster folders, it will organize some of the folders and put them under a new folder, thus creating a hierarchy of folders in which some folders contain other folders. The resultant hierarchy of folders forms a Directory Tree.

If an existing Directory Tree is given, Call Initial Category process. Initial Category process will then take the initial Directory Tree and create a Folder Description that is a description or encoding summarizing the contents of each of the categories (folders) in the directory. The description of each folder is used in Hierarchical Classification process for classifying new files into one of the folders in the Directory Tree. The process of operation is outlined in the following pseudo-code:

```
For each folder contained in the current folder:
    Call Initial Category process
    with the contained folder
    (to generate a description of the folder)
    Add the resulting description into
    the description of the current folder
For each file contained in the current folder:
    Generate a description for the file
    Add the resulting description into
    the description of the current folder
```

Initial Category process recursively creates a description for each folder in the directory tree. A description of a folder is generated by combining the description of each of its files and each of its folders. A description of a file or a folder is usually encoded in the form of a feature vector. The addition of two descriptions is performed by adding two vectors. The recursive steps will stop when the process completes all the folders contained in the directory tree.

If a New Files is given, Call Hierarchical Classification process. Hierarchical Classification process takes a new file and searches the descriptions of certain folders to find the most appropriate folder for the file. The process of operation is outlined in the following pseudo-code:

```
Generate a description for the new file
Let max be the similarity
(between the description of the file and
that of the root folder)
Let best folder be the root folder
Let current folder be the root folder
While current folder contains folders:
    Select the folder that has the maximum similarity
    to the file (among the contained folders)
    If the maximum similarity is larger than max:
        Let max be the maximum similarity
        Let best folder be the selected folder
    Let current folder be the selected folder
    (for continuing the search)
Put the new file into the resulting best folder
```

Hierarchical Classification process first generates a description (a feature vector) for the new file. It then searches in the directory tree for the most appropriate folder to which to assign the file. The most appropriate folder is the one that is most similar to the file. Similarity between two files or between a file and a folder is usually calculated using the dot product between two feature vectors. The search process starts at the root of the directory tree. From the root folder, it chooses the folder with the most similar feature vector to move downward toward. From the chosen folder it again chooses a folder to move downward, and so on until reaching a folder that does not contain any folder (i.e., a leaf folder). Along the search path from the root to a leaf folder, the process finds a folder that has the maximum similarity to the file. The new file is then classified and put into that folder.

If a file or a folder is added into or deleted from a folder, Call Update Category process. Update Category process will update the descriptions of a category or folder and all its parent folders whenever files or folders are added into or removed from a folder. The process of operation is outlined in the following pseudo-code:

```
If a new file is added into the current folder:
    Add the description of the file into
    the description of the current folder
If a file is deleted from the current folder:
    Recreate a description of the current folder
    (by combining all the description of
    its remaining files and its folders)
If a new folder is added into the current folder:
    Create a description of the new folder
    Recreate a description of the current folder
If a folder is deleted from the current folder:
    Recreate a description of the current folder
While the current folder is not the root folder:
    Update the description of the parent
    folder of current folder
    (by recreating the description of the parent
    folder to account for the updated description)
    Let current folder be the parent folder
    (to continue propagating the update
    upward to the root folder)
```

Update Category process adds the description of a newly created file into the description of the folder that is to contain the file. This is done by adding the feature vector for representing the file and the feature vector for representing the folder. If a file is removed from a folder, then the description of the folder is recreated using the remaining files and folders. If a new folder is created, then a description for the new folder is generated by adding the descriptions of all its files and its folders. If the new folder is put into or removed from the current folder, then the description of the current folder is recreated using all its files and folders. Since the description of a folder depends on the descriptions of all the folders contained in it, the update needs to be propagated upward from the folder to its parent folder and so on until the root folder is reached. This hierarchical arrangement of descriptions enables Hierarchical Classification process to search a single path from root to a leaf folder to find the most appropriate category for classifying a new file.

If the number of files or folders in a folder exceed a given limit, Call Dynamic Clustering process. Dynamic Clustering process partitions certain files or folders into additional categories that are stored as folders in the directory tree. Dynamic Clustering process can be used to group either files or folders into an additional folder(s). The process of operation is outlined in the following pseudo-code:

```
Let $n$ be the number of items
```

```
(which can be files or folders but not both)
Compare each pair of items to get a similarity number
Store all similarity numbers in an n*n matrix
Determine a similarity threshold from the matrix
(Partitioning Cluster method begin:)
Let current group be the group all n items
Let increasing be false and let decreasing be false
Create a queue
Repeat until a new folder is created or cannot cluster
  Let group too large be false
  While the number of items in current group is
  larger than a min limit:
    Let found be true
    For each pair (j, k) of items in the current group:
      If the similarity of the pair is
      less than the threshold
        Split the group into two groups
        (one excluding j and the other excluding k)
        Append the two groups into the queue
        Let found be false
        Break // the for loop
    If not found
      Remove the first item from the queue and
      assign it to current group
      Continue // checking the new current group
    Else // found
      If the number of items in the current group
      is larger than (n – min limit):
        Let group too large be true and found be false
        Remove the first item from the queue and
        assign it to current group
        Continue // checking the new current group
      Else: // found the right size
        Break // the while loop
    If found:
      Create a new folder to contain
      the items of the current group
      Return the new folder // done
    Else If group too large and not decreasing:
      Let increasing be true
      Increase the threshold by a factor
      Let group too large be false
      Let current group be the group of all n items
      Continue // all over with the new threshold
    Else If not increasing:
      Let decreasing be true
      Decrease the threshold by a factor
      Let current group be the group of all n items
      Continue // all over with the new threshold
    Else: // cannot be clustered with the current limit
      Return // no new folder created
```

The Dynamic Clustering process starts by identifying the total number of items "n" (files or folders) to be clustered. It compares each pair of items to determine how similar the pairs of items are. Similarity between two items is usually computed by using dot product between two vectors. The results of these comparisons are stored in a matrix for use in later steps. Based on the results, the process then determines a threshold in any conventional manner such as by taking the average similarity, the median or another percentile as the threshold. The process clusters files by partitioning the n given items into smaller and smaller groups. It starts with the n items as the initial group. It compares each pair of items in the group. If a pair (j, k) of items in the group have a similarity less than the threshold, then it splits the group into two groups; one containing all items excluding j and the other containing all items excluding k. Then, the process places the new groups into a queue and continues checking the two newly created groups. The process is continued in an iterative manner until a group is found wherein all pairs of items have a similarity larger than the threshold. To prevent a resulting group to be too small or too large, a user could provide a minimum limit (the variable "min limit" noted in the pseudo code is predefined or set by the user). A group is considered too small if the number of items in the group is less than the minimum limit. It is considered too large if the number of items in the group is larger than n minus the minimum limit. The limit is used to dynamically adjust the threshold such that the process will produce a group that is within the desired size. As shown in the pseudo code, the threshold is increased if all resulting groups are too large and is decreased if all resulting groups are too small. To prevent oscillation, after trying to increase the threshold, the process will not then decrease the threshold and likewise, after trying to decrease the threshold, the process will not increase it. If a group within the right size can be found, the process will create a new folder to hold the items in the group and return the new folder. Otherwise, no new folder is created.

This concludes the description of the invention to automatically organizing computer files by classification and clustering. This technology has been employed by companies to organize computer files on the Internet. It helps handle "information overloaded" by grouping webpages and documents hierarchical categories. It forms an important building block for a framework to knowledge engineering the Web.

IV. Knowledge-Based Summarization Technlogy

This section describes a knowledge-based system for automatic summarization [14], [15], [23], [24]. The knowledge-based system creates abstractive summary of texts by generalizing new concepts, detecting main topics, and composing new sentences. The knowledge-based system is built on the Cyc development platform that comprises the world's largest ontology of common sense knowledge and reasoning engine. The system is able to generate coherent and topically related new sentences by using syntactic structures and semantic features of the given documents, the knowledge base, and the reasoning engine. The system first performs knowledge acquisition by extracting syntactic structure of each sentence in the given documents, and by mapping the words and the relationships of words into Cyc knowledge base. Next, it performs knowledge discovery by using Cyc ontology and inference engine. New concepts are abstracted by exploring the ontology of the mapped concepts. Main topics are identified based on the clustering of the concepts. Then, the system performs knowledge representation for human readers by creating new English sentences to summarize the key concepts and the relationships of the concepts. The structures of the composed sentences extend beyond subject-predicate-object triplets by allowing adjective and adverb modifiers. The system was tested on various documents and webpages. The test results showed that the system is capable of creating new sentences that include generalized concepts not mentioned in the original text and is capable of combining information from different parts of the text to form a summary.

To create a summary of a document is not an easy task for a person or for a machine. For us to be able to summarize a document requires that we understand the contents of the document. To be able to understand a document requires the ability to process the natural language. It also requires the background knowledge of the subject matter and the commonsense knowledge of humanity. Despite the active research in Artificial Intelligence in the past half century, currently there is not machine that can understand the contents of a document and then summarizes the document based on its understanding.

Most past researches in automatic document summarization did not attempt to understand the contents of the documents, but instead used the knowledge of writing styles and document structures to find key sentences in the document that captured the main topics of the documents. For instance, knowing that many writers use topic sentences, the first sentence of a paragraph is considered as the key sentence that summarizes the contents of the paragraph.

Our research described in this section represents a small step toward the use of semantic contents of a document to summarize the document. There is a long way before we can try to use the word “understand” to describe the ability of a machine. Our research is recently made possible by the advance in natural language processing tools and the availability of large databases of human knowledge. For processing natural language, we chose Stanford parser [25] and SpaCy, which can partition an English sentence into words and their part-of-speech. To serve as the background knowledge of the subject matter and the commonsense knowledge of humanity, we chose ResearchCyc [26], which currently is the world's largest and most complete general knowledge base and commonsense reasoning engine.

With the help of the natural language processing tool and the largest knowledge base, our system is able to summarize text documents based on the semantic and conceptual contents. Since the natural language tool and the knowledge base only handle English, our system is currently only applicable to English text documents including texts retrieved from Web pages. When those tools are available for other languages, our proposed approaches should be able to be extended to process other languages as well. Since we use a knowledge base that organized concepts into hierarchies forming ontology in the domain of human consensus reality, our system is one of the first ontology-based summarization system.

Our system consists of knowledge acquisition, knowledge discovery and knowledge representation for human reader (Fig. 5) [15]. For knowledge acquisition, our system takes documents as inputs and transforms them into syntactic representation. It maps each word in the text to the appropriate Cyc concept and assigns word’s weight and associations to that concept. For knowledge discovery process, the system finds ancestors for each mapped Cyc concept, records ancestor-descendant relation and adds scaled descendant weight and descendant associations to the ancestor concept. This process allows system to abstract new concepts not explicitly mentioned in the original text. The system identifies main topics described in the text by clustering mapped Cyc concepts. During the knowledge representation process, the system creates English sentences for the most informative subjects identified in main topics. This process ensures that the summary sentences are composed using information from different parts of the text while preserving coherence to the main topics.

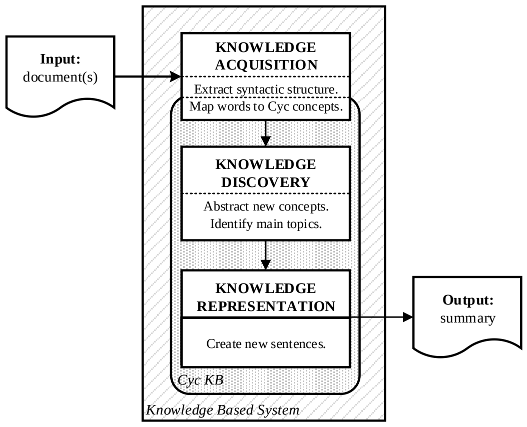


Fig. 5. Knowledge processing.

Mapping sub-process:

- For each word obtained on preprocessing step:
  - Map word to corresponding Cyc concept;
  - Assign word’s weight to mapped Cyc concept;
  - Map association head to corresponding Cyc concept;
  - Assign word’s association to mapped Cyc concept.

Fig. 6. Mapping words to Cyc concepts.

Knowledge acquisition consists of two sub-processes. Firstly, pre-processing extracts syntactic structure of the given document. It separates text into sentences, lemmatizes each word and assigns part of speech tags and dependency parser associations. Then it counts the weights of the words and their associations. Secondly, mapping process finds matching Cyc concepts for each word in the input text. Once the system finds appropriate concept, it assigns word’s weight and associations to that concept. Mapping sub-process is described in Fig. 6. Word’s weight is a frequency that is the number of times it is mentioned in the text. The association is a relation between two words in a sentence, derived by the syntactic parser. Each association has a weight assigned to it that shows how many times two words were used together in the text. Higher weights represent stronger associations. Cyc ontology contains semantic knowledge about the concepts and our system enhances it with syntactic structure features. Semantic knowledge and syntactic structure are two crucial parts that make summary cohesive and meaningful.

Knowledge discovery process consists of two sub-processes: (i) new concepts abstraction (Fig. 7) and (ii) main topics detection (Fig. 8). The first process (Fig. 7) starts by deriving ancestor for each concept mapped from the text. It assigns ancestor-descendant relation to derived ancestor and keeps track of descendants’ scaled weight. The scaling is defined by generalization parameter α. Next, it adds descendants’ weight and associations to ancestor concept if descendant-ratio is higher than the threshold. The threshold is defined by generalization parameter β. The descendant ratio is the number of mapped descendants divided by the number of all descendants of a concept. The parameters α and β regulate desired level of generalization. Higher α and lower β yield greater level of generalization giving more emphasis to ancestor terms. New concepts abstraction is an important part of summarization as it allows generalizing information derived from the input text. For example, our system can generalize "apple", "orange" and "mango" to an ancestor concept "fruit", which might not be mentioned in the text.

```
(i) New concepts abstraction sub-process:
o For each mapped Cyc concept:
  • Find concept's ancestor;
  • Record ancestor-descendant relation;
  • Update ancestor's number of descendants;
  • Update ancestor's descendants weight;
    ▪ Scale descendant's weight by α.
o For each mapped Cyc concept that has descendants:
  • Find the number of concept's mapped descendants;
  • Find the number of all concept's descendants;
  • Calculate descendant ratio:
        $desc\_ratio = \frac{\#\ mapped\ descendants}{\#\ of\ all\ descendants}$
  • If descendant-ratio is larger than β:
    ▪ Add descendants' weight to ancestor's weight;
    ▪ Add descendants' associations to ancestor associations;
      • Scale descendant's association weight by α.
```

Fig. 7. Abstracting new concepts.

```
(ii) Main topics detection sub-process:
o For each mapped Cyc concept:
  • Find defining micro theories;
o Count the frequencies of discovered micro theories;
o Pick top-n micro theories with highest frequencies.
```

Fig. 8. Detecting main topics.

```
(i) Candidate subjects discovery sub-process:
o For each micro theory in top-n micro theories:
  • For each concept mapped from the text:
    ▪ Find number of subject associations;
    ▪ Find number of all associations;
    ▪ Calculate subjectivity ratio:
        $subj\_ratio = \frac{\#\ of\ subject\ associations}{\#\ of\ all\ associations}$
    ▪ Calculate subjectivity rank:
        $subj\_rank = concept\_weight * subj\_ratio$
  • Pick top-n subjects with highest subjectivity rank.
```

Fig. 9. Discovering candidate subjects.

The second process (Fig. 8) detects main topics in the text. The assumption is that the topics are represented by the most frequent micro theories in Cyc knowledge base. Micro theories are the clusters of concepts and facts typically representing one specific domain of knowledge. For example, #$MathMt is the micro theory containing mathematical knowledge. Micro theories are the basis of the knowledge representation in Cyc. To find the most frequent micro theories, system derives the defined micro theories for each mapped Cyc concept, counts the frequencies of the discovered micro theories and picks the top-n micro theories with the highest frequencies.

Knowledge representation process starts by choosing concepts with the highest subjectivity rank (Figure 9). The chosen concepts become the candidate subjects for creating new sentences. Subjectivity rank is defined as the product of concept weight and subjectivity ratio. Subjectivity ratio is defined as the number of concept associations labelled as subject relation divided by the total number of concept associations. This ratio helps to identify concepts with the strongest subject roles in the text.

```
(ii) New sentence generation sub-process:
o For each subject in top-n subjects:
  • Convert subject Cyc concept to natural language representation (a);
  • Pick adjective with highest subject-adjective association weight;
  • Convert adjective Cyc concept to natural language representation (b);
  • Pick top-n predicates with highest subject-predicate association weights;
  • For each predicate in top-n predicates:
    ▪ Convert predicate Cyc concept to natural language representation (c);
    ▪ Pick adverb with highest predicate-adverb association weight;
    ▪ Convert adverb Cyc concept to natural language representation (d);
    ▪ Pick top-n objects with highest product of subject-object and predicate-object associations weights;
    ▪ For each object in top-n objects:
      • Convert object Cyc concept to natural language representation (e);
      • Pick adjective with highest object-adjective association;
      • Convert adjective Cyc concept to natural language representation (f);
      • Create new sentence using subject (a), subject-adjective (b), predicate (c), predicate-adverb (d), object (e), object-adjective (f) natural language phrases.
```

Fig. 10. Creating new sentences.

The system then creates new English sentences for the candidate subjects (Fig. 10) [15]. To generate new sentences, system uses subject–predicate–object structure enhanced with the adjective modifiers for subjects and objects, and the adverb modifiers for predicates, when available. Subject, predicate and object elements are mandatory while adjective and adverb modifiers are optional. The system chooses candidate elements for the sentence using the weight of the association between the concepts. The created sentences form the final summary of a given document.

The system was tested on various documents and webpages. The test results showed that the system is capable of creating new sentences that include generalized concepts not mentioned in the original text and is capable of combining information from different parts of the text to form a summary [14], [15], [27].

V. Conclusion and Future Work

Ontology is the first step toward knowledge engineering the Web. This paper outlined new approaches to build a new type of search engine that allows the users to have more interaction and control of their Internet. The proposed approaches organize webpages into hierarchical structures of categories. They allow the Web users to organize the webpages found in the Internet much like interacting on files stored their own disk directory tree.

The vast numbers of internet files and the speed in which the internet is growing make it impossible to use human labor to classify and organize those files into meaningful categories. Thus, this paper also presents new technics to automatically organize computer files or webpages into meaning categories, to add new computer files or webpages, and to maintain the resulting organization in hierarchical directory tree structure.

Moving toward knowledge, our research employs the largest ontology of common-sense knowledge and powerful inference engine as basis to build a knowledge-based summarization system. We attempt to address the challenging advancement of creating computer system that can "read", "comprehend", and "write".

These search engine, automatic classification, and automatic summarization technologies build a strong framework for knowledge engineering the Web. Although we have made progress on these attempts, much work remains to be done in this area of research.

The future advancements should focus on moving from data to knowledge, from data processing to knowledge processing. As soon as we are dealing with the word "knowledge", we must address the fundamental building block of knowledge, that is a knowledge database that must consist of common-sense knowledge and inference engine. Thus, another future advancement should focus on building better and more comprehensive knowledge database. More powerful computing systems requires the ability to process knowledge. Knowledge is Power!

Conflict of Interest

The author declares no conflict of interest.

Author Contributions

The author is the inventor of the patents and also is the creator and the supervisor of the projects described in this paper.

References

[1] B. Choi, "Method and apparatus for individualizing and updating a directory of computer files," United States Patent # 7,134,082, November 7, 2006.

[2] B. Choi, "My internet," in *Proc. the 2010 International Conference on Web Information Systems and Mining*, October 2010, pp. 171-175.

[3] B. Choi, "Making sense of search results by automatic web-page classifications," in *Proc. World Conference on the WWW and Internet*, pp.184-186, 2001.

[4] Berkhin P., "A survey on pagerank computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73-120, 2005.

[5] B. Choi and S. Tyagi, "Ranking web pages relevant to search keywords," in *Proc. The IADIS International Conference on WWW/Internet*, November 2009, pp. 200-205.

[6] Vaughn. (2008). Google search engine optimization information. [Online]. Available: http://www.vaughns-1-pagers.com/internet/googlerankingfactors.htm

[7] B. Choi and X. Peng, "Dynamic and hierarchical classification of web pages," *Online Information Review*, vol. 28, no. 2, pp. 139-147, 2004.

[8] Z. Yao and B. Choi, "Clustering web pages into hierarchical categories," *International Journal of Intelligent Information Technologies*, vol. 3, no. 2, pp. 17-35, April-June, 2007.

[9] B. Choi and R. Dhawan, "Agent space architecture for search engines," in *Proc. International Conference on Intelligent Agent Technology*, 2004, pp. 521-525.

[10] Choi B. and X. Huang, "Web page summarization by using concept hierarchies," in *Proc. International Conference on Agents and Artificial Intelligence*, January 2009, pp. 281-286.

[11] S. Baberwal and B. Choi, "Speeding up keyword search for search engines," in *Proc. the 3rd IASTED International Conference on Communications, Internet, and Information Technology*, 2004, pp. 255-260.

[12] B. Choi and Z. Yao, "Web mining by automatically organizing web pages into categories," *Distributed Artificial Intelligence, Agent Technology, and Collaborative Applications*, chapter XII, pp. 214-231, 2008.

[13] B. Choi and R. Dhawan, "Distributed object space cluster architecture for search engines," in *Proc. High Availability and Performance Computing Workshop*, 2003.

[14] A. Timofeyev and B. Choi, "Knowledge based system for composing sentences to summarize documents," *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pp. 164-183, 2019.

[15] A. Timofeyev and B. Choi, "Knowledge based Automatic Summarization," in *Proc. the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2017, vol. 1, pp. 350-356.

[16] B. Choi and Z. Yao, "Web page classification," *Foundations and Advances in Data Mining*, Springer-Verag, pp. 221-274, 2005.

[17] B. Choi, "Method and apparatus for automatic organization for coputer files," United States Patent # 8,473,532, June 25, 2013.

[18] Peng X. and B. Choi, "Document classifications based on word semantic hierarchies," in *Proc. the IASTED International Conference on Artificial Intelligence and Applications*, 2005, pp. 362-367.

[19] Z. Yao and B. Choi, "Automatically discovering the number of clusters in web page datasets," in *Proc. the 2005 International Conference on Data Mining*, 2005, pp. 3-9.

[20] B. Choi and Q. Guo, "Applying semantic links for classifying web pages," *Developments in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 2718, pp. 148-153, 2003

[21] Z. Yao and B. Choi, "Bidirectional hierarchical clustering for web mining," in *Proc. IEEE/WIC International Conference on Web Intelligence*, 2003, pp. 620-624.

[22] X. Peng and B. Choi, "Automatic web page classification in a dynamic and hierarchical way," in *Proc. IEEE International Conference on Data Mining*, 2002, pp. 386-393.

[23] B. Choi and X. Huang, "Web page summarization by using concept hierarchies," in *Proc. International Conference on Agents and Artificial Intelligence*, Proto, Portugal, January, 2009, pp. 281-286.

[24] B. Choi and X. Huang, "Creating new sentences to summarize documents," in *Proc. The 10th IASTED International Conference on Artificial Intelligence and Application*, February 2010, pp. 458-463.

[25] C. Manning and D. Jurafsky, *The Stanford Parser: A Statistical Parser*, The Stanford Natural Language Processing Group, 2008.

[26] Cycorp, ResearchCyc. (2019). [Online]. Available: http://www.cyc.com

[27] A. Timofeyev and B. Choi, "Building a knowledge based summarization system for text data mining," *Machine Learning and Knowledge Extraction*, pp. 118-133, 2018.

Ben Choi has a PhD degree in electrical and computer engineering and a pilot license for flying airplanes and helicopters. He is an inventor owning two patents on creating new search engines and on automatic organization of computer files. He is the editor of the book: Humanoid Robots. He was a keynote speaker at conferences speaking on Knowledge Engineering the Web and on creating multivalued computers. He received B.S., M.S., and PhD degrees all from The Ohio State University. He was a system performance engineer at (Bell Labs) Lucent Technologies and a visiting research scholar at DePaul University, University of Western Australia, and Hong Kong University of Science and Technology. He is an associate professor in computer science at Louisiana Tech University, researching on artificial intelligence, robotics, machine learning, knowledge engineering, data mining, multiagent system, fuzzy systems, parallel computing, and on theorizing the Universe as a computer.