Information Extraction based Approach for the NTCIR-10 1CLICK-2 Task

Tomohiro Manabe†, Kosetsu Tsukuda†, Kazutoshi Umemoto†, Yoshiyuki Shoji†, Makoto P. Kato, Takehiro Yamamoto, Meng Zhao, Soungwoong Yoon, Hiroaki Ohshima, Katsumi Tanaka
Department of Social Informatics, Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto 606-8501, Japan
† Research Fellow of Japan Society for the Promotion of Science
{ manabe, tsukuda, umemoto, shoji, kato, tyamamot, zhao, yoon, ohshima, tanaka }
@dl.kuis.kyoto-u.ac.jp

ABSTRACT
We describe a framework incorporating several information extraction methods for the NTCIR-10 One Click Access Task. Our framework first classifies a given query into pre-defined query types, and then extracts sentences and key-value pairs which seem relevant to the query. At this time, our framework adjusts the weights of the extraction methods depending on the judged class of the query. Finally, our framework summarizes ranked sentences and key-value pairs considering diversity.

Keywords
Information extraction, diversification, query classification

Team Name
KUIDL

Subtasks
Main task (Japanese, English)
Query Classification Subtask (Japanese, English)

1. INTRODUCTION
Kyoto University, Department of Informatics, Digital Library laboratory (KUIDL) participated in the NTCIR-10 One Click Access (1CLICK) task. 1CLICK refers to a task that aims to satisfy the user with a single textual output, immediately after the user clicks on the SEARCH button [1]. In this task, the system is expected to present important pieces of information first, which are different for different types of queries. To tackle these problems, we incorporated different information extraction (IE) techniques for each type of queries. We propose a general framework for the 1CLICK task, which first classifies a given query into pre-defined query classes, then extracts information from the prepared document collection by using extraction methods, and finally aggregates pieces of information into a short text.

2. FRAMEWORK
In this section, we describe our framework that consists of *query classifier*, *information extractor*, the *weighting function for extractors*, and *information summarizer*. The implementation except for information extractors is then introduced right after the description of our framework, and that of information extractors is described in the following section.

Our framework is depicted in Figure 1. A query classifier first classifies a given query into pre-defined eight query types, i.e. ARTIST, ACTOR, POLITICIAN, ATHLETE, FACILITY, GEO, DEFINITION and QA. All queries are processed by all information extractors and query classification is utilized for weighting the extractors. The information extractors retrieve information pieces, which are sentences and key-value pairs, from the prepared document collection, and compute the importance score of each piece. The system then passes those pieces to an information summarizer for ranking them by considering the redundancy of information. Finally, the framework outputs ranked information pieces, which are shortened to fit either DESKTOP run (500 characters for Japanese and 1000 characters for English) or MOBILE run (140 characters for Japanese and 280 for English). Note that we used the same algorithms for generating our DESKTOP and MOBILE runs.

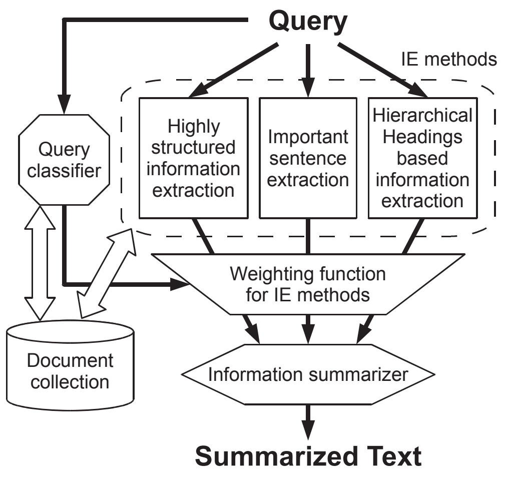
Query
IE methods
Query classifier
Highly structured information extraction
Important sentence extraction
Hierarchical Headings based information extraction
Weighting function for IE methods
Document collection
Information summarizer
Summarized Text

Figure 1. Our framework.

2.1 Query Classifier
First of all, we classified a query into eight types by using a multi-class support vector machine, where we incorporated 126 types of features (see Table 1) for Japanese queries, and 42 types of features (see Table 2) for English queries. Each feature is explained bellow:

- **Query length**: It contains two features. The first one is a binary feature that indicates whether the length of query is long or not. When terms occurred in the query are more then 10, this feature takes 1. Another one is the number of terms included in the query.
- **In dictionary:** We prepared 10 dictionaries such as GEO dictionary made from Wikipedia, zip code list taken from government site, musicians' list taken from record company sites and so on. When the query included in the dictionary,

Table 1. Features for Japanese Queries

Feature	# of features	Only on Open Runs
Query length	2	
In dictionary	10	
Frequency of parts-of-speech	9	
Query unigram	3	
Sentence pattern	2	
# of documents containing expanded query	19	*
# of search results	1	*
Terms in search results	36	*
Sites in search results	44	*
Total	126	

Table 2. Features for English Queries

Feature	# of features	Only on Open Runs
Frequency of parts-of-speech	12	
In dictionary	11	
Sentence pattern	2	
Sites in search results	9	*
Terms in search results	8	*
Total	42	

feature value takes 1, otherwise takes 0.

- **Frequency of parts-of-speech**: This feature represents the frequency of parts-of-speech (POS) in a query, for instance, noun, verb, adverb, and etc. We performed a morphological analysis for a given query, and computed the frequency of each POS in the query. The feature value of each POS was normalized by the number of morphemes that appear in the query. We used MeCab [1] for Japanese morphological analysis.

- **Query unigram**: This feature indicates what characters are included in a query. We empirically selected 85 single characters that were expected to be included especially in the name of personal name and LOCATION queries. For example, when a query includes a character such as "*o* (男)" or "*ko* (子)," the query is likely to be a personal name.
- **Sentence pattern**: In this feature category, there are two binary features: one is a feature indicating whether a sentence ends with terms such as "*ka* (か)," "*ha* (は)," or "?," while another is a feature indicating if the query includes an interrogative such as "who (だれ)," "why (なぜ)," "when (いつ)," and "where (どこ)." These feature values are set to 1 if the query matches those patterns.
- **Number of documents containing expanded query**: This feature is included for distinguishing CELEBRITY and DEFINITION, and is approximated in practice by a hit count of Web search results obtained for an expanded query. We prepared 15 prefixes such as "*san* (さん)," "*shi* (氏)," and "*towa* (とは)." The classifier modified the query by using the prefixes such as "*query-san* (*query* さん)" and "*query-sama* (*query* 様)," and got the number of Web search results obtained by the Yahoo! Japan Web search API[2]. If the ratio of hit counts of the modified queries to that of the original query is high, the query might be the name of person or not. Some prefixes such as "*senshu* (選手)" and "*gi-in* (議員)" aim to discriminate ARTIST, ACTOR, POLITICIAN and ATHLETE.
- **Number of search results**: If the number of search results is more than 10, this value is 1; otherwise 0.
- **Terms in search results**: We heuristically selected 36 terms that may characteristically appear in search results in response to each types of query. Examples of such terms are "born at," "proper name" and "profile." If these terms appear in a Web page, the page may contain information about a person. Other examples are "access," "minute walk away" and "*chome* (丁目)." These terms may be helpful to classify the query into the FACILITY and GEO types or others. We used Yahoo! Japan Web search API and counted terms in the returned snippets. The value was normalized by the total number of search results.
- **Sites in search result**: It is a binary feature that indicates whether the top 50 search results of a query contain the pre-defined sites: travel sites restaurant guide ranking sites and so on.

Training data was created manually including sample queries distributed to participants. Some features cannot be used on mandatory run, because those feature values are calculated by using web search engine. The number of training queries was 400, in which each class contained 50 queries.

2.2 Weighting Function for IE Methods

After we extracted information pieces by using all the information extractors we developed, we then merge those information pieces prioritizing more relevant information with a result of the query classification. More specifically, given a set of pairs of an extracted information unit and its relevance score, our weighting function multiplies a certain constant, which is heuristically pre-defined for each query type and information extractor, to the relevance score.

Letting t be the query type predicted, e be an information extractor used for $u \in U$, and $\mathrm{Score}(u)$ be the relevance score of u, we update the relevance score of u according to the following equation:

$$\mathrm{Score}'(u) = c_{t,e}\mathrm{Score}(u)\,,$$

where $c_{t,e}$ is a constant predefined for a query type t and an information extractor e and reflects the importance of information pieces obtained by the information extractor e to the query type t. The constant $c_{t,e}$ was heuristically set as shown in Table 3.

2.3 Information Summarizer

Through information extraction, information units that describe a given query are obtained. In order to summarize the obtained units under the restriction on the number of output characters, we need to consider both relevance of each output unit and diversity of the whole output. That is, the output should not contain similar units

[1] MeCab, http://mecab.sourceforge.net/

[2] Yahoo! Japan Web search API, http://developer.yahoo.co.jp/

Table 3. Values of the constant *c*.

	ARTIST	ACTOR	POLITICIAN	ATHLETE	FACILITY	GEO	DEFINITION	QA
IE for Highly Structured Information	2	2	1	2	5	4	2	1
Important Sentence Extraction	1	1	1	2	1	1	3	4
IE based on Hierarchical Headings	4	3	2	1	1	1	2	4

even if they are relevant, since the output length is strictly limited. Taking these two aspects (relevance and diversity) into account, we adopt Maximal Marginal Relevance (MMR), which is a document summarization method proposed by Carbonell and Goldstein [2], for summarization.

The summarizer receives a set of pairs of an extracted information unit and its relevance score. Here, relevance scores are calculated by the weighting function described the above. An output information unit is decided by the following formula:

$$\mathrm{MMR} = \underset{u_i \in U \setminus U_{\mathrm{sel}}}{\arg\max} \left[\lambda \cdot \mathrm{Score}(u_i) - (1-\lambda) \cdot \max_{u_j \in U_{\mathrm{sel}}} \mathrm{Sim}(u_i, u_j) \right],$$

where U is a given unit collection, $\mathrm{Score}(u_i)$ is a given score of a unit u_i, U_{sel} is a subset of units in U that are selected as the output, $U \setminus U_{\mathrm{sel}}$ is a subset of units in U that have not yet selected as the output, and $\mathrm{Sim}(u_i, u_j)$ gives the similarity between units u_i and u_i. The parameter λ controls balance between relevance and diversity.

This formula iteratively selects an information unit for the output. The selected unit is added to U_{sel} for every iteration process, and the formula is calculated again to select the next unit to be included. The iteration process stops when U_{sel} has enough units for the output.

The similarity between two information units is defined by a cosine value of their feature vectors. A feature vector is generated for each unit. First, morphological analysis is performed to a unit. All nouns, verbs, and adjectives except stop words are then extracted from the unit. TF weighting or TF-IDF weighting is used to make a feature vector. When a unit does not include any noun, verb, or adjective, it is treated as a zero vector. In this case, we define the similarity between the unit and any other unit as zero.

As mentioned at the beginning of this section, each information extractors retrieve information units in the form of sentences or key-value pairs. We propose the following three methods for calculating the similarity between two units:

$$\mathrm{Sim}(u_i, u_j) = \begin{cases} \mathrm{Sim}_{\mathrm{sen}}(u_i, u_j) \\ \left(\mathrm{Sim}_{\mathrm{key}}(u_i, u_j) + \mathrm{Sim}_{\mathrm{val}}(u_i, u_j)\right)/2 \\ \sqrt{\mathrm{Sim}_{\mathrm{key}}(u_i, u_j) \cdot \mathrm{Sim}_{\mathrm{val}}(u_i, u_j)} \end{cases},$$

where $\mathrm{Sim}_{\mathrm{sen}}(\cdot)$, $\mathrm{Sim}_{\mathrm{key}}(\cdot)$ and $\mathrm{Sim}_{\mathrm{val}}(\cdot)$ functions calculate cosine similarity between sentence vectors, key vectors and value vectors for two given units respectively. A key-value unit is treated as a sentence in the first function. When calculating key- or value-similarity, a sentence unit is treated as a key-value pair where a key is an empty string and value is the sentence itself. If both two units are sentences, we calculate the similarity between them using the first function.

3. IE METHODS

We propose three types of IE methods, which are designed to extract information from the specific structure of Web pages.

3.1 IE for Highly Structured Information

In this method, we hypothesize that important information about the given query can be represented by a pair of an attribute and its value. For example, a sentence of "His birth date is April 15." can be represented as (birth date, April 15), and "Tennis is his hobby." as (hobby, tennis). To extract such attribute-value pairs from Web pages, we take three approaches: (1) table-based extraction, (2) Wikipedia Infobox based extraction, and (3) regular expression based extraction.

3.1.1 IE from Table

First method extracts attribute-value pairs from *tables* in Web pages. Structured information is likely to be described in a tabular form in the Web pages. In this method, given a Web page, we first find tables that contain table header tags (<th>) by analyzing their DOM structures. If a table has the table header tags in its DOM, we treat these headers as attributes and the remaining rows/columns in the table as their values.

3.1.2 IE from Wikipedia Infobox

The method described in Section 3.1.1 extracts information from tables in Web pages. However, tables in Web pages often contain irrelevant information to the query, thus we need to detect more reliable sources that include as little noise as possible. As such a source, we leverage *Infobox* in Wikipedia. In Infobox of the query, attributes and their values are written in a well-structured table with a HTML <table> tag and Wikipedia articles are often highly relevant to the query. We extract pairs of attributes and their values by means of regular expression from the Infobox table.

3.1.3 IE based on Regular Expression

Some attributes such as postal addresses, phone and fax numbers, email addresses tend to be represented as some typical forms. For example, phone numbers can be represented as "TEL: *ddd-dddd-dddd*" (*d* means a digit number.) Moreover, these attributes are always important independently of the queries. Thus, we manually prepared the regular expressions to extract those attributes and their values. We use this approach for postal addresses, email addresses, phone and fax numbers.

3.2 Important Sentence Extraction

3.2.1 IE based on TextRank

The basic assumption behind the first important sentence extraction method is that, if a sentence has many similar ones in the search results for a query, then the sentence is important for the query.

In this method, we first obtain the top 200 Web search results for each query through Yahoo! Japan Web search API. From each snippet in the search results, we extract. We denote $S_x = \{s_1, s_2, \cdots, s_n\}$ as a set of extracted sentences (n means the number of extracted sentences from the search results). To calculate the importance of each sentence, we apply the LexRank [3] algorithm to the set of sentences S_x. Letting $\mathbf{p} = (p_1, p_2, \cdots, p_n)^{\mathrm{T}}$ be a vector that represents the importance of each sentence in S_x, the LexRank algorithm detects the importance of each sentence by the following recursive calculation like the PageRank algorithm:

$$\mathbf{p} = [d\mathbf{U} + (1-d)\mathbf{B}]^{T}\mathbf{p} ,$$

where **U** is a $n \times n$ square matrix whose elements are $1/n$, **B** is an adjacency matrix of the cosine similarity between two sentences, and d is damping factor. Intuitively, the LexRank algorithm assigns a high score to a sentence that is similar to many other sentences. Therefore, sentences that are likely to be written in many Web pages obtain higher scores.

After applying the LexRank algorithm, we obtain a set of sentence-value pairs:

$$T_x=\{(s_1,p_1),(s_2,p_2),\ldots,(s_n,p_n)\} ,$$

where p_i is the i_{th} value of **p**. When we output some sentences, it is better to have high diversity in the output. In the sentence-value pairs T_x, sentences with high values are similar to each other, thus it is inappropriate to select sentences in order of the value. Finally, we passed the set T_x to the information summarizer and summarized the sentences by using the MMR algorithm.

3.2.2 IE based on Access Information

Information on how to access a facility is important for the FACILITY and GEO queries. Such access information usually contains some typical words such as station names, distances and times. To extract sentences that are related to access information, we first define an important word list for access information in advance. Then, we create the classifier that classifies whether a sentence is access information by using the support vector machine. We prepared the important words like "km", "minutes", "go straight", "taxi", "buss" and "JR" and some features such as the length of a sentence to classify each sentence in the Web pages.

3.3 IE based on Hierarchical Headings

Many Web pages have hierarchical heading structure. In our laboratory, some members have been studying on heading structure extraction and block level Web page retrieval. Because these techniques seem to be also useful to coherent information unit extraction, we applied our prototype method to the prepared document collection. In this subsection, we describe such a trial.

3.3.1 Heading based Web Page Segmentation

In our method, all Web pages are segmented into blocks with added indirect headings at first. In this context, a block means a coherent information unit with the corresponding heading. Blocks may contain other blocks, and the content of the upper block includes ones of whole the lower blocks. On the other hand, a heading means a highly summarized description of the corresponding block. Note that blocks and headings in our definitions have one-to-one correspondence as above. Especially, indirect headings are headings of a kind, not corresponding to the focusing block directly, but corresponding to the upper blocks of the focusing block. Humans tend to supply indirect heading words to the content of the focusing block regardless of distance between the indirect heading and the focusing block. Therefore, it is useful to add indirect heading words to all blocks.

Our segmentation method is to be announced before long.

3.3.2 Weighted Topic Words Extraction

By this IE method based on hierarchical headings, we are able to extract query relevant information in the form of key-value pairs. We extract keys earlier than values. In this IE method, keys are topic words of entities referred to in the targeting query.

To extract topic words, firstly we retrieve blocks referring to the entities for extracting description of the entities. For this purpose, we utilize normal keyword search with the well-known search engine, namely Apache Solr, and its scoring function is BM25. To retrieve blocks, we store all blocks extracted from the prepared document collection into Solr, then execute the targeting query.

To extract topic words from retrieved blocks, we utilize very simple lexical patterns like "*w* of *t*" and "*t*'s *w*". In these patterns, *t* is a query term and *w* is a topic word. In Japanese, they correspond to "*t* の *w*". In this Japanese pattern, *w* means a topic word represented by an array of only Chinese characters and Japanese katakana. We attempt all terms in the targeting query as *t* to extract topic words comprehensively.

To improve usability of extracted topic words, we score and rank topic words. The score is based on this formula,

$$\mathrm{topicScore}(w) = \sum_{d \in D} \frac{\mathrm{occur}(w, d)}{\log_2 \mathrm{bingRank}(d) + 1},$$

where d is a document ranked as bingRank(d) in the prepared document collection D. If w appears on the lexical patterns in d then occur(w, d) is 1 else 0.

In the end of this phase, we filter out our predefined stop-words, like *photo*, *video* and so on from extracted topic words.

3.3.3 Weighted Key-Value Pairs Extraction

Finally, we execute query composed by targeting query terms and the focusing topic word on our block level retrieval system to retrieve description on the focusing topic of the entity referred to in the targeting query.

We utilize only main content of a block as the value of the key-value pair. For main content extraction, we simply extract the longest string in a style from the block. In this context, style means the information of visual appearance, namely font-size, font-weight, font-color and so on.

To weight the extracted key-value pairs, we multiply three feature values of the focusing pair. The first is topicScore(w) of the key w of the focusing pair, the second is the normalized rank logRank(d) of the page d from which the value of the focusing pair was extracted. The third is the normalized character count logLength(v) of the value v of the focusing pair.

The second feature value logRank(d) is based on this formula,

$$\mathrm{logRank}(d) = \frac{1}{\log_2 \mathrm{bingRank}(d) + 1},$$

which is similar to the formula of topicScore(w).

The third feature value logLength is based on this formula,

$$\mathrm{logLength}(v) = \frac{1}{\log_2 \mathrm{length}(v) + 20},$$

where length(v) is character count of v and length(v) is 500 or less. Note that if length(v) > 500 then logLength(v) = 0 because the length of whole an output string is limited under 500 even in the desktop run.

After all, key-value pair weighting function of this IE method is

$$\mathrm{weight}(w, v) = \mathrm{topicScore}(w)\mathrm{logRank}(d)\mathrm{logLength}(v),$$

where d is a page from which v is extracted.

At last, the system based on this IE method outputs weighted key-value pairs, where their keys are the topic words and their values

Table 4. Confusion matrix of KUIDL-QC-1 for Japanese Query Classification subtask.

gold \ system	ARTIST	ACTOR	POLITICIAN	ATHLETE	FACILITY	GEO	DEFINITION	QA	Recall
ARTIST	**9**	0	0	1	0	0	0	0	0.90
ACTOR	0	**10**	0	0	0	0	0	0	1.00
POLITICIAN	0	0	**10**	0	0	0	0	0	1.00
ATHLETE	1	0	0	**9**	0	0	0	0	0.90
FACILITY	1	1	0	0	**12**	1	0	0	0.80
GEO	0	0	0	0	0	**15**	0	0	1.00
DEFINITION	0	0	0	0	4	1	**10**	0	0.67
QA	0	0	0	0	0	0	3	**12**	0.80
Precision	0.82	0.91	1.00	0.90	0.75	0.88	0.77	1.00	

are the main contents of the retrieved blocks. The weights are normalized linearly between 0 and 1 where the best pair gets 1.

4. EVALUATION RESULTS

In this section, we describe the setting for each run, and our evaluation results. Obtaining output text for each query, we just shortened the text so that the length of the output meets the requirement for each run. Note that we did not use any methods specific to types of runs.

Overall, we submitted the following two runs for Query Classification subtasks: 1) KUIDL-QC-1 (Japanese query classification using all features in Table 1), and 2) KUIDL-QC-2 (Japanese query classification using first five feature categories in Table 1). As for Main tasks, we submitted the following eight runs: 1) KUIDL-J-D-MAND-1 (Japanese DESKTOP Mandatory run), 2) KUIDL-J-D-OPEN-2 (Japanese DESKTOP Open run), 3) KUIDL-J-M-MAND-3 (Japanese MOBILE Mandatory run), 4) KUIDL-J-M-OPEN-4 (Japanese MOBILE Open run), 5) KUIDL-E-D-MAND-1 (English DESKTOP Mandatory run), 6) KUIDL-E-D-OPEN-2 (English DESKTOP Open run), 7) KUIDL-E-M-MAND-3 (English MOBILE Mandatory run), and 8) KUIDL-E-M-OPEN-4 (English MOBILE Open run).

4.1 Query Classification Subtask

We first introduce the accuracy of our query classifier described in Section 2.1. Classification accuracies of KUIDL-QC-1 and KUIDL-QC-2 are 0.87 and 0.66, respectively. The possible reason why the former run achieved higher accuracy is that the classifier in KUIDL-QC-1 has additional useful features (*e.g.*, terms in search results) compared to one in KUIDL-QC-2.

We show the confusion matrix for KUIDL-QC-1 in Table 4. Our classifier achieved quite high accuracy for all the query types on the whole. In particular, we can find that our classifier could classify ACTOR, POLITICIAN and ATHLETE queries with high precision values. This could be mainly because the features that expand an original query with prefixes such as "*senshu* (*選手*)" and "*gi-in* (*議員*)" could be useful for classifying person name queries into ARTIST, ACTOR, POLITICIAN categories.

On the other hand, the precision values for FACILITY and DEFINITION queries were comparatively low. The table shows that some QA queries are misclassified into DEFINITION category. Since QA queries tend to be long compared to other type queries, and tend to contain interrogative words, we prepared two feature categories 1) query length, and 2) sentence pattern for QA query classification. However, misclassified QA queries (*e.g.*, queries for 1C2-J-0090 and 1C2-J-0091) do not have such characteristics. As for DEFINITION type, we did not prepare features for the type in classification setup. This could be main reason for the low precision for DEFINITION queries.

Table 5. Overall S#-measure for each run for 1CLICK-2 JAPANESE task.

Run Name	union	intersection
KUIDL-J-D-MAND-1	0.165	0.099
KUIDL-J-D-OPEN-2	0.187	0.114
KUIDL-J-M-MAND-3	0.154	0.092
KUIDL-J-M-OPEN-4	**0.190**	**0.118**

Table 6. Overall S#-measure for each run for 1CLICK-2 ENGLISH task.

Run Name	S#-measure
KUIDL-E-D-MAND-5	0.058
KUIDL-E-D-OPEN-6	0.076
KUIDL-E-M-MAND-7	**0.157**
KUIDL-E-M-OPEN-8	0.149

4.2 Main Task

In this subsection we first report overall results of our formal runs. Table 5 and 6 show S#-measure [1] of overall results for each run in Japanese task and English task respectively. In Table 5, rows named *union* denote the S#-measure scores computed based on the union between two assessors' iUnit matches, while rows named *intersection* denote the scores computed based on the intersection between them. Among our formal runs, the OPEN runs achieved higher S#-measures compared with the MANDATORY runs.

Next, we present more detailed results of our formal runs. Table 7 and Table 8 show S#-measure of KUIDL-J-D-MAND-1 and KUIDL-E-D-MAND-5 for each query type, respectively. Figure 2 and 3 show the per-query S#-measures for KUIDL-J-D-MAND-1 and KUIDL-E-D-MAND-5, respectively. The rest of this subsection we discuss our results for each query type.

Table 7. S#-measure of KUIDL-J-D-MAND-1 for each query type.

	union	intersection
ARTIST	0.249	0.142
ACTOR	0.300	0.185
POLITICIAN	0.119	0.058
ATHLETE	0.315	0.245
FACILITY	0.233	0.138
GEO	0.039	0.011
DEFINITION	0.117	0.093
QA	0.106	0.062

Table 8. S#-measure of KUIDL-E-D-MAND-5 for each query type.

	S#-measure
ARTIST	0.033
ACTOR	0.055
POLITICIAN	0.036
ATHLETE	0.043
FACILITY	0.050
GEO	0.025
DEFINITION	0.088
QA	0.110

4.2.1 CELEBRITY Queries

For CELEBRITY (namely ARTIST, ACTOR, POLITICIAN and ATHELETE) queries, the greater part of our handholds like Wikipedia Infobox, TextRank and Topic words seems effective. Actually, our S#-score about Japanese CELEBRITY queries is higher than ones about other types of queries (see Table 7).

However, about English queries, our S#-score is not so high (see Table 8). Meaningless topic words are an explanation of this difference. It is mainly because of too simple English lexical patterns (see Section 3.3.2).

Especially about POLITICIAN queries, our S#-score for both languages is lower than ones about other CELEBRITY queries.

4.2.2 FACILITY Queries

For FACILITY queries, highly structured information extraction worked well. Particularly, addresses and telephone numbers were successfully extracted by regular expression (see Section 3.1.2) in the most of the cases.

4.2.3 GEO Queries

For GEO queries, our system could not output useful information efficiently. It is because our method did not count queries for an object set. About other query types, the system is expected to output a description of an object. Our method is designed for these types of queries. In contrast, about GEO queries, the system is expected to output clearly separated multiple descriptions for each facility. However, our method has no special step to separate descriptions for multiple facilities as Section 2.

For example, about the English GEO Query 1C2-E-0140 ("oregon beaches"), our system output into KUIDL-E-D-MAND-5 multiple telephone numbers and Email addresses without their belonging.

4.2.4 DEFINITION Queries

Definition was the most difficult query type to classify as shown in Table 4, so that our result is affected by accuracy of query classification.

TextRank was useful for this type of queries. Definition can be considered as an attribute. Hence, the system might be able to filter out key-value pairs by heuristics about key names. However, we did not utilize this feature.

4.2.5 QA Queries

QA was also one of the most difficult query types for our system, even though we could classify the QA type queries with high precision. In this work we did not use QA-specific features such as answers of community QA sites. This kind of feature may improve our methods.

5. CONCLUSION

In this paper, we proposed an information extraction based framework for the NTCIR-10 1CLICK-2 task. Our framework first classifies a given query into the predefined eight categories, then extracts pieces of relevant information by combining three different methods from various Web pages, and finally summarizes the obtained pieces into a short text.

6. ACKOWLEDGMENTS

This work was supported in part by KAKENHI (#24240013, #24680008).

7. REFERENCES

[1] M. P. Kato, M. Ekstrand-Abueg, V. Pavlu, T. Sakai, T. Yamamoto and M. Iwata. Overview of NTCIR-10 1CLICK-2 Task. *NTCIR-10 Proceedings*, 2012.

[2] J. Carbonell and J. Goldstein. The use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335-336, 1998.

[3] G. Erkan and R.D. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1), pages 457-479, 2004.

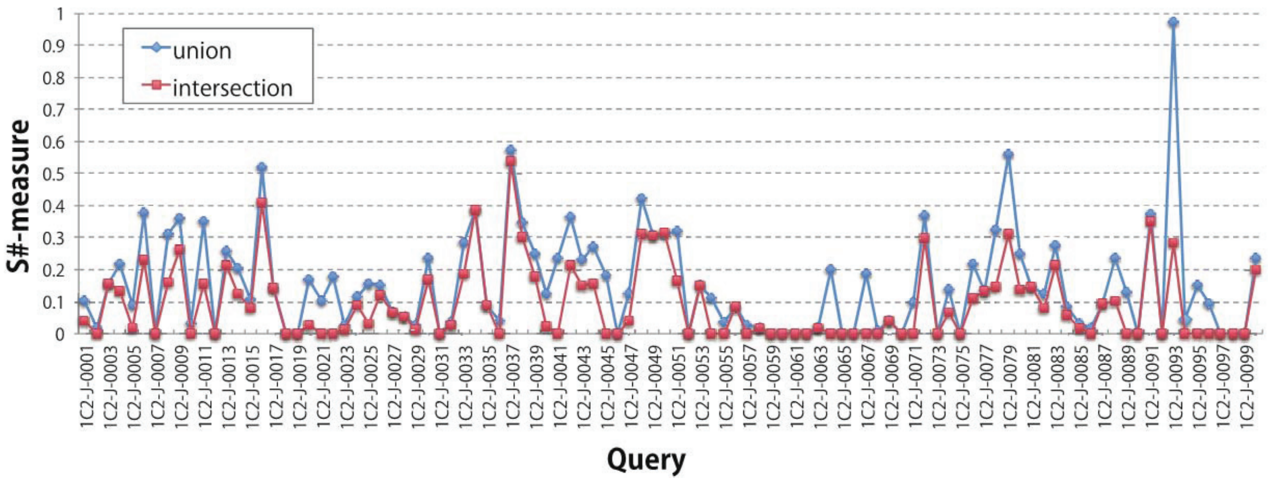

Figure 2. Per-query S#-measure of KUIDL-J-D-MAND-1.

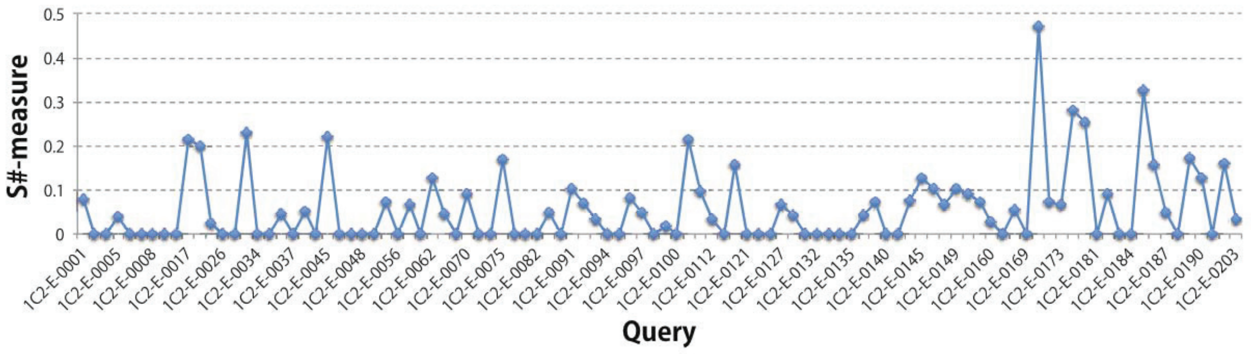

Figure 3. Per-query S#-measure of KUIDL-E-D-MAND-5.