# Improving iUnit Retrieval with Query Classification and Multi-Aspect iUnit Scoring: The IISR System at NTCIR-11 MobileClick Task

Chia-Tien Chang[†]
Department of CSE
Yuan Ze University
Taoyuan, Taiwan
s1003325@mail.yzu.edu.tw

Yu-Hsuan Wu[†]
Department of CSE
Yuan Ze University
Taoyuan, Taiwan
s1003312@mail.yzu.edu.tw

Yi-Lin Tsai
Department of ISA
National Tsinghua University
Hsinchu, Taiwan
s102065514@m102.nthu.edu.tw

Richard Tzong-Han Tsai[*]
Department of CSIE
National Central University
Taoyuan, Taiwan
thtsai@csie.ncu.edu.tw

## ABSTRACT

This paper describes our approach to the NTCIR-11 MobileClick task. Based on the assumption that different user intentions should be handled by different extraction/retrieval strategies, we first classify each query into one of our eight defined query types and set the weights of the extraction methods accordingly. Next, we extract the relevant parts of the search results and rank the extracted sentences. Finally, we apply a rule-based approach to iUnit extraction. Our system achieves an nDCG@10 score of 0.2134 and a Q@10 score of 0.1573, outperforming the baseline by 23.9% and 42.4%, respectively. This difference demonstrates the effectiveness of our query classification and multi-aspect iUnit scoring.

## Team Name

IISR

## Subtasks

iUnit Retrieval Subtask (English)

## Keywords

Information Retrieval, Query Classification, Multi-aspect iUnit Score, MobileClick

## 1. INTRODUCTION

In this paper, we describe our approach to the MobileClick iUnit Retrieval Subtask. This subtask is to answer a mobile user's query with a concise summary of relevant search results, providing immediate and direct information access.

In our approach, we first categorize queries into eight types based on the assumption that different user intentions should be handled by different extraction/retrieval strategies. These eight types are based on work by T. Sakai et al.[5]. For each type we assume that the user needs the following information:

[†]equal contribution
[*]corresponding author

**WHY**: The reason or motivation for something.
**HOW**: How to do something or solve a problem.
**VERSUS**: A comparison of information on two or more things.
**PROS_AND_CONS**: The advantages and disadvantages of something.
**IMPACT**: The impact of an event and following developments.
**PEOPLE**: Facts about a person or celebrity as well as related events and organizations.
**LOCATION**: Information such as geographical position, street address, phone number, opening hours, etc.
**OTHER**: The remaining queries are classified as OTHER.

Depending on the query type, we use different rule-based iUnit extraction methods tailored for either tabular webpage content or body text. Finally, all extracted iUnits are output in order of iUnit relevancy score.

The reminder of this paper is organized as follows. Section 2 describes our methods and implementation. Section 3 describes the evaluation results and discusses error analysis. Section 4 concludes this paper.

## 2. METHOD

In this section, we describe our system, which consists of three modules: query classification, tabular iUnit extraction, and body-text iUnit extraction. Figure 1 shows a flow chart of our system.

### 2.1 Rule-based Query Type Classification

This module predicts the type of a given query. The types are WHY, HOW, VERSUS, PROS_AND_CONS, IMPACT, PEOPLE, LOCATION and OTHER.

The classification rules are described below:

1. If a query
   (a) starts with "Why",
   (b) or it contains the keywords "reason", "motivate", or other synonyms,

   it will be classified into WHY.

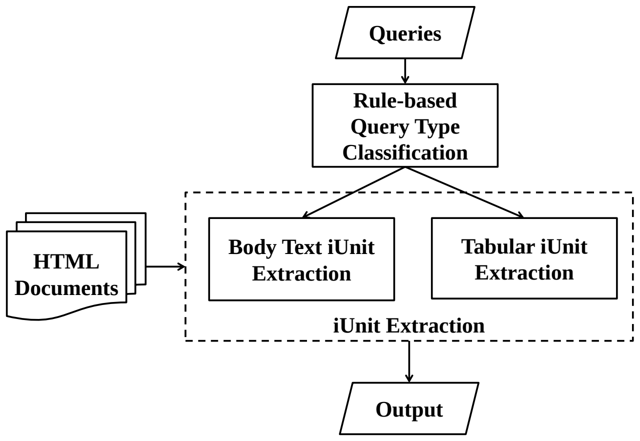

**Figure 1: System Flow Chart**

2. If a query starts with "How" or a gerund, it will be classified into HOW.

3. If a query contains "vs" or "difference", it will be classified into VERSUS.

4. If a query contains "benefit", "difficulty" or other synonyms, it will be classified into PROS_AND_CONS.

5. If a query contains the keywords "impact", "influence", "effect" or other synonyms, it will be classified into IMPACT.

6. If a query contains the name of a person from our list of celebrities, it will be classified into PEOPLE.

7. If a query contains a location from our location list, it will be classified into LOCATION.

8. Queries not belonging to any of the above types will be classified into OTHER.

For rules 1, 4 and 5, we construct near synonym sets according to search results from `Thesaurus.com`. The lists for rules 6 and 7 are based on pages from Wikipedia. Our PEOPLE list contains all person names under the category page "People" in English Wikipedia. Our LOCATION list comprises the countries, cities, counties, and states contained in the Wikipedia respective category pages, including abbreviations of place names.

## 2.2 iUnit Extraction

We found that the relevant search results for queries of different types appear in different HTML tags. Relevant iUnits for most query types are found in <body> text; however, iUnits for LOCATION queries often occur in HTML tables. Accordingly, we use separate extraction methods for tabular and body-text iUnits. For <body> text, we need to first score the relevance of each retrieved sentence before iUnit extraction. For tabular results, we can extract iUnits directly from table and list cells. For queries of the OTHER type, we combine both methods, as OTHER iUnits exhibit no regular characteristics.

### 2.2.1 *Body Text iUnit Extraction*

In this section, we describe the steps taken to extract iUnits from HTML web page <body> text. First, we filter out irrelevant elements of the HTML documents. Then, we rank the remaining sentences according to relevance. Finally, we extract iUnits use syntactic rules.

#### 2.2.1.1 *Filtering.*

We use the Apache OpenNLP[1] sentence splitter to separate sentences and filter out irrelevant sentences.

**Hyperlink Element Removal**

Roughly speaking, HTML pages contain two types of elements. The first hold text while the second contain hyperlinks to other web pages, delimited by <a> and </a> in HTML. Our goal is to extract the content of text elements, so we adopt Keyaki et al. [4]'s method of discarding hyperlink elements. Their method uses a ratio to determine if an element $e$ is text or hyperlink: the number of words in $e$'s <a> elements to the total number of words in $e$. If the ratio exceeds the threshold value $\tau$, $e$'s content is removed. In our system, we set the threshold to 0.4.

**Interrogative Sentence Filter**

Under the assumption that iUnits are found in declarative sentences, we filter out all interrogative sentences beginning with Wh- words, "be" verbs, or modal verbs (does, could, should) or ending with a question mark.

#### 2.2.1.2 *Sentence Ranking.*

We use the following methods to rank the remaining sentences according to relevance.

**Topic-based Scoring**

The latent Dirichlet allocation (LDA) model assumes that the words of each document arise from a mixture of topics, each topic being distributed over the vocabulary [1]. We use the LDA model to find the correlation between the given query and the words in a sentence.

We assume that a given query and its iUnits will be in the same topic. Our model learns five topics from each document (result page). The $n$-word sentence is a string of words $w_1$,..., $w_n$, $w_i$ is the $i^{th}$ word in a sentence, $P_{(i,w)}$ denotes the probability of word $w$ occurring in topic $i$, $Q_i$ is the $i^{th}$ token in a query, and $N$ means the number of words in a query. The score of topic $k$ in the given query is calculated as:

$$R_k = \frac{\sum_{i=1}^{N} P_{(k,Q_i)}}{\sum_{i=1}^{N}\sum_{j=1}^{5} P_{(j,Q_i)}}$$

The $TBS$ score of a given sentence $s$ is calculated by the following formula:

$$TBS(s) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{5}(R_j P_{(j,w_i)})}{n}$$

[1]`http://opennlp.apache.org`

**Similarity Scoring**

Assuming that sentences which exist in other similar sentences in different documents (search results) are more important, we apply the LexRank algorithm [2] to calculate the cosine similarity between each sentence. Following is the recursive calculation of LexRank:

$$p = [dU + (1-d)B]^T p$$

where $p$ is a vector, each element $p_i$ means the importance of the $i^{th}$ sentence in all documents, and $d$ is a dumping factor. $B$ is the similarity matrix. Each element $b_{i,j}$ corresponds to the cosine similarity between sentence $i$ and $j$. $U$ is a square matrix with all elements equal to $1/N$. $N$ denotes the total number of all sentences.

After $p$ convergence, the score of the $i^{th}$ sentence $s$ is calculated by:

$$ContentSim(s) = p_i$$

**TF-IDF Scoring**

If a sentence contains more query terms, it should be assigned with a higher score. If a query term appears in many sentences, it is ineffective to be used for distinguishing sentences. We use Lucene[2] to implement this method.

**Search Rank Scoring**

This weight refers to the original order returned by Bing. Top pages are more likely to contain iUnits. Therefore, this score is designed as follows.

$$SearchRank(s) = \frac{1}{log[Rank_{search}(s) + 1]}$$

Where $Rank_{search}(s)$ means the search rank of the page including the sentence $s$. We rank the sentences using $Score(s)$, which multiplies together the scores from the above methods.

$$\begin{aligned} Score(s) = TBS(s) \times ContentSim(s) \times \\ TF\text{-}IDF(s) \times SearchRank(s) \end{aligned}$$

### 2.2.1.3 *iUnit Extraction and Filtering.*

For iUnit extraction, we select 2,000 sentences with higher score from the sentences we ranked in section 2.2.1.2 as candidate sentences. We use full parser provided by the Stanford NLP[3] to identify clauses and separate the clauses by a comma or a semi-colon. Then, we employ the following strategies to extract iUnits for different query types.

1. For query type WHY, clauses not led by "because", "due to" or their synonyms are removed.
2. For query type HOW, clauses contain "the way" or the words of the given query are extracted.
3. For query type VERSUS, if a query contains "A vs B" or compares something between A and B, we extract clauses which contain both A and B, or a comparative adjective in it. For example, for the query "java vs python text processing", a clause containing "java and python" or "java is better" or "python is worse" will be extracted.
4. For query type PROS_AND_CONS, clauses contains "benefit", "difficulty" or their synonyms are extracted.
5. For query type IMPACT, clauses contains "impact", "influence", "effect" or their synonyms are extracted.
6. For query type PEOPLE, we first separate the given query into two parts: name and other. Clauses containing both are extracted.
7. For query type OTHER, clauses containing the last word of the given query are extracted.

[2]http://lucene.apache.org/
[3]http://nlp.stanford.edu/software/lex-parser.shtml

### 2.2.1.4 *iUnit Scoring.*

All extracted clauses are treated as iUnits. For each iUnit $u$, we assign a score according to their length, relevance to the given query and term frequency.

**Length Score**

In order to provide succinct information to users, shorter iUnits get higher scores. We define the length score as follows:

$$Len(u) = \begin{cases} 2, & u\text{'s length is between 2 and 10 words} \\ 1, & u\text{'s length is between 11 and 15 words} \\ 0 & \text{otherwise} \end{cases}$$

**Query-term-existence Score**

An iUnit is more likely to be relevant to the given query if it contains at least one non-stopword query term. We define the $Qte$ score as follows:

$$Qte(u) = \begin{cases} 2, & u \text{ contains a non-stopword word in query} \\ 0 & \text{otherwise} \end{cases}$$

**Frequency-of-common-terms Score**

For a given query, all correct iUnits may have common keywords. For instance, "L-Tryptophan" is a keyword to the query "why does turkey make you sleepy". We extract the 20 most frequent noun phrases (as tagged by parser) in all extracted clauses.

$$Fct(u) = \begin{cases} 3, & u \text{ contains one of 20 noun phrases} \\ 0 & \text{otherwise} \end{cases}$$

We then sum all the above scores to get the final score of an iUnit. If the score exceeds 5, a score of 5 is given to that iUnit.

$$TotalScore(u) = Len(u) + Qte(u) + Fct(u)$$

### 2.2.2 *Tabular iUnit Extraction*

For tabular iUnit extraction, we first retrieve the web pages for the top 15 search results. We then extract the information in all tables on these pages, considering each row an iUnit. We score these iUnits according to the following rules:

**Table 1: Performance of baseline and our system**

| TeamID | RunID | nDCG@5 | nDCG@10 | nDCG@80 | nDCG@400 | Q@5 | Q@10 | Q@80 | Q@400 |
|---|---|---|---|---|---|---|---|---|---|
| NUIR | 1 | 0.1834 | 0.1723 | 0.1328 | 0.1224 | 0.1440 | 0.1105 | 0.0350 | 0.0293 |
| NUIR | 2 | 0.1083 | 0.1011 | 0.0694 | 0.0608 | 0.1019 | 0.0836 | 0.0229 | 0.0163 |
| NUIR | 3 | 0.1195 | 0.1073 | 0.0726 | 0.0655 | 0.1367 | 0.0946 | 0.0273 | 0.0222 |
| IISR | 1 | **0.2197** | **0.2134** | **0.1929** | **0.1809** | **0.1892** | **0.1573** | **0.0647** | **0.0546** |

**Tabular Length Score**

In order to provide succinct information to users, shorter iUnits get higher scores. We define the tabular length score as follows:

$$tLen(u) = \begin{cases} 2, & u\text{'s length is between 1 and 3 words} \\ 1, & u\text{'s length is between 4 and 6 words} \\ 0 & \text{otherwise} \end{cases}$$

**Tabular Query-term-existence Score**

For tabular iUnit extraction, a keyword in a query is a very important indicator to show the relevance between iUnits and the query. For instance, a iUnit contains the location itself or the last word of the query, might be more important to a given query. We define the *tQte* score as follows:

$$tQte(u) = \begin{cases} 3, & u \text{ contains a keyword word in query} \\ 0 & \text{otherwise} \end{cases}$$

**Tabular Frequency Score**

The more frequently an iUnit appears in tables, the more likely it is to be relevant to the given query. We design a tabular frequency score in accordance with the above observation:

$$tFre(u) = \begin{cases} 2, & u \text{ apears more than 3 times} \\ 1, & u \text{ apears 1 to 2 times} \\ 0 & \text{otherwise} \end{cases}$$

To calculate the final score for a tabular iUnit, we sum the scores as in body-text iUnit extraction. If the score exceeds 5, a maximum score of 5 is given to that iUnit.

$$tTotalScore(u) = tLen(u) + tQte(u) + tFre(u)$$

## 3. EVALUATION

In this section, we report our results on the MANDATORY run in the NTCIR-11 MobileClick task. We submitted one run for the iUnit Retrieval Subtask.

We employ the normalized versions of DCG (i.e. nDCG) and Q-measure as our primary evaluation metrics, computing them for different cutoff thresholds $k$. The two metrics are based on the following two principles: (1) the more gold standard iUnits (GiUnits) a run ranks highly, the higher a score the run achieves; (2) redundant GiUnits do not improve the score[3]. Table 1 shows the overall nDCG and Q-measure score at our run (IISR-1) and baseline (NUIR). Our system achieves an nDCG@10 score of 0.2134 and a Q@10 score of 0.1573, outperforming the baseline by 23.9% and 42.4%, respectively.

Figure 2 and 3 show the mean nDCG and the mean Q-measure scores at different thresholds for our eight query types. In both evaluation metrics, we have poor performance on PROS_AND_CONS and IMPACT types. For these two types, our extraction strategies are not thorough enough–we extracted a limited number of keywords. For example, in the query "mechanical keyboard benefits" (in type PROS_AND_CONS), the iUnits such as "more accurate", "longer keystroke typing", and "sturdier keyboard" cannot be extracted since no keywords can be matched in these clauses. We believe we could improve performance in such cases by extracting comparative adjectives, as we did for the VERSUS type.

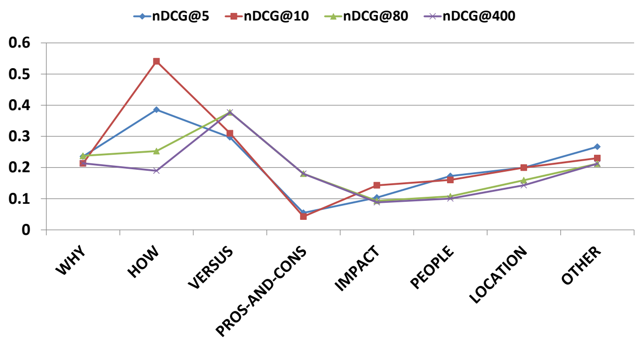

**Figure 2: Mean nDCG at different thresholds for our eight query types**

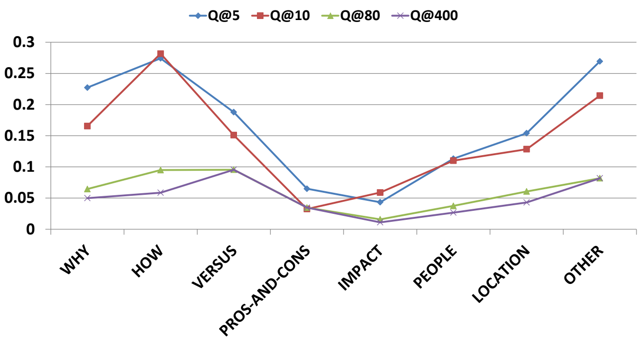

**Figure 3: Mean Q-measure at different thresholds of our eight query types**

We received zero nDCG and Q-measure values for queries 6, 11, 29, 41 and 47. Comparing the iUnits extracted by our system (CiUnits) for these five queries with the corresponding gold standard iUnits (GiUnits), we found our CiUnits were shorter than the GiUnits on average. Table 2 shows the average length. Since we want to provide succinct information to users, one of our score aspects gives a higher rating to shorter iUnits. However, in the five case above, our shorter CiUnits did not contain enough information. According to the organizer's assessment rules, an

**Table 2: The average length (in words) of the extracted iUnits and the Gold Standard iUnits (GiUnits)**

| Query ID | Query | Query Type | Average Length of our iUnits | Average Length of the GiUnits |
|---|---|---|---|---|
| MC-E-0011 | aaron rodgers belt celebration | PEOPLE | 6.26 | 12.16 |
| MC-E-0029 | government shutdown financial impact | IMPACT | 9.96 | 33.27 |
| MC-E-0041 | theaters texarkana | LOCATION | 3.34 | 5.57 |
| MC-E-0047 | pope francis humility | PEOPLE | 9.23 | 26.62 |

**Table 3: iUnit examples for query 6 "ron paul tea party"**

| | |
|---|---|
| the extracted iUnits with the highest scores (score of 5) | (1) Ron Paul is correct<br>(2) We all love Ron Paul<br>(3) Ron Paul is not the tea party leader |
| GiUnits with the highest scores (score of 3) | (1) Paul attended the Tea,Party Republican Presidential debate broadcasted by CNN<br>(2) Ron Paul speaks at,the Tea Party Express rally at the Capitol in Austin, Texas, on Sunday May 6, 2012<br>(3) Tea party audience boos Ron Paul for explaining motive of al Qaeda |

extracted iUnit that does not contain any GiUnit will get no score [3]. Table 3 shows a few examples for query 6, "ron paul tea party".

## 4. CONCLUSIONS

We describe our approach to the NTCIR-11 MobileClick task in this paper. Our approach comprises Query Type Classification and iUnit Extraction. For Query Type Classification, the given queries are classified into eight categories. In iUnit Extraction, we first separate extraction methods for tabular and body-text iUnits depending on the category of each query. We extract and rank the sentences in HTML documents. To generate the final results, we extract the iUnits according to their relevancy score.

## 5. REFERENCES

[1] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[2] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[3] M. P. Kato, M. Ekstrand-Abueg, V. Pavlu, T. Sakai, T. Yamamoto, and M. Iwata. Overview of the ntcir-11 mobileclick task. In *Proceedings of the 11th NTCIR Conference*, 2014.

[4] A. Keyaki, J. Miyazaki, K. Hatano, G. Yamamoto, T. Taketomi, and H. Kato. Xml element retrieval@ 1click-2. In *Proceedings of the 10th NTCIR Conference*, 2013.

[5] T. Sakai, M. P. Kato, and Y.-I. Song. Overview of ntcir-9 1click. *Presentation Slides, Dec*, 2011.