Information Extraction based Approach for the NTCIR-9 1CLICK Task

Makoto P. Kato, Meng Zhao, Kosetsu Tsukuda, Yoshiyuki Shoji,
Takehiro Yamamoto, Hiroaki Ohshima, Katsumi Tanaka
Department of Social Informatics, Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto 606-8501, Japan
{kato, zhao, tsukuda, shoji, tyamamot, ohshima, tanaka}@dl.kuis.kyoto-u.ac.jp

ABSTRACT
We describe a framework incorporating several information extraction methods for the NTCIR-9 One Click Access Task. Our framework first classifies a given query into pre-defined query classes, then extracts information from several Web resources by using a method suitable for the query type, and finally aggregates pieces of information into a short text.

Keywords
Information extraction, diversification, query classification

Team Name: KUIDL
Language: Japanese
External Resources Used: Yahoo! Japan Web search, Yahoo! Chiebukuro, and Japanese Wikipedia

1. INTRODUCTION
Kyoto University, Department of Informatics, Digital Library laboratory (KUIDL) participated in the NTCIR-9 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 several Web resources by using a method suitable for the query type, 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*, and *information summarizer*. The implementation of a query classifier and information summarizer is then introduced right after the description of our framework, and that of information extractors is described in the following sections.

Our framework is depicted in Figure 1. A query classifier first classifies a given query into pre-defined four query types, i.e. CELEBRITY, LOCATION, DEFINITION and QA. The classified query is processed by an information extractor that is suitable for the query type. The information extractor retrieves pieces of information from an appropriate data resource, and computes 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) or MOBILE run (140 characters). Note that we used the same algorithms for generating our DESKTOP and MOBILE runs.

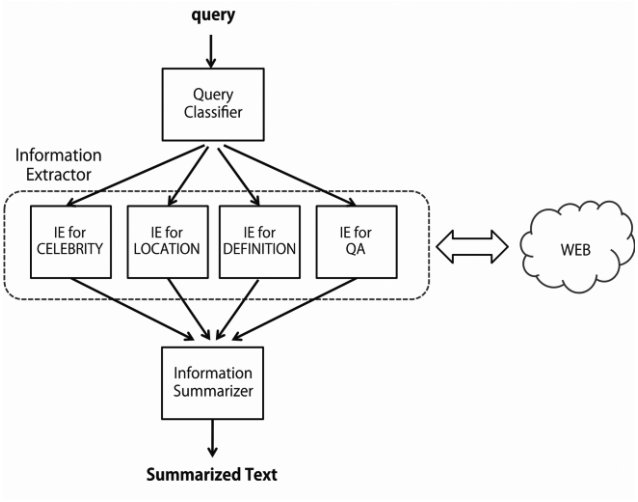

Figure 1. Our framework.

2.1 Query Classifier
First of all, we classified a query into four types by using a multi-class support vector machine, where we incorporated 8 types of features shown in Table 1.

- **Has Wikipedia article**: It is a binary feature that indicates whether an article about the query exists in Wikipedia[1]. This feature takes 0 or 1.
- **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 morphs that appear in the query. We used MeCab [2] 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 CEREBRITY 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.

[1] Wikipedia, http://ja.wikipedia.org/

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

Table 1. Features used in query classification.

Feature	# of features
Has Wikipedia article	1
Frequency of Parts-of-speech	44
Query unigram	85
Sentence pattern	2
Number of documents containing expanded query	15
Has Travel services	1
Number of search results	1
Terms in search results	39
Total	**185**

- **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)," "*query-sama* (*query* 様)," and "*query-senshu* (*query* 選手)" and got the number of Web search results obtained by the Yahoo! Japan Web search API[3]. 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 CELEBRITY.
- **Has travel service**: It is a binary feature that indicates whether the top 50 search results of a query contain the pre-defined travel sites.
- **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 39 terms that may characteristically appear in search results in response to CEREBRITY and LOCATION queries. 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 celebrity. Other examples are "access," "minute walk away" and "*chome* (丁目)." These terms may be helpful to classify the query into the LOCATION class or other classes. 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.

Training data was created manually including sample queries distributed to participants. The number of queries was 400, in which each class contained 100 queries.

2.2 Information Summarizer

Through information extraction, several sentences that describe a given query are obtained. The final output is made by summarizing the sentences. Out method is based on Maximal Marginal Relevance (MMR), which is a document summarization method proposed by Carbonell and Goldstein [2]. The summarizer needs to consider both relevance of each output sentence and diversity of the whole output. That is, the output should not contain similar sentences even if they are relevant, since the output length is strictly limited.

The relevance of a candidate sentence is computed differently according to the query type. Therefore, the input for the summarizer is a set of pairs of a sentence and its relevance score.

An output sentence is decided by the following formula:

$$\mathrm{MMR} = \underset{d_i \in D \setminus S}{\operatorname{argmax}} \left[\lambda(\mathrm{Score}(d_i) - (1-\lambda) \max_{d_j \in S} \mathrm{Sim}(d_i, d_j) \right].$$

Where D is a given sentence collection, $\mathrm{Score}(d_i)$ is a score of a sentence d_i, S is a subset of sentences in D that are selected as the output, $D \setminus S$ is a subset of sentences in D that have not yet selected as the output, and $\mathrm{Sim}(d_i, d_j)$ gives the similarity between sentence d_i and d_j. λ is a controlling parameter.

This formula selects a sentence for the output. The selected sentence is added to S, and the formula is calculated again to select the next sentence to be included. If S has enough sentences for the output, the method stops.

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

Some information extraction methods for CEREBRITY and LOCATION give sentences where each of the sentences consists of an attribute and its value. For example, a tuple ("Phone", "075-753-5385") is given as a sentence. In this case, words that appear in the attribute and ones in the value are treated differently. It means that even if the same word appears in both of them, they are treated as different dimensions in a generated feature vector.

3. IE METHODS

We propose four types of IE methods, which are designed so that required information for each query type can be extracted effectively.

3.1 IE for CELEBRITY Query

For the CELEBRITY type of queries, required information indicated in a document of NTCIR-9 1CLICK nugget creation policy[4] includes date/place of birth, real name, blood type, height, and etc. Those kinds of information can be represented as an attribute name and attribute value pair, though some of them are difficult to represent in such a form (e.g. personal history). Thus, we propose a method to extract pairs of attribute name and its value, and predict the importance of those pairs. On another front, it is difficult to precisely extract pairs of an attribute name and its value from unstructured text. For robustly extracting important pieces of information from the Web, we also introduce a method that extracts important sentences from the text by using a machine learning technique.

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

[4] http://research.microsoft.com/en-us/people/tesakai/1click.aspx

3.1.1 Attribute Name and Value Pair Extraction

In this method, we hypothesize that important information about the celebrity can be represented by a pair of an attribute and its value. For example, a sentence of "His birth date is April 15." is represented as (birth date, April 15), and "Tennis is his hobby." as (hobby, tennis). However, it is not so easy to correctly extract attributes and their values from documents, even from a sentence. Therefore, we need to detect sources that include as little noise as possible. As such a source, we leverage *Infobox* in Wikipedia. In Infobox of a celebrity, attributes and their values are written in a well-structured table with a HTML <table> tag. We extract pairs of attributes and their values by means of regular expression from the table. We then obtain a list of pairs of attributes and their values for a celebrity x as following:

$$L_x = [(a_1, v_1), (a_2, v_2),\ldots,(a_k, v_k)]$$

where a_i is an attribute, v_i is the value, and k is the number of pairs for the celebrity. We regard an attribute and its value written in upper rows in Infobox as more important information than those in lower rows. That is, the order of pairs in the list is that in Infobox.

We also use sentences in an article of a celebrity x in Wikipedia because the amount of information from Infobox is not enough for some of celebrities. However, it is difficult to extract attributes and values from sentences as discussed earlier. To avoid false detection, we use sentences as they are. First, we retrieve all the sentences from the article. Then we apply LexRank algorithm to the sentences. We explain the details of the LexRank algorithm in Section 3.3. The LexRank algorithm returns a set of sentence-importance pairs. Finally, we apply the MMR algorithm to the resultant set and obtain a set of sentence-ranking pairs.

$$R_x = \{(s_1,r_1),(s_2,r_2),\ldots,(s_n,r_n)\}$$

We join the sentences for the output of our system based on L_x and R_x. From L_x, we make an sentence of "a_1 is v_1, a_2 is v_2, …, and a_k is v_k". Then we append each sentence in R_x in order of r_i to the end of the sentence generated from L_x.

3.1.2 Important Sentence Extraction

In this method, we extract important sentences from Wikipedia articles as we believe those articles contain sufficient information that meet the need of a celebrity query. We extract twelve features from each sentence, and apply a machine learning method to predict the importance of those sentences. Before extracting each feature from sentences, we divide the whole article into several sections according to the list of contents.

- **RelativePosition**: The position of a sentence in a document implies the importance of the sentence in the document. For example, a sentence that appears at the very beginning of a document may be the most important sentence in the document. Thus, we utilize the position of a sentence in a section as a feature, which is defined as follows:

$$\mathrm{Score}_{\mathrm{Relative}(s_k)} = \frac{\sum_{i=1}^{k} \mathrm{Length}(s_i)}{\sum_{i=1}^{n} \mathrm{Length}(s_i)}$$

  where $\mathrm{Length}(s_i)$ is the length of a sentence s_i, n is the total number of sentences in each section.

- **ContainsQuery**: We include a feature indicating whether the current sentence contains the query or not.

$$\mathrm{Score}_{\mathrm{Query}(s_k)} = \begin{cases} 0, & \text{if the query word does not appear in this sentence} \\ 1, & \text{if the query word appears in this sentence} \end{cases}$$

- **ContainsInfobox**: As we described in 3.1.1, we have already extracted pairs of attributes and their values from Infobox. Here we use a binary feature that indicates whether a sentence contains any attribute or value word.
- **MaxCos**, **MinCos**, **MaxInnerProd**, **MinInnerProd**, **AvgCos**, **AvgInnerProd**: In these six methods, we first search by a given query through Yahoo! Japan Web search API to retrieve the top n (n = 20) search results. Then we calculate the cosine, and inner-product between each sentence from Wikipedia and each snippet from the search results. The maximum, minimum, and average of cosine and inner-product values are found for each sentence from Wikipedia. We apply normalization after calculating scores of all the sentences in a section.
- **ProperNounNum**: For each sentence in each section, we perform morphological analysis to count how many proper nouns are included, and finally apply normalization to get a score between 0 and 1.
- **NumberNum**: This feature indicates how many numbers are included in a sentence, as we think number information such as year information is important for some kinds of query.
- **ImportantWordNum**: We constructed an important word list in advance. Those words include "first", "hometown", "debut", "get a prize", "nickname", "like" and "hate". Then we count how many times those important words appear in each sentence and finally apply normalization to get a score between 0 and 1.

In addition, we extract link information, such as the URLs of each celebrity's twitter, blog, official personal page, and so on.

To predict the importance of each sentence, we built a support vector machine regressor. Training data were constructed by test queries that were distributed to participants before the formal run. We extracted sentences for the test queries and features of those sentences. Two assessors evaluated the importance of nuggets for each query at 3 scales, and simply used the sum of the two assessors' scores as the importance score of each nugget. We then estimated the importance of sentences based on the scored nuggets. To automatically estimate the importance score of sentences, we proposed a method to estimate the relevance of a sentence to a nugget. It is difficult to automatically estimate the relevance of text to a nugget, and the difficulty is caused by several factors. Nuggets are typically very short, can take different forms for the same meaning, and sometimes contain digits and symbols, e.g. phone number, date, and address. Thus, we estimated the nugget relevance as not binary value but probability of the relevance. Moreover, character matching was used instead of word matching to calculate the nugget relevance. We defined the relevance of a sentence s to a nugget n as follows:

$$\mathrm{Rel}(n,s) = \frac{|C_n \cap C_s|}{|C_n|}$$

where C_n and C_s is a set of characters contained in the nugget n and sentence s, respectively. The nugget relevance occasionally achieves score to some extent when a given sentence is long, even if the sentence is not relevant to a nugget. Therefore, we set a threshold to exclude low nugget relevance, which is usually noise. Based on the nugget relevance, we computed the importance of each sentence as follows:

$$\mathrm{Score}(s) = \sum_{n \in N} \mathrm{Score}(n)\mathrm{Rel}(n,s)$$

where Score(n) is the sum of the two assessors' evaluation scores, and N is a set of nuggets for a query.

Finally, we trained a support vector machine regressor with the training data to predict the importance score. We opted to use LIVSVM[5] with a default parameter setting for this regression. To avoid the redundancy of information, we applied the information summarizer to sentences with the importance score, and obtained ranked sentences. The output of a celebrity query was obtained by concatenating top-ranked sentences up to the limit length.

3.2 IE for LOCATION Query

For the LOCATION query, the system needs to return information that enables the user to physically visit or contact a facility. Thus, the system should return facts such as postal and email addresses, phone and fax numbers, opening hours, how to access the facility by train/bus/car, nearest stations, time required for the travel, whether the facility has a car park, its opening hours, and etc., according to the NTCIR-9 1CLICK nugget creation policy.

In this method, we first search for Web pages that may contain those kinds of required information. We assume that required information for the LOCATION query is described in the *official* Web pages of the facility. For example, when the query is "Kanazawa University", required information is usually described in the homepage of Kanazawa University. To find these official Web pages, we first use the Yahoo! Japan Web search API, and obtain the domain of the top search result in response to the given query. We then create new queries by combining the domain and query, as well as some pre-defined words such as "access", "address" and "inquiry." For example, when the query is "Kanazawa University", the domain of the top search result would be "www.kanazawa-u.ac.jp". Then, the method creates queries like "site:www.kanazawa-u.ac.jp AND Kanazawa University AND access", and obtains top ranked Web pages by issuing the query. In this way, we gather Web pages that are likely to contain nuggets that meet the need of a LOCATION query.

After obtaining the Web pages, we try to extract attribute name and attribute value pairs from these pages, and adopt the following three approaches:

- **Regular expression based extraction:** 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. We use this approach for postal addresses, email addresses, phone and fax numbers.
- **Sentence based extraction:** Information on how to access the facility is represented as a sentence. Those kinds of information contain some typical words such as station names, distances, times. We define an important word list for access information in advance. Therefore, we create the classifier that classifies whether a sentence is access information by using the support vector machine to extract access information. 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.
- **Table based extraction**: Some attributes such as an opening hour of library and check-in and check-out time of a hotel are query dependent, thus we cannot prepare regular expressions or important words for extracting those attributes in advance. On the other hand, those attributes are likely to be described in a tabular form in the Web pages. In this approach, we find tables in the Web pages by analyzing their DOM structures and extract attribute names and their values from the tables.

Finally, we aggregate the obtained attributes. To assign scores to the attributes, we heuristically set weights for each attribute. After assigning scores to the attributes, the method passed them to the information summarizer and obtains summarized X-strings from it.

3.3 IE for DEFINITION Query

There are various ways to detect the definition of a term: searching for dictionaries available on the Web, summarizing a Wikipedia article about the term. In this paper, we focus on the phrase of "*towa* (とは)" in Japanese. In Japanese, most sentences that have a form of "*x* とは …" contain the definition of term *x*.

In this method, we first receive a query x and search with a new query "x とは". We then obtain the top 200 Web search results for each query through Yahoo! Japan Web search API. From each Web page in the search results, we extract sentences that include a phrase of "x とは". We denote $S_x = \{s_1, s_2, \ldots, s_n\}$ as a set of extracted sentences (n means the number of extracted sentences from the all Web pages in the search results). To calculate the importance of each sentence, we apply the LexRank [3] algorithm to the set of sentences S_x . Letting $\mathbf{p} = (p_1, p_2, \ldots, p_n)^T$ 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:

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

where $\mathbf{U}$ is a $n \times n$ square matrix whose elements are $1/n$, $\mathbf{B}$ is a 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 $\mathbf{p}$. 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.

3.4 IE for QA Query

We opted to use Yahoo! Japan Chiebukuro[6] as a main data resource for the QA type of queries, as this Q&A service contains over 70 million questions asked in Japanese, and might be able to cover various kinds of questions. A basic idea is that important

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[6] http://chiebukuro.yahoo.co.jp/

information for a QA type query is a set of answers for the query, which were posted for questions most similar to the query. We used Yahoo! Japan Chiebukuro Web API to access the Q&A data, retrieved QA pairs relevant to a given query, and extracted answers as important information from those pairs. To avoid confusion, we shall refer to a given query as *original query*, and a query generated to retrieve Q&A data as *search query* in this subsection.

Our IE method for the QA query type first generates search queries from an original query for retrieving QA pairs. Our method is to extract nouns, adjectives and adverbs from an original query. The method is quite simple but it works well because the dictionary is powerful. Our dictionary contains all of the Japanese Wikipedia entries and nouns, adjectives and adverbs from IPAdic legacy dictionary[7]. The number of terms is more than 1.3 million. Term extraction is performed according to the longest-match principle. In addition to nouns, adjectives and adverbs that might be effective for QA pair retrieval, we also extracted words specific to the question, which include "what," "where," and "who" ("*nani* (なに)," "*doko* (どこ)," and "*dare* (だれ)" in Japanese). Having extracted terms for QA pair retrieval, we generated a set of search queries by taking power sets of those terms.

QA pairs were exhaustively retrieved by using search queries generated from the original query, and scored based on the similarity between their questions and the original query. As we wanted to take into account sentence relevance rather than bag-of-words relevance, we extracted unigrams as well as bigrams and trigrams from all the retrieved questions, and computed the question similarity between an original query q and question q_i as follows:

$$\mathrm{Sim}(q, q_i) = \sum_{k=1}^{N} w_k \mathrm{Sim}_k(q, q_i)$$

where N was set to 3, and w_k is the weight for k-gram similarity. Sim_k was defined as follows:

$$\mathrm{Sim}_k(q, q_i) = \frac{\left| W_{k,q} \cap W_{k,q_i} \right|}{\left| W_{k,q} \right|}$$

where $W_{k,q}$ and $W_{k,qi}$ are sets of k-grams that are included in the original query q and a question q_i, respectively. Thus, the question similarity indicates the overlap between two questions in terms of unigram, bigram, and trigram.

Yahoo! Japan Chiebukuro provides an opportunity that a question asker can select one of the posted answers as the best answer. We leveraged this feedback system to identify the most relevant answer to a question, and basically output the best answer for the most similar question to an original query. However, this method includes a risk that the most similar question to an original query is not always relevant to the query. Thus, we extracted best answers to the most similar n questions to reduce the risk, and concatenated the first m words of those answers to produce X-string. In case that there is no best answer provided, we computed the k-gram similarity between an original query and answers, and then selected the most similar answer to the query as a pseudo best answer.

[7] http://sourceforge.jp/projects/ipadic/

Table 2. Result of query classification.

	CELEBLITY	LOCATION	DEFINITION	QA
True	15	15	15	15
True positive	13	14	15	14
False positive	0	0	4	0

4. EVALUATION RESULT

We describe the setting for formal run, and our evaluation results in this section.

Four formal runs were submitted: two runs for DESKTOP run, and two runs for MOBILE run. We utilized two different methods for CELEBLITY queries, which are described in Section 3.1.1 and 3.1.2, respectively. For the other queries, run results were produced using exactly the same method across four runs. Obtaining output text for each query, we just shortened the text so that the length of the output meets the requirement for each run, i.e. 500 characters for DESKTOP run, and 140 characters for MOBILE run. (Note that we did not use any methods specific to types of runs.) Overall, we submitted the following four formal runs: 1) KUIDL-D-OPEN-1 (DESKTOP run using the method described in Section 3.1.1 for CELEBLITY queries), 2) KUIDL-D-OPEN-2 (DESKTOP run using the method in Section 3.1.2), 3) KUIDL-M-OPEN-1 (MOBILE run using the method in Section 3.1.1), and 4) KUIDL-M-OPEN-2 (MOBILE run using the method in Section 3.1.2).

We first introduce the accuracy of our query classifier described in Section 2.1. Sixty queries were distributed to participants for the formal run, and were manually labeled as CELEBLITY, LOCATION, DEFINITION, or QA for this evaluation. The result of query classification is shown in Table 2.

The accuracy is 0.93, which is defined as the total number of true positive queries divided by the total number of queries. The four false positive queries ("GACKT," "Shinsuke Shimada," "Kanazawa University" and "the three national duties in Japan") were incorrectly classified into the DEFINITION type. The query "GACKT" was misclassified probably because our training data contained typical Japanese person names for the CELEBLITY type, but do not include plenty of special names such as English names and stage names, which are sometimes used especially for celebrities. We manually classified the query "the three national duties in Japan" into the QA type, while this query can also be considered as a DEFINITION type query. Thus, this misclassification might not negatively affect the overall performance. On the whole, our query classifier achieved quite high accuracy for the four query types.

Table 3 shows an overall result of our formal runs, where parenthetical value indicates score computed based on the union between two assessors' nugget matches, while non-parenthetical value indicates score based on the intersection (cf. [1]). Results for four runs are sorted by their S-measure score. As the runs KUIDL-D-OPEN-1 and KUIDL-D-OPEN-2 are different only in terms of a method for CELEBLITY queries, this result suggests that a method based on extraction of attribute name and value pairs slightly outperformed one based on extraction of sentences in the DESKTOP run. On the other hand, in the MOBILE run, the method based on sentence extraction achieved as high weighted recall and S-measure as that based on attribute name and value pair extraction. The difference between the two types of runs may indicate that the first 140 characters achieved almost the same

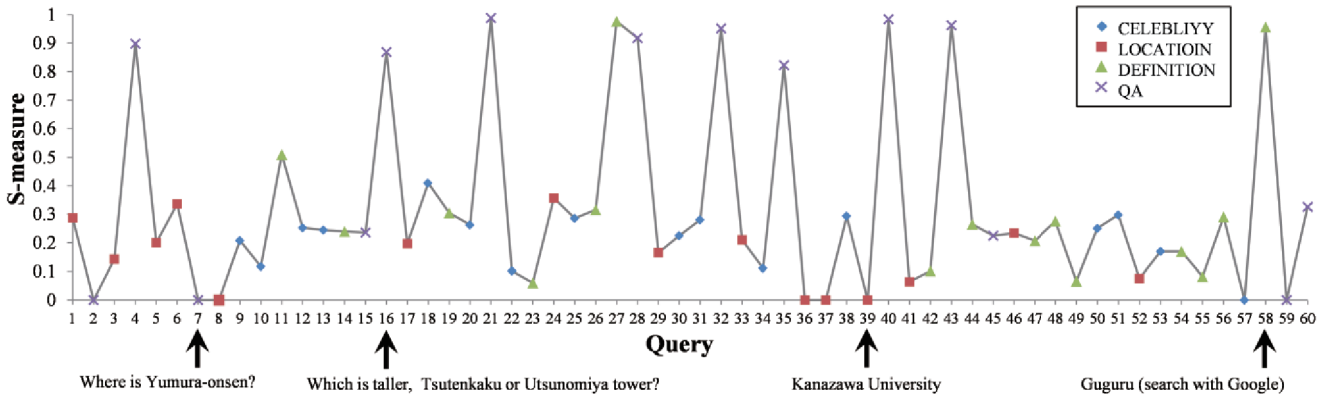

Figure 2. S-measure of a KUIDL-D-OPEN-1 result for each query.

Table 3. Overall result of our formal runs.

	Weighted recall	S-measure
KUIDL-D-OPEN-1	0.346 (0.423)	0.313 (0.381)
KUIDL-D-OPEN-2	0.341 (0.407)	0.290 (0.346)
KUIDL-M-OPEN-2	0.214 (0.262)	0.221 (0.273)
KUIDL-M-OPEN-1	0.204 (0.264)	0.219 (0.283)

Table 4. Evaluation results of KUIDL-D-OPEN-1 for each query type.

	Weighted recall	S-measure
CELEBLIYY	0.159 (0.191)	0.226 (0.273)
LOCATIOIN	0.151 (0.177)	0.261 (0.292)
DEFINITION	0.406 (0.460)	0.321 (0.354)
QA	0.644 (0.750)	0.552 (0.635)

quality in terms of weighted recall and S-measure, while the rest of output text by the method based on attribute name and value pair extraction was superior to that based on sentence extraction. (Recall that the first 140 characters of our DESKTOP and MOBILE runs contained exactly the same output text.)

Table 4 shows evaluation results of KUIDL-D-OPEN-1 for each query type. Overall, DEFINITION and QA types of queries obtained higher weighted recall and S-measure than CELEBLITY and LOCATION types. In general, DEFINITION and QA types of queries had much fewer nuggets than CELEBLITY and LOCATION types, and this probably made DEFINITION and QA queries easy to achieve high performance. The QA type of queries obtained the best result among the four types. Using a domain-specific corpus (i.e. Yahoo! Japan Chiebukuro) might lead to the reasonable performance.

To further investigate our evaluation result, we show S-measure of the KUIDL-D-OPEN-1 result for each query in Figure 2. Overall, our method achieved quite high S-measure for some DEFINITION and QA queries (e.g. queries "Guguru," and "Which is taller, Tsutenkaku or Utsunomiya Tower?" indicated in Figure 2), while it failed to output any relevant information for some LOCATION and QA queries (e.g. queries "Kanazawa University," and "Where is Yumura-onsen" indicated in Figure 2). A possible explanation for high S-measure for some QA queries is that we utilized a domain-specific site (i.e. a community QA site) to extract information, where in fact some queries were asked in the site, and were answered with most of the nuggets for those queries. On the other hand, methods for LOCATION and QA queries might sometimes fail to obtain any relevant information probably because a site used as a data source did not contain any required nuggets. (Recall that we focused mainly on official Web pages to extract information for LOCATION queries.) In our failure analysis on LOCATION queries, we found that some queries referred to multiple locations, and our method failed to identify one intended to be retrieved. Such an object identification problem was not observed for most of the CELEBLITY queries, and our method stably output relevant information to some extent. Our method for DEFINITION queries also achieved stable results for most of the queries, and quite high S-measure for some queries.

5. CONCLUSION

In this paper, we proposed a general framework for the 1CLICK task, which first classifies a given query into pre-defined query classes, then extracts information from several Web resources by using a method suitable for the query type, and finally aggregates pieces of information into a short text. We utilized difference types of IE methods for CELEBLITY, LOCATION, DEFINITION, and QA queries within our framework.

6. REFERENCES

[1] T. Sakai, M. P. Kato and Y.-I. Song. Overview of NTCIR-9 1CLICK. *NTCIR-9 Proceedings*, 2011.

[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.