Element-based Retrieval@MobileClick-2

Atsushi Keyaki
Tokyo Institute of Technology
keyaki@lsc.cs.titech.ac.jp

Jun Miyazaki
Tokyo Institute of Technology
miyazaki@cs.titech.ac.jp

Kenji Hatano
Doshisha University
khatano@mail.doshisha.ac.jp

ABSTRACT

In this paper, we report our effort of tackling MobileClick-2 with the element-based retrieval approach. The goal of element-based retrieval is to identify only relevant descriptions to a query and show them to a user. We believe this is essentially similar to that of MobileClick-2, which is why we employed element-based retrieval approach to this task. The problem is that the output unit of element-based retrieval is *element* whereas that of MobileClick-2 is *iUnit*. Thus, we calculate an iUnit score from a similarity with highly ranked elements. The results of formal runs showed that our system attained intermediate and the best accuracy in iUnit Ranking Subtask and iUnit Summarization Subtask, respectively.

Team Name

TITEC

Subtasks

iUnit Ranking Subtask (English)
iUnit Summarization Subtask (English)

Keywords

element-based retrieval, iUnit, information extraction

1. INTRODUCTION

In this paper, we report our effort of tackling MobileClick-2[7] with the element-based retrieval approach. The goal of element-based retrieval is to identify only relevant descriptions to a query and show them to a user. We believe this is essentially similar to that of MobileClick-2, which is why we employed element-based retrieval approach to this task.

Element in the element-based retrieval is an element in structured documents such as XML documents and HTML documents. Concretely, an element is a text between corresponding tags. In the element-based retrieval, an element contained all and only relevant descriptions is expected to be retrieved.

In our participation of 1CLICK-2 at NTCIR-10, we found that search accuracy of element-based retrieval differs by query type. Search accuracy of some types of queries such as *DEFINITION* and *QA* is higher than other types of queries [2]. These types of queries aim to return document centric output rather than data centric output that other types of queries aim to return. It suggests that important descriptions in HTML documents are successfully extracted with element-based retrieval. However, our system cannot compute post-processing, namely, extracting iUnits from a relevant element.

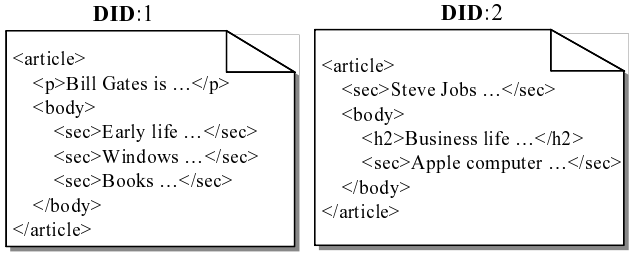

Figure 1: XML document

On the other hand, extracting iUnits from an element is out of scope with MobileClick-2 because iUnits are provided by task organizers. Participants are expected to focus on ranking and summarizing provided iUnits. Briefly and intuitively, we arrange iUnits highly similar to element with higher score with element-based retrieval at high rank for iUnit Ranking Subtask, meanwhile, we basically take the baseline approach of MobileClick-1 [4] for iUnit Summarization Subtask.

As a result of formal runs, our group attain decent intermediate accuracy for iUnit Ranking Subtask and achieved the best accuracy for iUnit Summarization Subtask.

We show some basic concepts of element-based retrieval in Section 2 followed by related studies in Section 3. Then, we move to our challenges to iUnit Ranking Subtask and iUnit Summarization Subtask in Sections 4 and 5, respectively. Next, evaluation results are discussed in Section 6. Finally, we conclude this paper in Section 7.

2. OVERVIEW OF ELEMENT-BASED RETRIEVAL

We show concrete examples in Figures 1, 2, and 3 to explain the definition of (XML) elements. Figure 1 illustrates an example of XML document. Figure 2 depicts a tree that is translated from Figure 1. An XML document can be expressed as a tree, which helps to understand the structure of the document.

The author of a document makes structures (e.g. chapters, sections, paragraphs.) We utilize these structures to

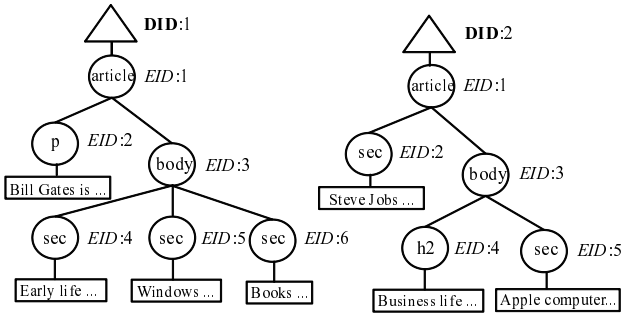

Figure 2: XML tree

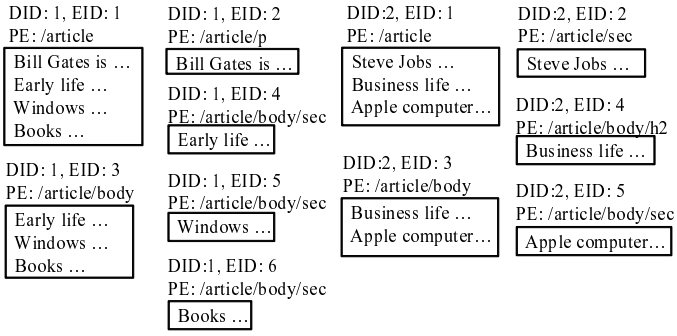

Figure 3: XML element

identify the best description for satisfying the users' information needs. In other words, we suppose that texts in an item are based on the same theme. Then, we extract the item that discusses the theme of the users' interest.

A pair of start and end tags represents an XML element node in an XML tree, and the nested structure of XML elements represents an ancestor-descendant relationship. Each element in Figure 3 is the text that is composed of a set of text nodes in the XML tree in Figure 2. This demonstrates why there are overlapping XML elements in XML documents.

Suppose a user seeks information about "Early life . . . ", "Windows . . . " and "Books · · · ". XML element retrieval systems try to present an element whose root node is `body` to the user because the element contains all of the information that the user needs and no further information.

3. RELATED STUDIES

In article [10], it is reported that tags in structured documents are largely classified into two groups; A) tags surrounding self-contained content and B) tags enabling separate content. In this paper, we define tags of A) as structural tags. Concrete examples of structural tags are `HEAD`, `BODY`, and `P` tags of HTML. These tags are quite commonly used and can be meaningful clues for identifying useful and appropriate granular elements.

In addition, some HTML tags defined in HTML5 [3] such as `ARTICLE`, `SECTION`, `NAV`, and `ASIDE` tags are also structural tags. A `SECTION` tag can be nested, and each of the other tags represents specific or semantic context. This means

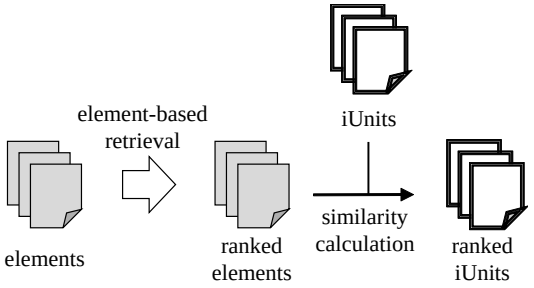

Figure 4: The process for ranking iUnits

that a physical document structure agreeing to a logical structure of the document can be generated with these tags. However, there is a possibility that we cannot utilize these newly defined tags because these tags are not widely used yet.

In contrast, tags of B) are used for representing decoration, attribute, and specific idea. To enumerate some examples, `B`, `FONT`, `I` tags are applicable. Most of HTML tags are classified into B), because HTML is defined for the sake of being used for displaying with a browser. Compared with tags of B), the number of tags of A) is small.

There are some kinds of tags which perform a boundary between one topic and another [5], for example, Heading tags (`H1`–`H6` tags), `HR` tag, and `BR` tag. These tags are leveraged to split content according to a topic. Our survey [2] found that it is true that these tags are helpful to identify a boundary, however, these does not always indicate the scope of each topic.

4. IUNIT RANKING SUBTASK

In the iUnit Ranking Subtask, iUnits provided by the task organizers are expected to be ranked according to their importance. We introduce our approach with Figure 4.

First we extract elements from provided html documents. Next, all elements are scored with an arbitrary element-based term weighting scheme. Consequently, ranked elements are obtained. Then, similarities between elements and iUnits are calculated to rank iUnits. In more detail, the number of terms occurred both in each element and each iUnit is counted. We suppose that an iUnit similar to highly ranked elements should be scored higher. Thus, the score of iUnit u is calculated as follows:

$$Score(u) = \sum_{e \in E} \frac{count(u, e)}{rank(e)} \quad (1)$$

Where let E be an element set, e be an element in E, $count(u, e)$ be the number of term co-occurred in u and e, and $rank(e)$ be the rank of e.

5. IUNIT SUMMARIZATION SUBTASK

In iUnit Summarization Subtask, we basically employ the baseline method of MobileClick-1 [4] besides using element-based retrieval for iUnit ranking. One more our effort is to rank intents according to their importance.

The process of generating search results for iUnit Summarization Subtask is as follows:

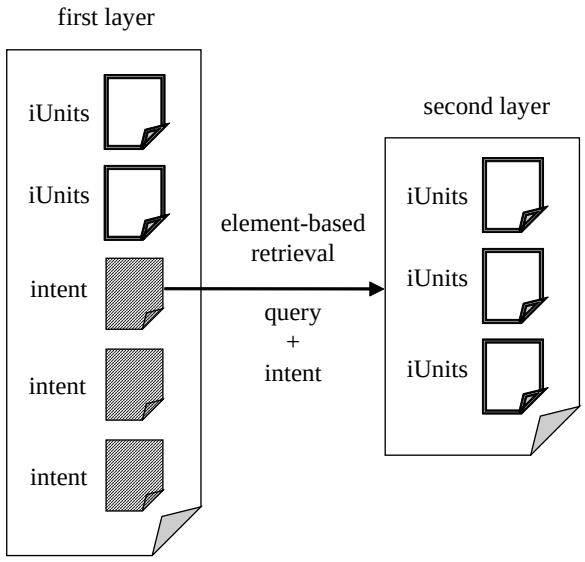

Figure 5: The process for iUnit Summarization

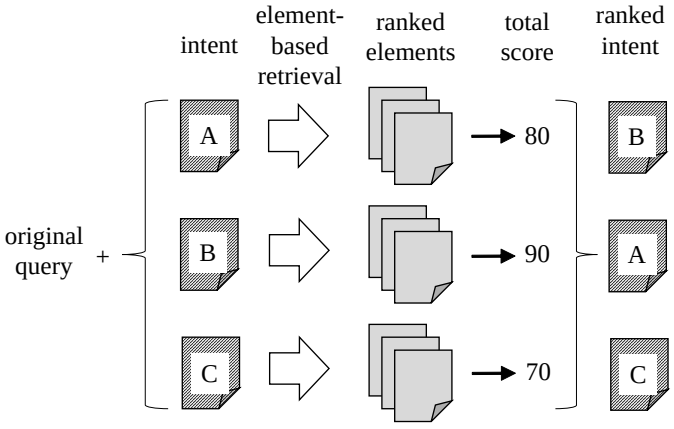

Figure 6: The process for ranking intents

1. ranking intents,
2. allocating intents in the first layer,
3. ranking iUnits with a original query and all intents,
4. allocating iUnits obtained in 3. in the first layer,
5. ranking iUnits with a original query and each intent, and
6. allocating iUnits in the second layer using the results of 5.

The process is illustrated in Figure 5.

For ranking intents, we first conduct element-based retrieval with an original query and each intent as a new query. As a result, ranked elements are gained. We define the score of each intent is sum of these elements' score. This is based on a hypothesis that an intent has more information if sum of elements' score is larger. Accordingly, intents are ranked.

We demonstrate a tangible example with Figure 6. We issue three queries, i.e., {original query + intent A}, {original query + intent B}, {original query + intent C}. Ranked elements are obtained for each query. The total score of ranked elements of {original query + intent A} is 80, which become the score of intent A. Finally, intents are ranked in the order of B, A, C.

After intents are ranked, these are allocated in the first layer according to their ranks. Next, iUnits are allocated if there is vacant for output text. Note that a query for the iUnits differs from the one for iUnit Ranking Subtask. The query is concatenation of an original query and all intents. We suppose that we can allocate more informative iUnits in the first layer.

Subsequently, we build second layers. Each second layer comprises iUnits ranked with an original query and each intent as Figure 5 indicates.

6. EVALUATION

6.1 Experimental Settings

The following preprocesses are applied before elements are extracted from the HTML documents provided by task organizers.

1. removing attributes, comments, and special characters of HTML documents,
2. removing the stop words by SMART stop list [9],
3. applying stemming step by Porter [8], and
4. validating corresponding relations in tags with CyberNeko HTML Parser [1].

Note that we only use `BODY` and `P` tags for elements for simplicity.

We calculate a relevancy score of each element with BM25E [6] which is one of the most popular term weighting schemes for element retrieval. Note that we adopt the tag-based approach for global weight calculation, because the number of documents is not enough for calculating accurate global weights with the path expression-based approach.

The PC that we used for the experiments runs Oracle Enterprise Linux 5.5. It has four Intel Xeon X7560 CPUs (2.3GHz), 512GB of memory, and a 4.5TB disk array. The indices were implemented using BerkeleyDB in GNU C++.

6.2 iUnit Ranking Subtask

The results of the formal runs for iUnit Ranking Subtask are shown in Figure 7. Accuracy of our system (*TITEC*) is 0.9003, which is 12nd out of 25 runs (6th out of 11 groups).

Note that we did not utilize intents in the formal run of this subtask. Then, accuracy slightly improves and becomes 0.9012 when intents are added into an original query in the element-based retrieval step.

6.3 iUnit Summarization Subtask

Similarly, results of the formal runs for iUnit Ranking Subtask are shown in Figure 8. Our system (*TITEC*) achieved 18.2596, which is the best out of 16 runs (the best out of 11 groups). The fact that our system performed good results in iUnit Summarization Subtask although accuracy of iUnit Ranking Subtask is intermediate suggests that ranking of

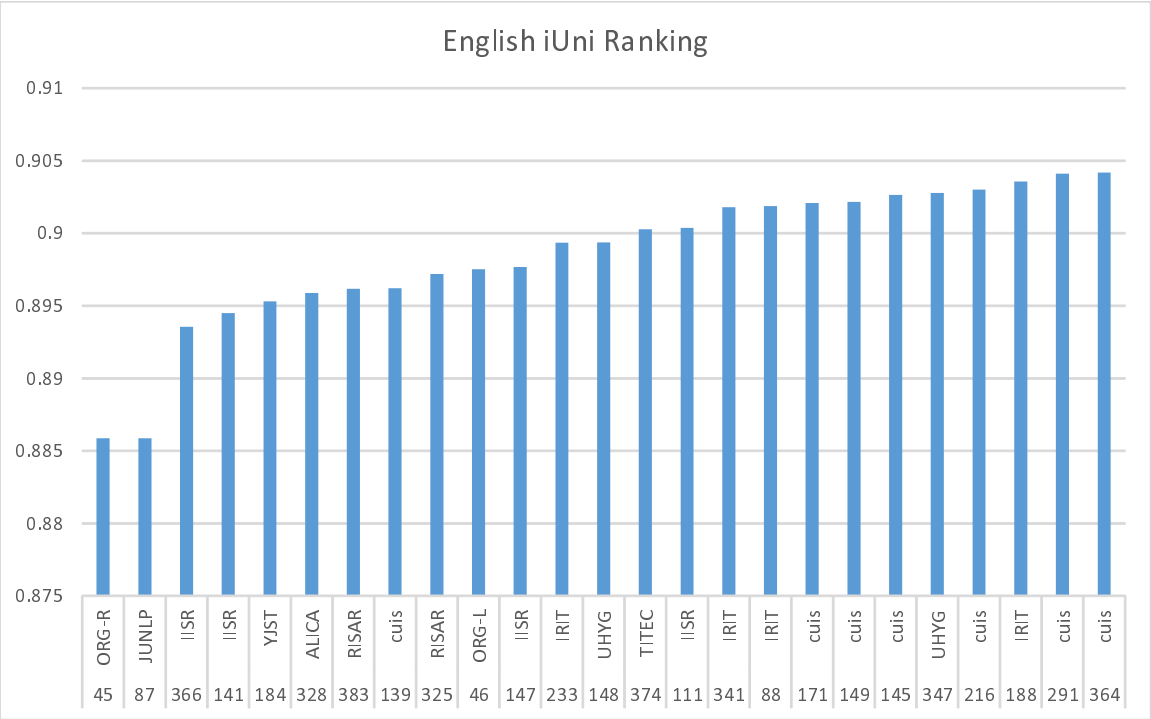

Figure 7: The result of iUnit Ranking Subtask

intents may affect the results. We need to investigate more deeply as a part of our future work.

Moreover, our system has a possibility to be improved with using better iUnit ranking methods of other groups. This is also our future work.

7. CONCLUSION

In this paper, we report our effort on element-based retrieval at MobileClick-2. As a result of formal runs, we attained intermediate and the best accuracy for iUnit Ranking Subtask and iUnit Summarization Subtask, respectively.

[Acknowledgments]

This work was partly supported by JSPS KAKENHI Grant 15K20990.

8. REFERENCES

[1] M. G. Andy Clark. Cyberneko html parser (1.9.19). `http://nekohtml.sourceforge.net/index.html`, 2013. Accessed in November, 2013.

[2] K. H. G. Y. T. T. Atsushi Keyaki, Jun Miyazaki and H. Kato. XML Element Retrieval@1CLICK-2. In *Proc. of the 12th NTCIR Conference*, 2013.

[3] R. Berjon, S. Faulkner, T. Leithead, E. D. Navara, E. O'Connor, S. Pfeiffer, and I. Hickson. HTML5. `http://www.w3.org/TR/html5/`, 2014. Accessed in November, 2013.

[4] M. P. Kato, M. Ekstrand-Abueg, V. P. T. Sakai, T. Yamamoto, and M. Iwata. Overview of the NTCIR-11 MobileClick Task. In *Proc. of the 11th NTCIR Conference*, 2014.

[5] S.-J. Lim and Y.-K. Ng. Converting the Syntactic Structures of Hierarchical Data to Their Semantic Structures. *Information Organization and Databases*, 579:343–355, 2000.

[6] W. Liu, S. Robertson, and A. Macfarlane. Field-Weighted XML Retrieval Based on BM25. In *Formal Proc. of INEX 2005 Workshop*, volume 3977 of *LNCS*, 2006.

[7] T. Y. V. P. H. M. Makoto P. Kato, Tetsuya Sakai and S. Fujita. Overview of the NTCIR-12 MobileClick Task. In *Proc. of the 12th NTCIR Conference*, 2016.

[8] M.F.Porter. An Algorithm for Suffix Stripping. In *Computer Laboratory, Cambridge*, 1980.

[9] G. Salton. *The SMART Retrieval System–Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[10] T. Tokuda and K. Tajima. Classification of XML Tags according to Their Roles in Document Structure. *DBSJ Journal*, 8(1):1–6, 2009. (in Japanese).

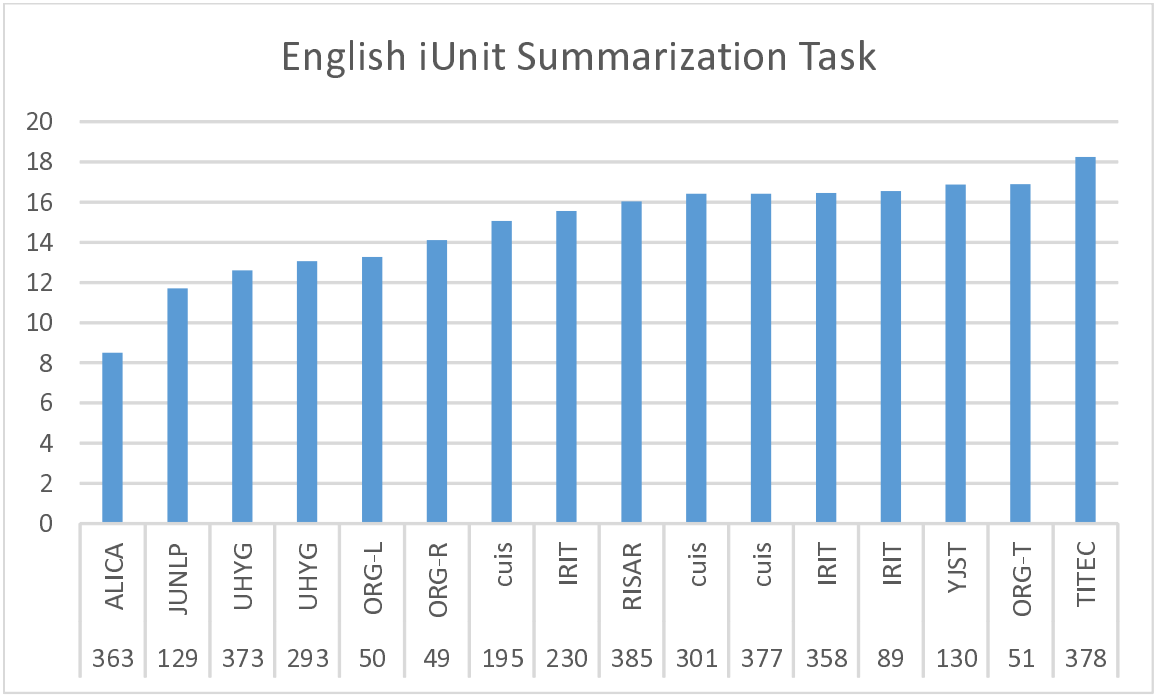

Figure 8: The result of iUnit Summarization