www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 12 December 2014, Page No. 9543-9546

Creating a Knowledge Base by extracting Top-K lists from the web

Ramya.Y, K.Ramana Reddy
M.Tech Student,CSE
Aurora's Technological & Research Institute
Hyderabad,Telangana,India
Assistant Professor,CSE Department
Aurora's Technological & Research Institute
Hyderabad,Telangana,India
ramya07chitti@gmail.com

Abstract: The World Wide Web is currently the largest source of information. However, most information on the web is unstructured text in natural languages, and extracting knowledge from natural language text is very difficult. Still, some information on the web exists as lists or web tables coded with specific tags such as <ul>, <li>, and <table> on html pages. However, it is questionable how much valuable knowledge we can extract from lists and web tables. It is true that the total number of web tables is huge in the entire corpus, but only a very small percentage of them contain useful information. Nowadays we have So many Search engines. These search engines provide top-k lists as search results. The search results contains huge amount of information in which the user is not interested to visit all the pages except the top 2 or 3.Moreover the search results may also contain unwanted data. Hence to avoid this drawback we will be developing a better method for mining top-k lists. In proposed system we use Path Clustering Algorithm which better processes the top-k web page. The system displays only required top lists related to our top-k title. It is very useful to the user which saves user's time. The extracted lists can also be used as background knowledge for Q/A system. We present an efficient method that extracts top-k lists from web pages with high performance. This system collects top-k lists of various interests which can be called a knowledge base and provides a search option to mine them.

Keywords: Web information extraction, top-k lists, list extraction, web mining.

I. INTRODUCTION

In World Wide Web much of the information contains structured data such as tables and lists which are very valuable for knowledge discovering and data mining. This structured details because it is relatively easier to unlock information from data with some regular patterns than free text which makes up most of the web content. However, when encoded in HTML, structured data becomes *semi-structured*. And because HTML is designed for rendering in a browser, different HTML code segments can give the same visual effect at least to the human eye. As a result, HTML coding is much less stringent than XML, and inconsistencies and errors are abundant in HTML documents. All these pose significant challenges in the extraction of structured data from the web [1]. In this demo, we focus on list data in web pages. In particular, we are interested in extracting from a kind of web pages which present a list of *k* instances of a topic or a concept. Examples of such topic include "20 Most Influential Scientists Alive Today", "Ten Hollywood Classics You Shouldn't Miss", "50 Tallest Persons in the World". Informally, our problem is: given a web page with a title that contains a integer number *k*, check whether the page contains a list of *k* items as its *main content*, and if it does, extract these *k* items. We call such lists *top-k lists* and pages that contain the entirety of a top-k list *top-k pages*[2]. There are also lists that span multiple pages but are connected by hyperlinked "Next" button, but these are not considered in this work. A typical scenario we consider, like the one in Figure1, is that each list item is more than an instance of the topic in the title, but instead contains additional information such as a textual description and images. Our objective is to extract the actual instance whenever possible, but in the worst case, produce list items that at least *contain* the wanted instances.

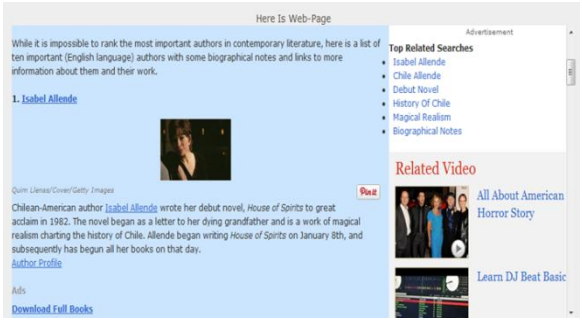

Fig 1.Sample Top-K web page

The main approach of this work goes along the line of analyzing similar tag paths in the DOM tree with the help of visual features to identify the main list in a page. The key contributions of this demo are:

- We defined a novel top-k list extract problem which is useful in knowledge discovery and fact answering;
- We designed an unsupervised general-purpose algorithm along with a number of key optimizations that is capable of extracting top-k lists from any web pages.

Index	Content	URL
1	Isabel Allende	http://contemporarylit.about.com/od/authorprofiles/p/allende.htm
2	Margaret Atwood	http://contemporarylit.about.com/cs/authors/p/atwood.htm
3	Jonathan Franzen	http://contemporarylit.about.com/cs/authors/p/franzen.htm
4	Ian McEwan	http://contemporarylit.about.com/od/authorprofiles/p/mcewan.htm
5	David Mitchell	http://contemporarylit.about.com/od/authorprofiles/p/mitchell.htm
6	Toni Morrison	http://contemporarylit.about.com/cs/authors/p/morrison.htm
7	Haruki Murakami	http://contemporarylit.about.com/od/authorprofiles/p/murakami.htm
8	Philip Roth	http://contemporarylit.about.com/od/authorprofiles/p/roth.htm
9	Zadie Smith	http://contemporarylit.about.com/od/authorprofiles/p/smith.htm
10	John Updike	http://contemporarylit.about.com/cs/authors/p/updike.htm

Fig 2.Sample extracted list

II.PROBLEM DEFINITION

Let a web page be a pair (t, d) where tis the page title, and d is the HTML body of the page.

- A page (t, d) is a top-k page if:

1) From title t we can extract "Top" and "k" .

2) Where k is a natural number.

3) From the page body d we can extract k and only k items.

.

The top-k extraction problem can then be defined as three sub-problems (in terms of three functions):

- Title recognition F1 : (t, d)→(k, "top")
- List extractor F2 : (k) → I Where I is the set of terms which are instances of c and |I| = k
- Content extractor F3 : (c, d, I) →(T, S)

Where T is a table of attribute values for the elements in I and S is its schema.

III OUR APPROACH

Algorithm: *TAG PATH CLUSTERING ALGORITHM.*

. Our basic algorithm runs in four steps[3][4]. First, we compute the tag path for every node in the DOM tree of the input page. An HTML *Document Object Model* (DOM) is a tree structure whose nodes are HTML elements identifiable by HTML tags and associated attributes. Second, we group nodes with an identical tag path into one *equivalence class* [5], from which we extract exactly *k* members as our candidate list. A *tag path* is a path from the root to an arbitrary node in the DOM tree. A top-*k* list item or *list item* in short, is a segment of HTML page which represents a unique item in the top-*k* list of a top-*k* page. A list item can contain multiple *item components* such as a title (which is often the name of an entity), some descriptive text or an image. Now the item components that belong to the same list item are grouped together. Finally the candidate list is projected as a top-k list. If there doesn't exist any candidate list, then this input page is not a top-*k* page.

The basic algorithm solves most but not all top-*k* list extraction problems. To improve the result quality and the performance, we further introduce four optimization heuristics [6].

1) Visual Area: In the basic algorithm, the relative importance of a candidate list to the whole page is calculated as number of text characters in the list against the total number of text characters. This doesn't work when the list contains large non-text areas such as white space or images, or the list uses large fonts. As an enhancement, we try to estimate the total *visual area* of the candidate list versus the total area of the page. This can done by the calculating the combination of image sizes, font sizes and potential white spaces.

2) Interleaving Lists: For aesthetic reasons, a top-*k* list may have an "interleaving pattern", where list items have alternate visual styles such as background colors or fonts. In the basic algorithm, this kind of list may be treated as two lists of size *k*=2 and removed mistakenly. We include a special heuristic to detect such interleaving patterns and reconstruct the whole list from two smaller candidate lists.

3) *k* + 1 Problem: Some top-*k* pages have a top-*k* list with an additional header or footer that looks almost exactly the same and has the same tag path as the real list items. The basic algorithm cannot distinguish between the real items and this "fake" item. There is a similar *k*+1 problem where the first or the last item of the real top-*k* list has a slightly different style and is excluded from the

equivalence class. We pay special attention to equivalence classes of size k + 1 and k -1 and try to identify these boundary cases by comparing the sub tree of the items or analyzing the text content.

4) Item Title Extraction. From our observation of many top-k pages from different domain, we generalized a number of rules for identifying the title entities in a top- k list. For example, titles are often placed before other item components, and often occupy a single line in the rendered display with stylized fonts.

IV .IMPLEMENTATION DETAILS

As the input of the system, the web page is first parsed by a HTML parser [3] to obtain a complete DOM representation. Then the title classifier attempts to recognize the page title. If it is a "top-k like" title, the classifier outputs the list size (the number k) and a set of possible concepts mentioned in the title. With the number k, the candidate picker extracts all lists of size k from the page body as candidate lists [6]. Only one of them will be the actual list of interest. With the concept set, the top-k ranker can score each candidate list and pick the best one as the "top-k" list. Finally the content processor analyzes the list content and extracts the entity names and attributes.

1 Title Classifier

The title of a web page (string enclosed in <title> tag) helps us identify a "top-k" page. The goal of the classifier is to recognize "top-k like" titles[7], the likely name of a "top-k" page. In general, a "top-k like" title represents the topic of "top k" list. Note that a "top-k like" title may contain multiple segments, and usually only one segment describes the topic or concept of the list.

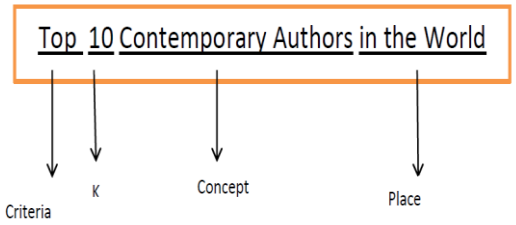

Fig 3.Top-K Title

2) Candidate Picker

Given an HTML page body and the number k, the candidate picker collects a set of lists as candidates. Each list item is a text node in the page body. We define a *tag path* of a node as a path from the root to this node in the DOM tree. Items in a "top-k" list usually have similar format and style, and therefore they share an identical tag path. Our *Default* algorithm takes advantage of this observation to identify lists of size k. Note that there might be multiple lists of the same size from a given page.

3) Content Processor

The content processor takes as input a "top-k" list and extracts the main entities as well as their attributes. Sometimes the text within an HTML text node contains a structure itself, e.g. "Hamlet By William Shakespeare". The content processor infers the structure of the text by building a histogram for all potential separator tokens such as "By", ":" and "," from all the items of the "top-k" list. If we identify a sharp spike in the histogram for a particular token, then we successfully find a separator token, and we use that token to separate the text into multiple fields.

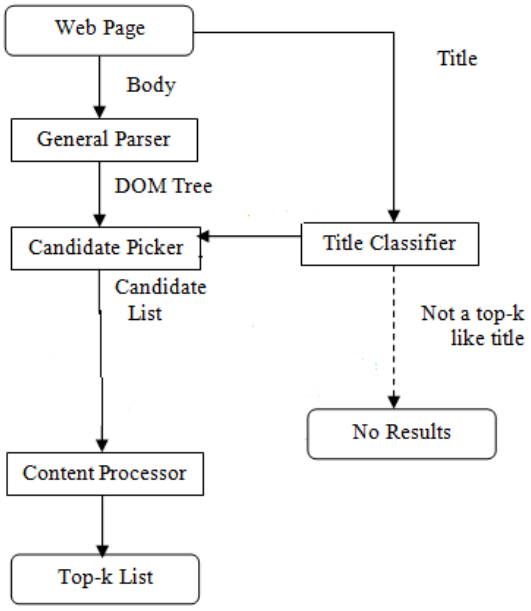

Fig 4.System Flow

V. EXPERIMENTAL RESULTS

Experiments are made with the prototype application that demonstrates the usefulness of the application. The application is evaluated with measures such as precision, recall and F-measure. The results are evaluated using data and location based, rule based and learning based approaches.

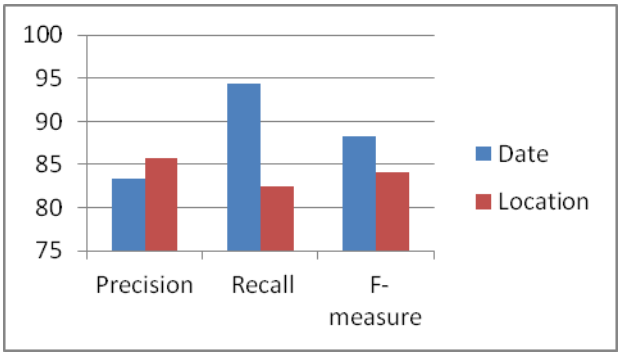

Figure 5– Results with respect to date and location

As can be seen in Figure 5, it is evident that the rule based and learning based approaches are evaluated. When compared with the rule based approach, the learning based approach exhibited high precision and recall.

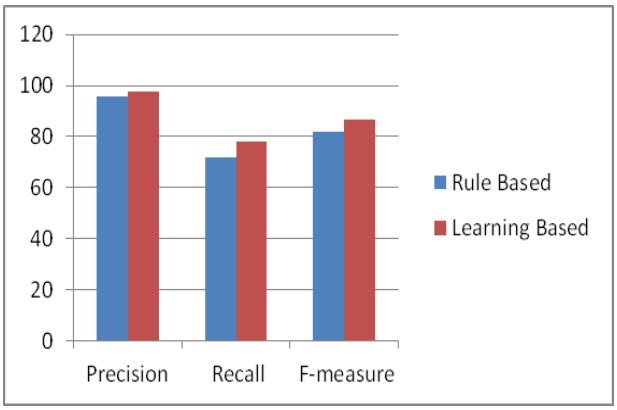

Figure 6 – Results with respect to list extraction

As can be seen in Figure 6, it is evident that the rule based and learning based approaches are evaluated. When compared with the rule based approach, the learning based approach exhibited high precision and recall.

VI .CONCLUSION AND FUTURE WORK

In this paper, we study the automatic extraction of top-k lists from web pages available over WWW. We take multiple web pages as input and generate top-k lists from that in order to make a summary of top-k lists. The summarized top-k lists can help users to gain knowledge that will help in making well informed decisions. We built a prototype web application that takes web pages as input and extract top-k lists from the given pages. The output is the well formatted top-k summary that is more meaningful and avoids searching for the required information. It does mean that the output can be taken away by the user directly. Our project can be extended to extract information from nested links and try to reduce computational work[8][9].

References

1. Soumen Chakrabarti Mining The Web: Discovering Knowledge From Hypertext Data".
2. Zhixian Zhang, Kenny Q. Zhu , Haixun Wang Hongsong Li ,"Automatic Extraction of Top-k Lists from the Web",, IEEE , ICDE Conference, 2013, 978-1-4673-4910-9.
3. G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser,"Extracting data records from the web using tag path clustering".
4. G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser, "Extracting data records from the web using tag path clustering," in WWW, 2009, pp. 981–990.
5. C.-H. Chang and S.-C. Lui, "Iepad: information extraction based on pattern discovery," in WWW, 2001, pp. 681–688.
6. Smith Tsang, Ben Kao, Kevin Y. Yip, Wai-Shing Ho and Sau Dan Lee, "Decision Trees for Uncertain Data",IEEE conference,2011
7. F Z. Zhang, K. Q. Zhu, and H. Wang, "A system for extracting top-k lists from the web," in KDD, 2012
8. . W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain independent information extraction from web tables", In WWW, pages 71–80. ACM Press, 20
9. W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in SIGMOD, 2012.
10. Fumarola, T. Weninger, R. Barber, D. Malerba, and J. Han, "Extracting general lists from web documents: A hybrid approach," in IEA/AIE (1), 2011, pp. 285–294.
11. J. Wang, H. Wang, Z. Wang, and K. Q. Zhu, "Understanding tables on the web," in ER, 2012, pp. 141–155.
12. M. J. Cafarella, E. Wu, A. Halevy, Y. Zhang, and D. Z. Wang, "Webtables: Exploring the power of tables on the web", in VLDB Auckland, New Zealand, 2008.