Microsoft Research Asia at the NTCIR-9 1CLICK Task

Naoki Orii*
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213 USA
norii@cs.cmu.edu

Young-In Song
Microsoft Research Asia,
Beijing, P.R.C
yosong@microsoft.com

Tetsuya Sakai
Microsoft Research Asia,
Beijing, P.R.C
tetsuyasakai@acm.org

ABSTRACT

Microsoft Research Asia participated in the 1CLICK task at NTCIR-9 using two different techniques: a statistical ranking approach and the utilization of semi-structured web knowledge sources. The evaluation results show the effectiveness of our approach: we found a 50% increase in S-measure relative to the baseline. We present a module-by-module error analysis, showing directions for future work.

Keywords

information access, nuggets, knowledge annotation

Team Name: MSRA
Language: Japanese
External Resources Used: Bing, Yahoo! Chiebukuro, and Japanese Wikipedia

1. INTRODUCTION

Web search engines have long responded to users' information needs by simply returning a flat, ranked list of documents. While document retrieval has become the de facto standard in many search engines, several insufficiencies exist from the users' standpoint. One of the problems with current document retrieval is that users are not provided with direct answers to their information needs, but are provided with surrogate documents that have the possibility of contaiining answers to their needs. Thus, users are at times forced into a lengthy process in which they must first scan the ranked list of documents, then click on one or more pages that might answer his/her information needs, and finally locate text segments that actually answer these needs. This problem is especially prominent in the mobile environment, where the display size is limited and communication speed is yet slow. When the user intent is clear, in some cases it would be ideal if search systems could provide direct answers to his or her information need.

Recently, many search engines have started to display relevant content directly on the search result page, which Chilton and Teevan refer to as *Answers* [2]. For example, if a user searches for the query 'beijing weather', most search engines now show weather forecast information on the top of the result page.

In this paper, we describe Microsoft Research Asia's attempt to build a search system that directly satisfies the user's information need with a single textual output. We use a combination of statistical methods and utilization of semi-structured knowledge sources on the web. We present the effectiveness of our approach, as seen in the NTCIR-9 1CLICK task [13].

The remainder of this paper is organized as follows. Section 2 briefly outlines the NTCIR-9 1CLICK task, and Section 3 discusses the architecture of our system. In Section 4, we describe the evaluation results and present a module-by-module error analysis. We conclude our work in Section 5.

2. NTCIR-9 1CLICK TASK

The NTCIR-9 1CLICK task attemps to promote research in creating and evaluating systems that directly answers users' information needs. Formally, given a query, a system must return a single textual output of character length X or less, referred to as the X-string, that directly satisfies the user's information need. The query types are closed domain, with four types: celebrity (CE), local (LO), definition (DE), and question answering (QA). The evaluation is based on *nuggets*, where a nugget is defined as 'a short factual statement such that an assessor can judge whether a given text shows or clearly implies that statement to be true.'[1] Evaluation is done by manually matching parts of the X-string with gold-standard nuggets where they are semantically equivalent. Systems are awarded higher scores for presenting important pieces of information first and minimizing the amount of text the user has to read. For more detail on the NTCIR-9 1CLICK task, we refer the reader to [8] and [13].

3. SYSTEM ARCHITECTURE

Our system takes a pipeline approach to the task, consisting of 5 modules: Query Type Classifier, Document Retriever, Candidate Generator, Candidate Ranker, and Summarizer. The input to the system is processed by the modules in the order listed above.

The user's input query is first sent to the Query Type Classifier, which labels the query to one of the four given categories. The Document Retriever then finds relevant documents given the type-labeled query. Next, the Candidate Generator takes the retrieved documents and extracts candidate answers. The Candidate Ranker then ranks the candidates according to estimated relevance. Finally, the Summarizer combines the ranked candidates to generate the

*This work was done when the first author was an intern at Microsoft Research Asia.

[1] `http://research.microsoft.com/en-us/people/tesakai/nuggetpolicy.pdf`

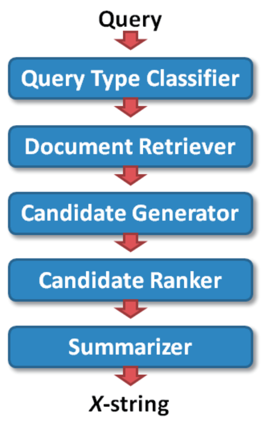

Figure 1: System overview

X-string that is presented to the user. We represent an overview of our system in Figure 1.

In the following subsections, we describe in detail each module of the system.

3.1 Query Type Classifier

The Query Type Classifier categorizes the input query into one of the four given types (CE, LO, DE, and QA). We took a statistical approach to this problem, using a multi-class SVM classifier. Next, we outline the features employed by the classifier.

3.1.1 Features from Query String

1. Query length: Different types of queries show different distributions for their length. For example, while a CE or LO query is generally short, consisting of two or three morphemes, a QA query is relatively longer than others. Based on this, we define a query length feature as:

   ```
   **if** $|q| \leq 5$ **then return** $|q|$
   **else return** 5
   ```

   where $|q|$ indicates the number of morphemes in a query.

2. Appearance of Particles: The appearance of particles in a query can also be an indicator for its query type. For example, QA queries usually contain rich amount of particles, such as "は" or "が", but CE queries rarely have even a single particle. While LO or DE queries can also contain particles, those which appear with such queries are of limited type. For instance, of the LO or DE queries with particle(s) in the training set, most of them have only "の" (of). Based on this observation, we define a binary feature with an output of 1 if a query has at least one particle which is not "の", and 0 otherwise.

3. Appearance of clue words: One of our observations is that there are some morphemes or characters that strongly indicate the specific type of a given query. For instance, a query that ends with the kanji character "園" (park) or the morpheme "学校" (school) is likely to be of type LO, and a query containing the morpheme "か" is highly likely to be of type QA. Also, the kanji character "子" appears more frequently with CE queries than of other types.To model this, we manually defined clue word lists for CE, LO, and QA and formulated binary valued features which indicate whether a query contains a clue word for a given type.

4. Character type combinations: The Japanese language has three different types of characters, hiragana, katakana, and kanji, and each type of queries has a unique characteristic of the character combinations. For example, CE queries tend to be all kanji, while LO tends to have a mix of katakana and kanji. QA is likely to be a mix of hiragana, katakana, and kanji. We utilize this observation by assigning a unique id to each possible character combination and defining a feature function returning an id corresponding to a given query.

3.1.2 Features from Web Search Results

Because a query is usually short, a query string often does not provide sufficient evidences to classify its query type. However, if we can collect its relevant web pages reasonably well, additional evidences could be collected from those web pages, enabling a better classification judgment. Based on this intuition, we derive several features by regarding the top 10 documents from a commercial search engine as relevant web pages for a given query[2].

1. Content words in the snippets: the intuition here is that specific words in snippets can hint at the type of a query. For example, the morpheme "生年月日" (birthdate) or "趣味" (hobby) strongly implies that a query is of CE type. Based on this intuition, we count the number of times a morpheme appears in the top 10 snippets for the training queries, and those morphemes with frequency greater than N for a query of a given type are used as binary features. In the experiments, we empirically set N to 3.

2. URL hosts: similar to above, URL hosts of retrieved documents also suggest the type of the given query. Observing a document of host `talent.yahoo.co.jp`[3] in the ranked list suggests that the query is of type CE, and a document of host `chiebukuro.yahoo.co.jp`[4] suggests that the query is of type QA. We model this observation by counting the number of times a URL host name appears in the top 10 retrieved documents for the training queries, and those with frequency greater than N for a specific query type are used as features. Again, N was empirically chosen to be 3.

3.2 Document Retriever

The Document Retriever is responsible for collecting relevant text from the web in response to a type-labeled query. We simply use the Bing API to retrieve the top 10 web documents when given a query. As an option, the Document

[2]We utilized the Bing API (`http://msdn.microsoft.com/en-us/library/dd251056.aspx`) to retrieve the top 10 pages in the experiments.

[3]A celebrity dictionary service ran by Yahoo! Japan

[4]The Japanese version of Yahoo! Answers, a community Q&A site

Retriever can also perform query expansion. The query expansion terms are manually constructed, and are specific for each query type. For example, if the query type is CE, we use query expansion terms such as “birth date” and “hometown.” If the query type is LO, we use terms such as “address” and “telephone.” When query expansion is enabled, we retrieve the top 1 document for each expansion term, for a total of $10 + N$ documents (where N is the number of query expansion terms for the type of the given query).

3.3 Candidate Generator

This module is responsible for generating candidate text fragments from the documents collected by the Document Retriever. The documents are first segmented by NLP-Win [10, 14]. Next, the module takes a sliding window n-gram approach and lists all possible n-grams (in all of our runs, n was empirically chosen to be 4). N-grams with sentence boundary markers within the intermediate elements are discarded.

During the above process, we record the absolute position of each n-gram. By “absolute”, we mean that an n-gram’s position is taken with respect to the total set of documents. For example, if we suppose that the last n-gram for the first document is at position 100, the first n-gram for the second document will be at position 101. The position of each n-gram will play a role when we need to combine and summarize the n-grams in section 3.5.

3.4 Candidate Ranker

This module collects the n-grams from the Candidate Generator and ranks them according to estimated relevance. We take a machine learning approach, specifically SVM Rank [4], which has shown its effectiveness in many literature.

As with any supervised learning method, one needs to first train a model, specifically, in this case, a ranking model. In 1CLICK, we are provided with gold-standard nuggets for 44 queries. On first thought, one might try to use these gold-standard nuggets directly as training data for constructing the ranking model. However, with supervised learning, the training data and test data should have the same “format”: using coherent, manually-crafted nuggets as training data and using (at times) incoherent, automatically-generated n-gram candidates as test data will most probably not rank the candidates successfully during test phase. In addition, there exists a practical problem where all of the gold-standard nuggets are “relevant” for a given query. Thus, if we were to use these gold-standard nuggets as training data, we would only have positive training data and have no negative training data at all. [5]

During testing phase, we are given candidate n-grams as input and would like to rank them in order of relevance. Thus, our training data should consist of pairs of an n-gram and its (estimated) relevance. We create this *pseudo* training data by a method similar to that used to automate document-summarization evaluations [3]: first, for each of the 44 queries, we perform document retrieval and candidate generation. Next, we automatically match each n-gram candidate against the list of gold-standard nuggets. N-grams that match a gold-standard nugget will be deemed relevant (positive) and n-grams without a successful match will be deemed non-relevant (negative). These pairs of n-grams and corresponding relevance information will comprise the training set for our ranking model. We note that distinct ranking models are constructed for each of the four query types.

The criteria for a match in the above process is as follows:

```
if a candidate n-gram contains a morpheme sequence of any relevant vital nugget for the given query, the candidate will be regarded as relevant
else if the candidate n-gram contains all of morphemes in any relevant nugget, it will be also regarded as relevant
else the candidate n-gram will be regarded as non-relevant
```

We next explain features used by SVM Rank.

3.4.1 URL-based Weighting Score

The URL of a given web document can hint at the quality of the information contained within the document for a given query type. For example, documents from `talent.yahoo.co.jp` can be more likely to be relevant to a CE type query than documents from other sites. Also, a part of a URL string can also hint at the content of page itself. For example, the token ‘access’ is often used by organizations to specify that a web page contains geographical information such as address, map, transportation, or direction: `http://www.osakatoin.ed.jp/access/index.html` lists the map and direction information of Osaka-Toin Middle School. Based on this idea, we define a URL-based weighting score and use it as one of features for ranking candidate n-grams as follows:

Given query q and candidate n-gram n that is extracted from a document with a URL consisting of m URL tokens $t_1 \cdots t_m$, the URL-based weighting score of n is defined as:

$$S_{URL}(n) = \max(w(t_1), \cdots, w(t_m)) \qquad (1)$$

In the equation, $w(t)$ is a weighting function for a URL token, which is an averaged likelihood of a candidate with t as its URL token to be relevant for the type T of query q:

$$w(t) = \frac{1}{C(q'; f_T(q') = T)} \sum_{n \in N_t} P(R|n \in N_t, q') \qquad (2)$$

where q' is a query in the labeled training data, N_t indicates a set of candidate n-grams containing URL token t, $C()$ is a count function, R is a random variable indicating relevance, and f_T is a function that returns the query type given a query.

3.4.2 Positive and Negative Words

There are certain key phrases that hint at the relevance of a given n-gram. For example, for a CE query, we can expect that an n-gram that contains the words “birthday” or “birth town” is probably relevant, while n-grams that contain the words “log in” or “terms of service” is probably not relevant. We name the former as positive words and the latter as negative words. Similar to the above URL-based weighting score, we calculate the strength of positive and negative words by using an average likelihood of a morpheme m to be relevant (or irrelevant) for a specific query type,

$$w_{pos}(m) = \frac{1}{C(q'; f_T(q') = T)} \sum_{n \in N_t} P(R|n \in N_m, q') \qquad (3)$$

[5]Note that the word “training” used here refers to the training of the candidate ranking model. We need to keep this distinct from the word “training” used for the 44 training queries. To avoid confusion, unless otherwise specified, the word “training” used in this subsection refers to the former.

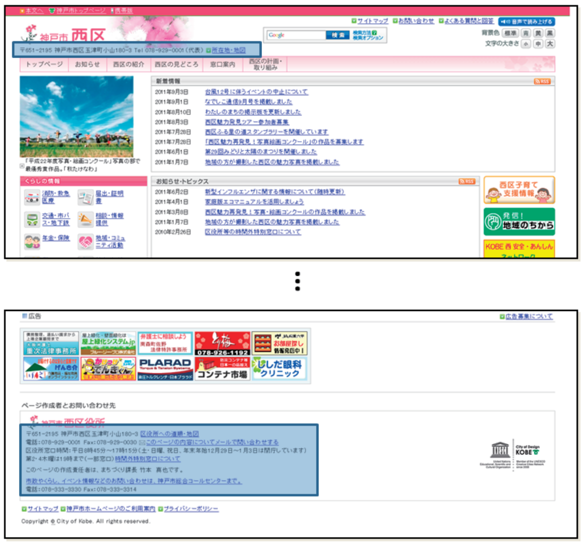

Figure 2: Examples of text chunks (shaded areas) on the official homepage of Kobe City West Municipal Office

$$w_{neg}(m) = \frac{1}{C(q'; f_T(q') = T)} \sum_{n \in N_t} P(\overline{R}|n \in N_m, q') \quad (4)$$

We then take the weight of the strongest positive or negative word and use them as two real-valued features.

3.4.3 Content similarity

The web is plagued with countless spam pages about "enlargement" and "free investment", and non-content phrases such as "copyright" and "privacy policy." In order to prevent n-grams that contain spam and/or non-content phrases from being ranked high on the list, we look at the content similarity of a given n-gram and the retrieved documents. More specifically, we treat the retrieved 10 documents (or 10+N documents in the case with query expansion enabled) as one large single document and calculate the cosine similarity between this document and a particular n-gram. Assuming that the Document Retriever did a good job in collecting non-spam pages, spam or non-content n-grams should have a fairly low similarity with respect to the rest of the documents.

3.5 Summarizer

This module merges the ranked n-grams given from the Candidate Ranker and generates the final textual system output that is presented to the user. The following heuristic is encoded during the merging process: we observed that what we deem as 1CLICK's answer often appeared in web documents in "chunks", with each chunk containing multiple nuggets that match with gold-standard nuggets. These chunks are often discrete, with a page usually containing from one up to four or five chunks. The merging algorithm exploits this observation and tries to reconstruct chunks from the ranked n-grams. An example of a web page with its corresponding chunks is shown in Figure 2. The psuedocode for this process is shown in Algorithm 1. Chunks are represented as a set of sets of n-grams.

As previously mentioned, in 1CLICK, systems are awarded for minimizing the amount of text the user has to read. However, it is highly probable for the n-grams given from the Candidate Ranker to contain duplicate information. Thus, in line 8, we check to see if an n-gram is similar to any of the selected text chunks and remove redundant n-grams. In addition, we prevent a text chunk from becoming too long by checking its length in line 6. In all of our runs, we empirically set the cut-off threshold L to 100. A given n-gram n is adjacent to a chunk c if the n is adjacent to any of the n-grams contained within c. The similarity measure in line 8 is a combination of cosine similarity and word overlap. We concatenate the list of chunks C to produce the system's final output.

Algorithm 1 Merge ranked n-grams into list of chunks

```
 1: Input: ranked n-grams $N = \{n_1, ..., n_{|N|}\}$
 2: Output: list of chunks $C$
 3: $C \leftarrow \{\{n_1\}\}$
 4: $N \leftarrow N \setminus n_1$
 5: for $n \in N$ do
 6:   if $n$ is adjacent to chunk $c \in C$ and $length(c) < L$ then
 7:     merge $n$ into $c$
 8:   else if $n$ is not similar with $c \in C$ then
 9:     create new chunk $c_{new} = \{n\}$
10:     $C \leftarrow C \cup c_{new}$
11:   end if
12: end for
13: return $C$
```

3.6 Knowledge Annotation of Wikipedia and Yahoo! Chiebukuro

In actuality, the method described in the previous subsections applies only to CE and LO query types in our system. For the DE and QA queries, we decided to utilize semi-structured knowledge sources on the web, a technique which Lin and Katz refer to as *knowledge annotation* [6], in order to enhance our system performance. The basis here is that user queries obey Zipf's law, where a small fraction of frequently asked question account for a significant proportion of all questions. These frequently asked questions could be easily answered by leveraging Web knowledge sources. Specifically, we used Wikipedia[6] and Yahoo! Chiebukuro[7] in this task.

We indexed the Japanese version of Wikipedia and Yahoo! Chiebukuro using Lucene.[8] For QA queries, when given a query we find the most similar question from Yahoo! Chiebukuro and return a concatenation of its answers. For DE queries, we do the same as above and in addition, find the most relevant Wikipedia article and return its abstract. The retrieval is done using Lucene's default similarity function. If no relevant Chiebukuro questions or Wikipedia articles are found, the system defaults to the statistical method described in the previous subsections.

4. EXPERIMENTS

In this section, we report on our results in NTCIR 1CLICK. We submitted two Desktop open runs, one run with query expansion and another without.

[6]http://ja.wikipedia.org/
[7]http://chiebukuro.yahoo.co.jp/
[8]http://lucene.apache.org/

4.1 Metrics for Evaluation

In NTCIR-9 1CLICK, systems are evaluated by manually matching the X-string with gold standard nuggets. During the matching process, assessors record the positional information for each successful match. The S-measure and the weighted nugget recall are the two evaluation metrics used for this task. The weighted nugget recall is the ratio of the sum of the weights of nuggets with successful match against the sum of the weights of all gold-standard nuggets. The S-measure [9] is an extension of the weighted nugget recall, in which the positions of the matches are also taken into consideration. As previously mentioned, systems are awarded higher scores for presenting important pieces of information first.

Each *X*-string is independently assessed by two assessors. Accordingly, two-versions of S-measure and weighted recall are used for evaluation:

- **I** computed based on the intersection between the two sets of nugget matches.
- **U** computed based on the union of the nugget matches.

Unless otherwise specified, we shall only consider **U** scores in the following subsections. For a disussion on inter-assessor disagreement, we refer the reader to Section 5.4 of [8].

4.2 Results

Table 1 shows the mean S-measure and weighted nugget recall values for our two runs and the manual and automatic runs. MSRA1click-1 is our run with query expansion, and MSRA1click-2 is our run without query expansion. The Manual run is an ideal run that is manually created using gold-standard nuggets, and acts as a practical upperbound. The Automatic run is automatically created by concatenating snippets retrieved by the Bing API, and can be considered as the baseline for this task. For more information on the manual and the automatic runs, we refer the reader to [9].

Overall, our run shows a general improvement in both S-measure and weighted recall compared to the baseline. On average, the S-measure and weighted recall improve from 0.203 to 0.327-0.329 and from 0.222 to 0.336-0.339, respectively, which is roughly a 50% increase in scores[9]. Scores drastically improve for QA queries, with S-measure and weighted recall improving from 0.182 to 0.540-0.575 and from 0.218 to 0.559-0.596, respectively. However, for the DE queries, the S-measure and weighted recall drops from 0.338 to 0.367-0.371 and from 0.423 to 0.409-0.413.

Contrary to our prior expectations, query expansion did not significantly improve the scores. The S-measure and nugget recall for MSRA1click-1 (with query expansion) are 0.329 and 0.336, respectively, while scores for MSRA1click-2 (without query expansion) are 0.327 and 0.339. This difference is not significant according to the Wilcoxon signed-ranked test with $p = 0.05$. If we consider only CE and LO queries, where effects of query expansion are expected to be most salient [10], we find that query expansion actually hurts the scores, with S-measure dropping from 0.189 to 0.167 for CE queries and from 0.209 to 0.205 for LO queries. We examine this phenomena by looking at the average number of nugget matches per query, which is shown in Table 2. Since the number of possible candidates monotonically increases due to additional documents retrieved by the expanded queries, if we assume that the candidate ranker and summarizer works ideally, it can be expected that query expansion should increase the number of nugget matches. However, as Table 2 shows, for CE queries the average number of nugget matches per query decreases from 7.867 to 7.067 and for LO queries from 4.133 to 3.800. This implies that non-relevant candidates added by query expansion are being ranked higher than pre-existing relevant candidates, indicating weaknesses in the candidate ranker and summarizer modules.

[9]The baseline, or the Automatic run, was evaluated by a single assessor, and can be considered as of type **U**.

[10]Most DE and QA queries are handled by the knowledge annotation method, not by the statistical method, so query expansion is expected to have no effect.

Table 2: Average number of nugget matches per query

	MSRA1click-1	MSRA1click-2
CE	7.067	7.867
LO	3.800	4.133
DE	3.200	2.877
QA	2.000	1.800

The per-query run performances for MSRA1click-2, the Manual run, and the Automatic run is shown in Figue 3. The general trend of MSRA1click-2 and Automatic runs resemble that of each other. We observe that average scores and variance are low for CE and LO queries, while average scores and variance are high for DE and QA queries. This suggests that the statistical ranking approach employed for the CE and LO queries are robust but shallow, while the knowledge annotation method employed for the DE and QA queries are effective but brittle and tend to founder depending on the given query.

4.3 Module-by-module Error Analysis

We represent a module-by-module error analysis in this subsection. Ideally, we would manually create gold-standard data (perfect input) for each module in order to evaluate the modules independently. However, as it is diffiult to define gold-standard input and gold-standard output for the candidate generator and candidate ranker, we approximate by using estimated weighted nugget recall for the candidate generator and resort to a qualitative evaluation for the candidate ranker. The data below all pertain to those without query expansion.

4.3.1 Query Type Classifier Performance

The confusion matrix for the query classification step is shown in Table 3. The classifier performed well, correctly classifying all of the CE and LO queries and all but one of the QA queries. The classifier had some difficulty with labeling DE queries. We show the list of misclassifications in Table 4. We can see that one of the misclassifications with the DE queries was Query 0051 "カラシニコフ" (Kalashnikov), which can be argued as of type CE (which our classifier predicts), considering that Kalashnikov is a gun inventor's name. Overall, our classifier showed 91.7% accuracy.

4.3.2 Document Retriever Performance

The estimated weighted nugget recall for the document retriever is shown in the leftmost column of Table 5. This estimation is done in a similar fashion to the way we matched

Table 1: Run mean performance. S-measure (in bold) / weighted recall (in italics)

	MSRA1click-1 (U)	MSRA1click-2(U)	Manual	Automatic
CELEBRITY	**.167**/*.122*	**.189**/*.136*	**.601**/*.530*	**.113**/*.099*
LOCAL	**.205**/*.214*	**.208**/*.254*	**.859**/*.861*	**.130**/*.150*
DEFINITION	**.367**/*.413*	**.371**/*.409*	**.865**/*1.000*	**.388**/*.423*
QA	**.575**/*.596*	**.540**/*.559*	**.985**/*1.000*	**.182**/*.218*
all	**.329**/*.336*	**.327**/*.339*	**.828**/*.824*	**.203**/*.222*

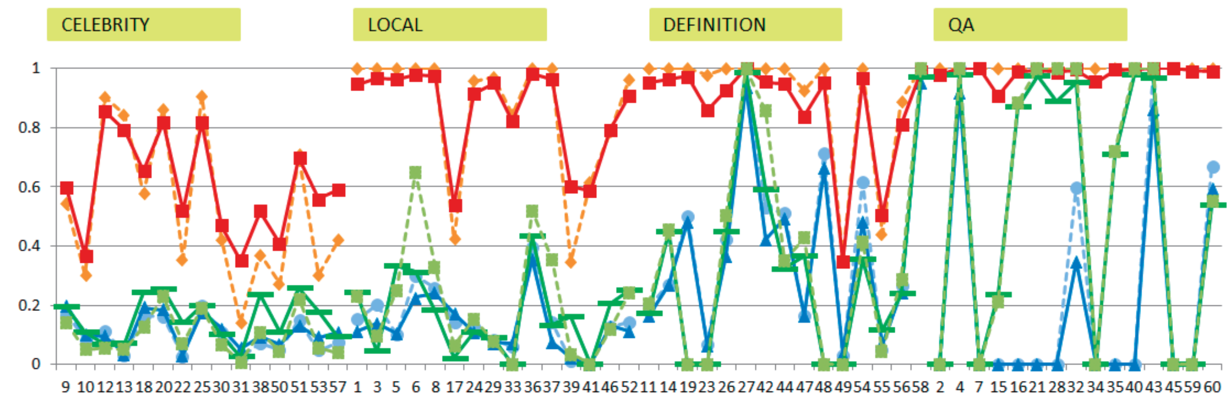

Figure 3: Per-query run performances: MSRA1click-2 S-measure/weighted recall (green solid/light green dotted lines); manual run S-measure/weighted recall (red solid/orange dotted lines); automatic run S-measure/weighted recall (blue solid/light blue dotted lines). The x-axis shows the query IDs

Table 3: Query Type Classification confusion matrix

		System Classification			
		CE	LO	DE	QA
Gold Standard	CE	15	0	0	0
	LO	0	15	0	0
	DE	3	0	11	1
	QA	0	0	1	14

Table 4: Query Type Classification Errors

Query ID	Query String	Actual type	Predicted Type
0023	カラシニコフ	DE	CE
0027	日本国民の三大義務	DE	QA
0048	一国一城令	DE	CE
0049	イカ天	DE	CE
0060	ホスピタリティ	QA	DE

n-gram candidates to the gold-standard nuggets in order to obtain training data for the ranking model, as described in section 3.4. Although this estimation is far from perfect, we believe that it is sufficient to grasp the overall trend of the modules' perfomance. We can gauge the precision of this estimation method by comparing the estimated weighted nugget recall for the summarizer and its corresponding actual value, which we show in the rightmost column of Table 5, with the actual value in parentheses. Apart from the moderate divergence in the scores for QA queries, we see that the estimated weighted nugget recalls generally resemble the actual scores.

Returning back to the document retriever's performance, we observe that while the document retriever has difficulty in retrieving relevant nuggets for LO queries, other query types have typically around 90% weighted nugget recall.

4.3.3 Candidate Generator Performance

Table 5: Estimated Weighted Nugget Recall for Document Retriever, Candidate Generator, and Summarizer

	Document Retriever	Candidate Generator	Summarizer (Actual)
CE	0.897	0.788	0.183 (0.136)
LO	0.584	0.453	0.270 (0.254)
DE	0.919	0.783	0.424 (0.409)
QA	0.898	0.697	0.437 (0.559)

The estimated weighted nugget recall of the candidate generator is shown in the middle column of Table 5. We can observe that the sliding-window based segmentation of the documents into candidates has a reasonably degrading effect on the integrity of the nuggets, with all of the query types experiencing a 10% to 30% drop in the weighted nugget recall. This highlights the weakness of a sliding window n-gram approach, one of which is where the window length might not be sufficient to fully cover a meaningful nugget.

4.3.4 Candidate Ranker and Summarizer Performance

As briefly touched upon in Section 4.2, the loss of performance from query expansion indicates that there are cases when the candidate ranker and summarizer fail to correctly rank relevant nuggets above non-relevant nuggets. One illustrative example is CE Query 0051 "田口ランディ" (Randy Taguchi, an essayist and writer), where the X-string for MSRA1click-1 (with query expansion) have 7 successful nugget matches while the X-string for MSRA1click-2 (without query expansion) have 18. The problem with the former output is that the candidates from query expansion documents contain multiple mentions of "生年月日" (birthdate) in contexts that have nothing to do Randy's birthdate. For example, the two highest ranking candidates are :

- "私の友人で、同じ生年月日の... 野沢さんが来てくれます" (My friend, Mr.Nozawa, who happens to have the same birthdate as me,... will come visit us), and
- "お名前・ご希望の日時・生年月日など、... 必要事項をご記入の上お申し込みください" (Apply by filling out your name, preferred date, birthdate ...)

This over-sensitivity to positive words was observed numerous times during analysis.

5. CONCLUSIONS

Microsoft Research Asia's participation in NTCIR-9 1CLICK Task focused on using a statistical ranking method and utilizing semi-structured web knowledge sources. Our analysis of per-module performance shows that the Query Type Classifier and Document Retriever performed relatively well, while the Candidate Generator, Candidate Ranker, and Summarizer have room for further improvement. Clearly, there are several weaknesses with the current system:

- Singularity of the generated candidates: there is an obvious weakness with the sliding window n-gram approach to generate candidates. As Zhou points out [12], a single n-gram window may contain several separate pieces of information, or may not even contain even a single piece of information. Developing a method to segment text by semantic units is critical, as subsequent steps heavily rely on the output of the candidate generator.
- Coherence of the generated sentence: chunks of n-grams candidates are indiscrimiately combined together. There is no guarantee that the X-string will be coherent to a human reader.
- Multiple intent queries: our system does not detect or distinguish multiple intent queries. For example, one of the formal run queries is "三本松駅" (Sanbonmatsu station). Because there are two train stations with the name "Sanbonmatsu" in Japan, it is possible for the X-string to contain a mix of information of these two separate locations, which could potentially be of harm to the user.

We will explore these issues in a future study.

6. REFERENCES

[1] J. Carbonell, and A. Goldstein. The Use of MMR and Diversity-Based Reranking for Reodering Documents and Producing Summaries. In *Proceedings of ACM SIGIR 1998*, pages 335-336, 1998.

[2] L. B. Chilton and J. Teevan. Addressing people's information needs directly in a web search result page. In *Proceedings of ACM WWW 2011*, 2011.

[3] A. Harnly, A. Nenkova, R. Passonneau, and O. Rambow. Automation of Summary Evaluation by the Pyramid Method. In *Proceedings of the Conference of Recent Advances in Natural Language Processing (RANLP) 2005*, 2005.

[4] T. Joachims. Training Linear SVMs in Linear Time. In *Proceedings of ACM KDD 2006*, 2006.

[5] J. Lin and B. Katz. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Processings of ACM CIKM 2003*, 2003.

[6] J. Lin and D. Demner-Fushman. Automatically Evaluating Answers to Definition Questions. In *Proceedings of HLT/EMNLP 2005*, 2005.

[7] A. Nenkova, R. Passoneau, and K. McKeown. The pyramid method: Incorporating human content selection variation in summarization evaluation. ACM Transactions on Speech and Language Processing, 4(2):Article 4, 2007.

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

[9] T. Sakai, M.P. Kato, Y.-I. Song. Click the Search Button and Be Happy: Evaluating Direct and Immediate Information Access. In *Proceedings of ACM CIKM 2011*, 2011.

[10] H. Suzuki. Phrase-Based Dependency Evaluation of a Japanese Parser. In *Proceedings of European Language Resources Association*, 2004.

[11] R. Varadarajan and V. Hristidis. A System for Query-Specific Document Summarization. In *Proceedings of ACM CIKM 2006*, 2006.

[12] L. Zhou, N. Kwon, and E. Hovy. A semi-automatic evaluation scheme: automated nuggetization for manual annotation. In *Proceedings of NAACL HLT 2007*, 2007.

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

[14] http://research.microsoft.com/en-us/projects/japanesenlp/