FAQ Mining via List Detection

Yu-Sheng Lai[1,2], Kuao-Ann Fung[1], and Chung-Hsien Wu[1]
[1]Dept. of Computer Science & Information Engineering, National Cheng Kung University,
1, Ta-Hsueh Rd., Tainan, Taiwan 701, R.O.C.
[2]Advanced Technology Center,
Computer & Communication Research Laboratories/Industrial Technology Research Institute,
E000, 195-11, Sec. 4, Chung Hsing Rd. Chutung, Hsinchu, Taiwan 310, R.O.C.
laiys@itri.org.tw, p7689151@dec4000.cc.ncku.edu.tw, chwu@csie.ncku.edu.tw

Abstract

This paper presents an approach to FAQ mining via a list detection algorithm. List detection is very important for data collection since list has been widely used for representing data and information on the Web. By analyzing the rendering of FAQs on the Web, we found a fact that all FAQs are always fully/partially represented in a list-like form. There are two ways to author a list on the Web. One is to use some specific tags, e.g. <li> tag for HTML. The lists authored in this way can be easily detected by parsing those special tags. Another way uses other tags instead of the special tags. Unfortunately, many lists are authored in the second way. To detect lists, therefore, we present an algorithm, which is independent of Web languages. By combining the algorithm with some domain knowledge, we detect and collect FAQs from the Web. The mining task achieved a performance of 72.54% recall and 80.16% precision rates.

Introduction

The World Wide Web has become a fertile area, storing a vast amount of data and information. One of them we are interested is the Frequently Asked Questions (FAQs). For customer services, message providing, etc., many Websites have created and maintained their own FAQs.

A large collection of FAQs is very useful for many research areas in natural language processing. Especially in question answering, it exemplifies many questions and their answers. It is also a database for the applications of FAQ retrieval, e.g. AskJeeves (www.ask.com), .faq finder (members.tripod.com/~FAQ_Home/), and FAQFinder (www1.ics.uci.edu/~burke/faqfinder/).

By analysing the rendering of FAQs on the Web, we divide them into 6 types according to 2 viewpoints. Among these types, we found a fact that all FAQs are always fully/partially represented in the form of list as well as much useful information.

There are two ways to represent a list in a Web Page. One is to use some specific tags, e.g. <li> tag for HTML. Another one is to use other tags. The lists authored in the first way can be easily detected by parsing those specific tags. However, most of FAQs are authored in the second way. Therefore, this paper presents an algorithm for detecting lists in Web Pages. Then, we verify each detected list whether it determines a set of FAQs or parts of it by some constraints of domain knowledge.

1 Web FAQs

An FAQ file is a file gathering a set of question-answer pairs, QA-pairs for short, with an identical topic together. A Website may contain FAQ files with one or more topics. The FAQ files authored in Web Pages are called Web FAQs. In the following we will explore Web FAQs from three aspects : (1) Web languages used for authoring Web FAQs, (2) taxonomy of Web FAQs and (3) domain knowledge.

1.1 Web Languages

Web languages are the languages developed and used for authoring Web Pages. A popular Web language – SGML (Standard Generalized Markup Language) [ISO 8879, Goldfarb 1990] is a metalanguage developed for tagging text. It provides the rules for defining a markup language based on tags. For example, the most popular markup language used for the Web, HTML, is an instance of SGML. For simplicity and clearness, all examples in this paper will be authored in HTML [Raggett et al. 1998]. Besides, an available metalanguage for Semantic Web, XML (eXtensible Markup Language) [Bray et al. 2000], is also a subset of SGML.

A European alternative to SGML is the ODA (Office Document Architecture) that is also a standard [ISO 8613]. However, it is not used very frequently nowadays.

Fig. 1 shows the taxonomy of SGML-based Web languages [Baeza-Yates and Ribeiro-Neto 1999]. Under the definition of SGML, each instance of SGML describes a text by using the text itself and tags. Our approach is available for the Web Pages authored by SGML-based Web languages [Francis et al. 1999, Pemberton et al. 2000, Raggett et al. 1998].

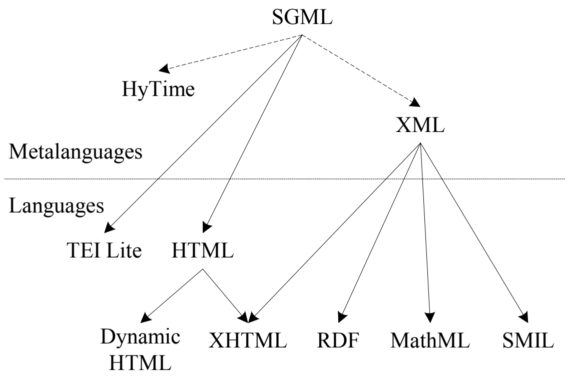


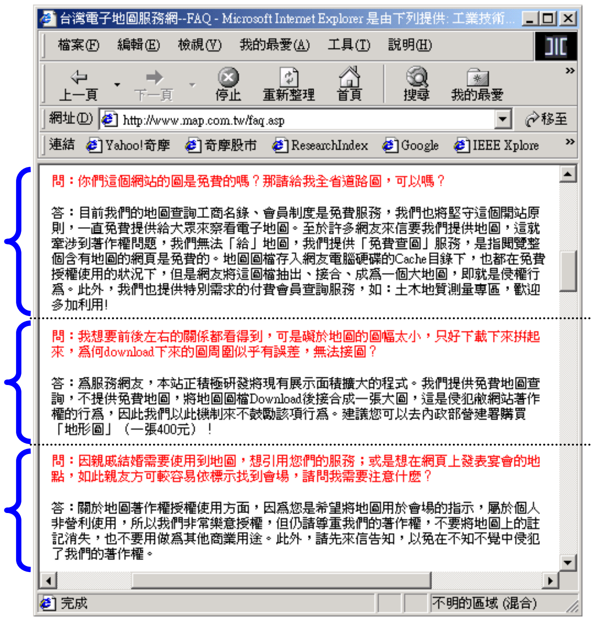
Fig. 1 Taxonomy of SGML-based Web languages.

1.2 Taxonomy of Web FAQs

Identifying the types of Web FAQs is conducive to tracing QA-pairs within a Website. We classify Web FAQs according to the two viewpoints : (A) the relational position between a question and its answer and (B) the types of the hyperlinks from the questions to the answers.

From the viewpoint of the relational position, Web FAQs can be divided into the following two types:

Type A.1 Side-by-Side QA-pairs – In an FAQ file, every question is immediately followed by its answer. Generally speaking, this type of questions equip no hyperlinks for their answers due to their closely relational position. The whole FAQ file looks like a list of QA-pairs. An example of this type is shown in Fig. 2.


台灣電子地圖服務網--FAQ - Microsoft Internet Explorer 是由下列提供: 工業技術...
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 下一頁 停止 重新整理 首頁 搜尋 我的最愛
網址(D) http://www.map.com.tw/faq.asp 移至
連結 Yahoo!奇摩 奇摩股市 ResearchIndex Google IEEE Xplore
完成
不明的區域 (混合)

問：你們這個網站的圖是免費的嗎？那請給我全省道路圖，可以嗎？

答：目前我們的地圖查詢工商名錄、會員制度是免費服務，我們也將堅守這個開站原則，一直免費提供給大眾來察看電子地圖。至於許多網友來信要我們提供地圖，這就牽涉到著作權問題，我們無法「給」地圖，我們提供「免費查圖」服務，是指閱覽整個含有地圖的網頁是免費的。地圖圖檔存入網友電腦硬碟的Cache目錄下，也都在免費授權使用的狀況下，但是網友將這圖檔抽出、接合、成為一個大地圖，即就是侵權行為。此外，我們也提供特別需求的付費會員查詢服務，如：土木地質測量專區，歡迎多加利用!

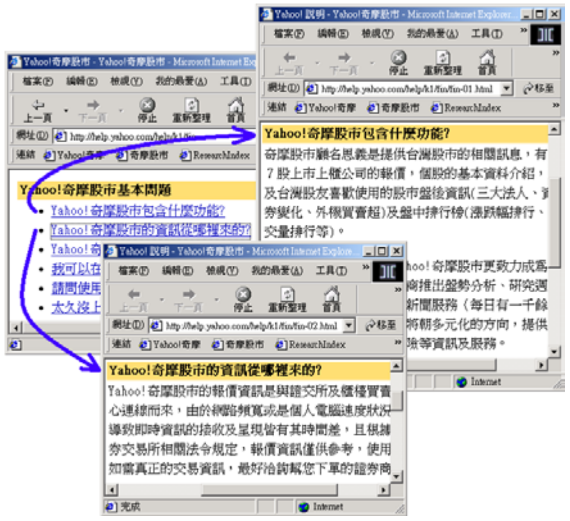
問：我想要前後左右的關係都看得到，可是礙於地圖的圖幅太小，只好下載下來拼起來，為何download下來的圖周圍似乎有誤差，無法接圖？

答：為服務網友，本站正積極研發將現有展示面積擴大的程式。我們提供免費地圖查詢，不提供免費地圖，將地圖圖檔Download後接合成一張大圖，這是侵犯敝網站著作權的行為，因此我們以此機制來不鼓勵該項行為。建議您可以去內政部營建署購買「地形圖」（一張400元）！

問：因親戚結婚需要使用到地圖，想引用您們的服務；或是想在網頁上發表宴會的地點，如此親友方可較容易依標示找到會場，請問我需要注意什麼？

答：關於地圖著作權授權使用方面，因為您是希望將地圖用於會場的指示，屬於個人非營利使用，所以我們非常樂意授權，但仍請尊重我們的著作權，不要將地圖上的註記消失，也不要用做為其他商業用途。此外，請先來信告知，以免在不知不覺中侵犯了我們的著作權。

Fig. 2 An example of the FAQ file consisting of side-by-side QA-pairs.


Yahoo!奇摩股市 - Yahoo!奇摩股市 - Microsoft Internet Ex
Yahoo! 說明 - Yahoo!奇摩股市 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T)
上一頁 下一頁 停止 重新整理 首頁
網址(D) http://help.yahoo.com/help/k1/fin/fin-01.html 移至
網址(D) http://help.yahoo.com/help/k1/fin/fin-02.html 移至
連結 Yahoo!奇摩 奇摩股市 ResearchIndex
完成
Internet

Yahoo!奇摩股市基本問題
- Yahoo!奇摩股市包含什麼功能?
- Yahoo!奇摩股市的資訊從哪裡來的?
- Yahoo!奇
- 我可以在
- 請問使用
- 太久沒上

Yahoo!奇摩股市包含什麼功能?

奇摩股市顧名思義是提供台灣股市的相關訊息，有7股上市上櫃公司的報價，個股的基本資料介紹，及台灣股友喜歡使用的股市盤後資訊(三大法人、資券變化、外機買賣超)及盤中排行榜(漲跌幅排行、交量排行等)。

hoo!奇摩股市更致力成為商推出盤勢分析、研究週新聞服務（每日有一千餘將朝多元化的方向，提供險等資訊及服務。

Yahoo!奇摩股市的資訊從哪裡來的?

Yahoo!奇摩股市的報價資訊是與證交所及櫃檯買賣心連線而來，由於網路頻寬或是個人電腦速度狀況導致即時資訊的接收及呈現皆有其時間差，且根據券交易所相關法令規定，報價資訊僅供參考，使用如需真正的交易資訊，最好洽詢幫您下單的證券商

Fig. 3 An example of the FAQ file consisting of hyperlinked QA-pairs.

Type A.2 Hyperlinked QA-pairs – In an FAQ file, all questions are organized as a list, but the answers do not follow their questions immediately. For users' convenience, in this type of FAQs, every question always have a hyperlink to the position of its answer, in which the hyperlink may direct to a position inside/outside the listing page containing the question list. Thus users can find the answers by simply clicking the hyperlinks. An example of this type is shown in Fig. 3.

From the viewpoint of the hyperlinks, we divide Web FAQs into the following four types:

Type B.1 Without Hyperlinks – The questions have no hyperlinks to their answers. In view of humanity design, it reasonably implies that the questions must be followed by their answers. In other words, this type is equivalent to Type A.1.

Type B.2 With Inside Hyperlinks – In an FAQ file, every question has an inside hyperlink to its answer. When a question has the hyperlink in the form of <A HREF=”#*label-name*”>, its answer must be the content of an <A NAME=“*label-name*”> tag, i.e. enclosed by tag pair <A NAME=”label-name”>... </A>.

Type B.3 With Single-Page Outside Hyperlinks – For a list of questions, all hyperlinks to their answers direct to a single outside page. The directed page usually contains a list of QA-pairs like Type A.1. That is, a replica of the questions also appears in the directed page, but probably in different forms.

Type B.4 With Multi-Page Outside Hyperlinks – For a list of questions, each of their answers is directed by a hyperlink to an outside page. The directed page usually contains a QA-pair alone, as shown in Fig. 3.

By the above analysis, we can locate an answer according to its question and the information attached. An important discovery and fact is that at least a portion of a Web FAQ, namely questions or QA-pairs, is always in the form of *list*. The ‘*list*’ indicates that the rendering looks like a list, but it does not mean it should be authored in some specific tags for authoring lists, such as <li> tag. This paper emphasizes the detection of visually list-like segments.

1.3 Domain Knowledge

Except for Web FAQs, some undesirable segments could be rendered as lists, such as the categories of portal sites, product categories, etc. To prevent the redundant segments from being extracted, we exploit some domain knowledge as follows:

(1) Chinese linguists traditionally divide questions into four types of **interrogative forms**: *question-word*, *disjunctive*, *tag*, and *particle questions* [Li and Thompson 1981]. By identifying the interrogative types for all questions in the possible FAQ files [Lai and Wu 2000], we can obtain a ratio of questions in the possible FAQ files. We decide it is an FAQ file if the ratio is greater than a threshold.

(2) In practice, an exceptional type – *declarative questions* is possibly appear. It occupied around 18% in a collection of 18034 QA-pairs. A punctuation mark – question mark ends some of them. **Question mark** is useful for identifying questions. Besides, some **prefix terms**, such as “Q,” and “問”, are also available.

To achieve better accuracy, some other domain knowledge is needed, but is not the emphasis of this paper.

2 List Detection

This section introduces how to detect lists from Web Pages. We will first define several important terms for our approach. Then, we will describe the approach to list detection via a markup language independent algorithm.

2.1 Terms for Our Approach

In the following we will introduce several important terms we defined for our approach, including *tag-content pair string*, *regular tag-content string*, *tag string*, *di-tag*, and *markup parse tree*.

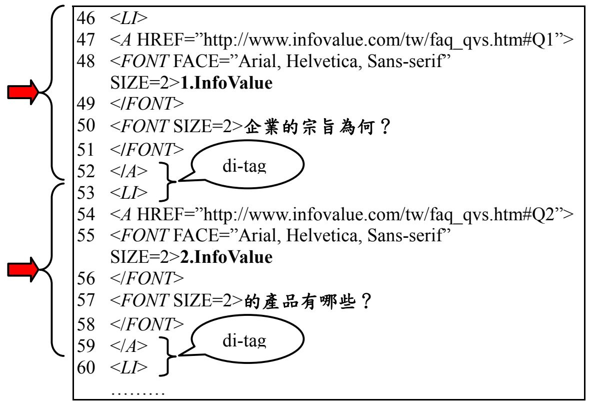
◆ ***Tag-Content pair String***

Without losing the correctness of rendering, each markup language document can be transformed into a list of tag-content pairs, TC-pairs for short, and the content is allowed being empty. Fig. 4 shows the partial source code of a markup language document extracted from http://www.infovalue.com/tw/faq_qvs.htm.

Fig. 5 lists an example source code transformed from the source code in Fig. 4. In Fig. 5, there is a TC-pair in line 50, in which its tag is <FONT SIZE=2> and its content is “企業的宗旨爲何？”. Besides, line 46 (empty tag) and line 47 (start tag) list two TC-pairs with empty content.

```
<LI>
<A HREF="http://www.infovalue.com/tw/faq_qvs.htm#Q1">
    <FONT FACE="Arial, Helvetica, Sans-serif"
SIZE=2>1.InfoValue</FONT>
    <FONT SIZE=2>企業的宗旨為何？</FONT>
</A>
<LI>
<A HREF="http://www.infovalue.com/tw/faq_qvs.htm#Q2">
    <FONT FACE="Arial, Helvetica, Sans-serif"
SIZE=2>2.InfoValue</FONT>
    <FONT SIZE=2>的產品有哪些？</FONT>
</A>
<LI>
………
```

Fig. 4 Partial source code from an example markup language document.



```
46  <LI>
47  <A HREF="http://www.infovalue.com/tw/faq_qvs.htm#Q1">
48  <FONT FACE="Arial, Helvetica, Sans-serif"
    SIZE=2>1.InfoValue
49  </FONT>
50  <FONT SIZE=2>企業的宗旨為何？
51  </FONT>
52  </A>
53  <LI>
54  <A HREF="http://www.infovalue.com/tw/faq_qvs.htm#Q2">
55  <FONT FACE="Arial, Helvetica, Sans-serif"
    SIZE=2>2.InfoValue
56  </FONT>
57  <FONT SIZE=2>的產品有哪些？
58  </FONT>
59  </A>
60  <LI>
    ………
```

Fig. 5 The source code in the form of TC-pairs transformed from Fig. 4. The italics indicate the element names and the bolded words are the contents.

◆ *Regular TC-pair String*

To precisely detect the position of a list in a markup language document, we define a Regular TC-pair String (RTCS) to be a string of TC-pairs that meets the regular constraint: "*All the tags in the Regular TC-pair String must be properly nested and an end tag closes all omitted start tags up to the matching start tag.*" The term "regular" comes from "regular expressions" formalism stated in Automata Theory [Hopcroft and Ullman 1979]. Meeting the regular constraint implies that:

Lemma 1. An RTCS must start with a start-tag and end at an end-tag, in which the end-tag need not match the start tag.

Lemma 2. In an RTCS, all the unclosed tags before the content of a TC-pair influence the rendering of the content.

The regular language L_R, which generates a set of RTCSs, can be described by regular expressions as follows:

$$L_R = \{x^* \mid x = ay(\tilde{a}+\varepsilon), a, \tilde{a} \in \Sigma, y \in L_R\} \qquad (1)$$

where Σ denotes a finite set of symbols, i.e. TC-pairs in this paper, a and $\tilde{a}$ are symbols in Σ, in which a denotes a TC-pair consisting of a start or empty tag and a content, notated as $a=(t_i,c_i)$, and $\tilde{a}$ denotes a TC-pair consisting of an end tag corresponding to a's and a content, notated as $\tilde{a}=(\tilde{t}_i,c_j)$, and ε denotes a null and stands for that a is a TC-pair with an empty tag or $\tilde{a}$ is omitted.

By Lemma 1, an RTCS may contain one or more sub-RTCSs. Therefore an elementary RTCS is well-defined as an RTCS that is wrapped in a start-tag and its matching end-tag, in which the start-tag and its matching end-tag is called a representative of the elementary RTCS. That is, the final tag must be an end-tag and match the first tag, or the matching end-tag of the first tag is omitted if the final tag does not match the first tag. A single empty tag can be regarded as an elementary RTCS.

While authoring Web Pages, one may intentionally/unintentionally ommit some tags. By Lemma 2, the ommited tags can be identified and recovered.

◆ *Tag String and Di-tag*

For a TC-pair string, after removing their content parts, the remainders are called *tag string*. For two consecutive tags, i.e. a tag string consisting of two tags, we refer to it as *di-tag*.

◆ *Markup Parse Tree*

The two arrowed regions in Fig. 5, lines 46~52 and lines 53~59, are two RTCSs. There are two differences between them. (1) Their values of the HREF attributes in the two <A> tags (lines 47 and 54) are different. Since the two values are hyperlinks to their corresponding answers, they are certainly different. (2) Their individual contents (the bolded texts) represent two different questions. Except for the differences, the other portions of these two strings are identical.

However, not all lists are so regular especially in those containing answers. Every two answers in an FAQ file are not only different in the aspects of the textual content but also in other aspects, such as the style, the font, etc., which relate to the interpretation of markups. For example, Fig. 6 shows the rendering of two QA-pairs corresponding to the two answers authored by the source code in Fig. 5. Comparing the two QA-pairs in Fig. 6, they are apparently dissimilar in the number of the paragraphs, the styles, and the fonts. Their own source codes are almost incomparable.

As mentioned before, there are always hyperlinks between the question list and their corresponding answers for Type A.2. No doubt, the answers can be detected by the hyperlinks. However, there are no such relationships for Type A.1. To solve this problem, we propose a tree-matching-based method. Summarily, two segments of markups to be compared are first parsed into two parse trees, called *markup parse trees* (MPTs).

Transforming an RTCS into an MPT is described more detailly below.

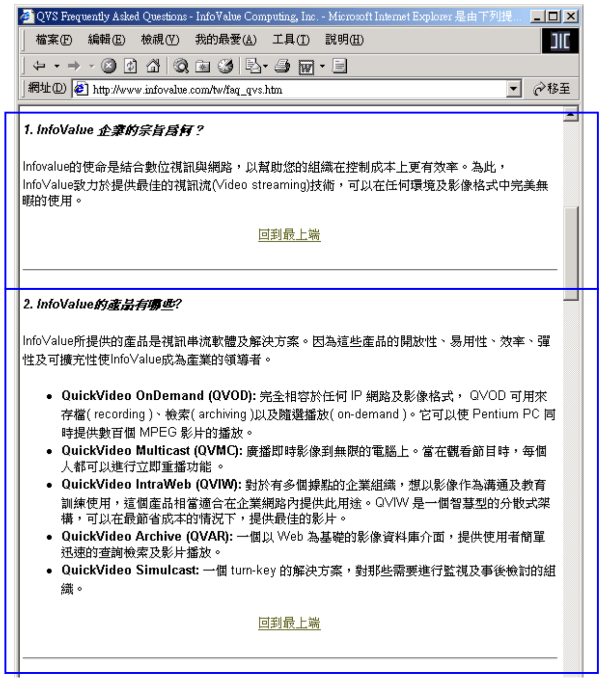
(1) For an RTCS, it is transformed into a tag string by purging the content from each TC-pair, in which the tag string is still regular since this transformation does not change its structure. It is also nested.

(2) According to the nested structure of the tag string, we construct a parse tree, namely the MPT. The construction obeys the following principles :

   i. An MPT owns a unique vertex, called root, which virtually stands for the whole tag string.
   ii. Each sub-tree in an MPT is constructed from an elementary RTCS and the representative of the elementary RTCS labels the root of the sub-tree.
   iii. Each external vertex is constructed from an elementary RTCS consisting of either one pair of matching tags or a single empty tag.
   iv. Suppose one pair of matching tags wraps another, its corresponding vertex is an ancestor of another.
   v. Suppose two pairs of matching tags are side by side, their corresponding vertices are siblings in a fixed order.
   vi. A sub-tree of an MPT is still an MPT.

2.2 What is a list?

Before introducing how to detect lists, let us find out what a list is. Visually, a list consists of two kinds of components, items and intervals. Each two consecutive items are separated by an interval. Our key idea is to detect the intervals and then compare the consecutive items separated by the intervals. If we can find a sequence of *similar items* separated with *the same interval*, it is a list, in which the items are actually RTCSs.


QVS Frequently Asked Questions - InfoValue Computing, Inc. - Microsoft Internet Explorer 是由下列提...
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
網址(D) http://www.infovalue.com/tw/faq_qvs.htm
移至

1. InfoValue 企業的宗旨爲何？

Infovalue的使命是結合數位視訊與網路，以幫助您的組織在控制成本上更有效率。為此，InfoValue致力於提供最佳的視訊流(Video streaming)技術，可以在任何環境及影像格式中完美無暇的使用。

回到最上端

2. InfoValue的產品有哪些?

InfoValue所提供的產品是視訊串流軟體及解決方案。因為這些產品的開放性、易用性、效率、彈性及可擴充性使InfoValue成為產業的領導者。

- **QuickVideo OnDemand (QVOD):** 完全相容於任何 IP 網路及影像格式， QVOD 可用來存檔(recording)、檢索(archiving)以及隨選播放(on-demand)。它可以使 Pentium PC 同時提供數百個 MPEG 影片的播放。
- **QuickVideo Multicast (QVMC):** 廣播即時影像到無限的電腦上。當在觀看節目時，每個人都可以進行立即重播功能 。
- **QuickVideo IntraWeb (QVIW):** 對於有多個據點的企業組織，想以影像作為溝通及教育訓練使用，這個產品相當適合在企業網路內提供此用途。QVIW 是一個智慧型的分散式架構，可以在最節省成本的情況下，提供最佳的影片。
- **QuickVideo Archive (QVAR):** 一個以 Web 為基礎的影像資料庫介面，提供使用者簡單迅速的查詢檢索及影片播放。
- **QuickVideo Simulcast:** 一個 turn-key 的解決方案，對那些需要進行監視及事後檢討的組織。

回到最上端

Fig. 6 The rendering of the QA-pairs respectively corresponding to the questions embedded in Fig. 5.

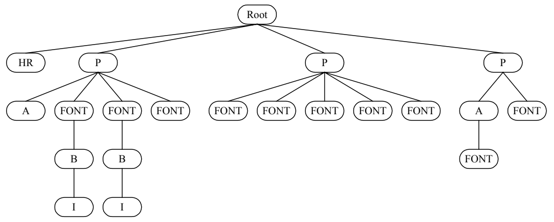


Fig. 7 An MPT constructed from the first RTCS, i.e. the part in the first block, in Fig. 6.

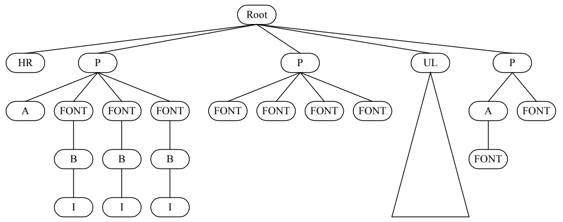


Fig. 8 An MPT constructed from the second RTCS, i.e. the part in the second block, in Fig. 6.

2.3 Measurement of the Similarity between two RTCSs

For the term "similar" mentioned above, we need a measurement for computing the similarity between two RTCSs. We can measure their similarity by comparing their own MPTs even though their RTCSs cannot be compared directly. Figs. 7 and 8 illustrate two similar MPTs that are transformed from two RTCSs of the rendering shown in Fig. 6, namely the two blocks.

For each two MPTs T_1 and T_2, we measure their similarity as follows:

$$Sim_{MPT}(T_1,T_2)=\begin{cases}0, \text{ if } root(T_1)\neq root(T_2) \\ 1, \text{ if } root(T_1)=root(T_2), \text{ and} \\ \quad \text{both } T_1 \text{ and } T_2 \text{ are single-vertex trees} \\ \delta_n+(1-\delta_n)\cdot Sim_{sub-MPT}(T_1,T_2), \text{ elsewise}\end{cases} \quad (2)$$

where $root()$ returns the root tag of an MPT, δ_n denotes a relational weight of a parent to its children at level n, and $Sim_{sub-MPT}()$ is a function of estimating the similarity between the two MPTs based on their sub-trees' similarities, which is defined as follows:

$$Sim_{sub-MPT}(T_1,T_2)=\frac{|T_A|}{|T_B|}\max_g\sum_{k=1}^{|T_A|}Sim_{MPT}(T_{A,k},g(T_{A,k})) \quad (3)$$

where T_A is one of the two MPTs T_1 and T_2 such that it subsums fewer sub-trees, T_B is another one, and g denotes a one-to-one function from T_A to T_B.

2.4 List Detection Algorithm

The algorithm for detecting lists from a hypertext is briefly described as follows:

Step 1. Transform the hypertext into an RTCS.

Step 2. Transform the RTCS into a tag string.

Step 3. Find all the di-tags in the form of end-start tags. The positions the di-tags located are possible intervals. For example, Fig. 5 shows two identical di-tags at lines 52-53 and 59-60. In this case, they are intervals.

Step 4. Cluster all the di-tags. That is, the identical di-tags with different positions are clustered together.

Step 5. For each cluster, sort its di-tags. Thus, the TC-pair string between each two di-tags in a cluster is a possible item.

Step 6. Find all possible lists in each cluster by concatenating the consecutive items whose similarities are greater than a threshold. Note that the items to be concatenated must be RTCSs.

Step 7. Some of the possible lists maybe overlap. For a group of overlapped lists, we choose the one with the highest cumulative similarity as an output.

This algorithm is able to detect lists embedded in the documents authored in markup languages. It utilizes the characteristic of the items in a list being structurally similar. It is independent of the tags specifically used for authoring lists. That is, it is markup language independent.

3 Experimental Results

A Spider [Cho et al. 1998, Introna and Nissenbaum 2000] was constructed to automatically collect 14 categories of Websites. Each category contains 100 Websites. 14 people, who did not take part in the core task, were asked to label the Websites. For each Website, they manually label the number of the FAQ files and their located pages. After removing the Websites authored in the languages other than Traditional Chinese, 30,007 pages in 901 Websites are retained, in which there are 293 FAQ files in 76 Websites.

Table 1 Experimental results of FAQ mining from 9 categories of Websites consisting of 30,007 pages in 901 Websites.

Categories	*FAQ Files*	*True Mining*	*False Mining*	*Recall Rate (%)*	*Precision Rate (%)*
Medium	22	12	13	54.55	48.00
Leisure	6	4	0	66.67	100.00
Region	8	3	1	37.50	75.00
Society	22	16	21	72.73	43.24
Politics	32	29	9	90.63	76.32
Science	30	15	5	50.00	75.00
Computer	76	72	1	94.74	98.63
Network	38	35	0	92.11	100.00
Medicine	50	20	1	40.00	95.24
Total	284	206	51	72.54	80.16

Table 1 shows the experimental results, in which we remove 5 categories with fewer FAQ files. We evaluate the performance in recall rate, precision rate, and accuracy rate. Summarily, the system achieved 72.54% recall rate and 80.16% precision rate. Since most of the Websites contain no FAQ files, the average accuracy is 99.42%. It means almost all the redundant lists can be discarded correctly.

By error analysis, several main errors are as follows:

(1) Many news articles are entitled in inciting questions. It makes some false mining.

(2) Some Websites place FAQ files nearby in one page. The nearby FAQ files are recognized as one FAQ file.

(3) The Spider missed scratching some pages containing FAQ files.

Conclusion

List is an efficient way to represent information. Many kinds of lists are widely used on the Internet, especially for those we are interested – FAQ. By analysing the Web FAQs, we found a fact that FAQs are always fully/partially authored in the form of list. This paper presents an algorithm to detect lists embedded in Web Pages. The algorithm is independent of the tags specifically used for authoring lists. That is, it is markup language independent. By combining the algorithm with some constraints of domain knowledge, we can automatically detect and collect FAQs from the Web. The mining task achieved a performance of 72.54% recall rate and 80.16% precision rate.

Acknowledgements

This paper is a partial result of Project A311XS1211 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, R.O.C.

References

Baeza-Yates R. and Ribeiro-Neto B. (1999) *Modern Information Retrieval*, 1st ed., New York: ACM Press, chap. 6, pp. 149-162.

Bray T., Paoli J., Sperberg-McQueen C. M. and Maler E. (2000) *eXtensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation.

Cho J., Garcia-Molina H. and Page L. (1998) *Efficient Crawling through URL ordering*, In Proc. 7th Intl. World Wide Web Conference (WWW 98), Brisbane, Australia, April 14-18, pp. 161-172.

Francis B., Homer A. and Ullman C. (1999) *IE 5 Dynamic HTML Programmer's Reference*, 1st ed., Wrox Press Ltd.

Goldfarb C. (1990) *The SGML Handbook*, Oxford University Press, Oxford.

Hopcroft J. E. and Ullman J. D. (1979) *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, chap. 6, pp. 13-54.

Introna L. and Nissenbaum H. (2000) *Defining the Web: The Politics of Search Engines*, Computer, vol. 33, pp. 54-62.

ISO 8613 (1989) *Office Document Architecture (ODA) and Interchange Format.*

ISO 8879 (1986) *Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML).*

Lai Y. S. and Wu C. H. (2000) *Intention Extraction and Semantic Matching for Internet FAQ Retrieval Using Spoken Language Query*, In Proc. 6th Intl. Conference on Spoken Language Processing.

Li C. N. and Thompson S. A. (1981) *Mandarin Chinese: A Functional Reference Grammar*, University of California Press, pp. 520-563.

Pemberton S., Altheim M., Austin D., Boumphrey F., Burger J., Donoho A. W., Dooley S., Hofrichter K., Hoschka P., Ishikawa M., Kate W. ten, King P., Klante P., Matsui S., McCarron S., Navarro A., Nies Z., Raggett D., Schmitz P., Schnitzenbaumer S., Stark P., Wilson C., Wugofski T. and Zigmond D. (2000) *XHTML^{TM} 1.0: The Extensible HyperText Markup Language*, W3C Recommendation.

Raggett D., Hors A. L. and Jacobs I. (1998) *HTML 4.0 Specification*, W3C Recommendation, pp. 24-26, April 1998.