An API-based Search System for One Click Access to Information

Dan Ionita
University of Twente
Enschede, The Netherlands

Niek Tax
University of Twente
Enschede, The Netherlands

Djoerd Hiemstra
University of Twente
Enschede, The Netherlands

ABSTRACT

This paper proposes a prototype *One Click* access system, based on previous work in the field and the related 1CLICK-2@NTCIR10 task. The proposed solution integrates methods from previous such attempts into a three tier algorithm: query categorization, information extraction and output generation and offers suggestions on how each of these can be implemented. Finally, a thorough user-based evaluation concludes that such an information retrieval system outperforms the textual preview collected from Google search results, based on a paired sign test. Based on validation results possible suggestions on future improvements are proposed.

Team Name

UT

Subtasks

Query Classification (English)

Keywords

query classification, information extraction, API

1. INTRODUCTION

Current web search engines usually return a ranked list of URLs in response to a query. The user often has to visit several web pages and locate relevant parts within long web pages. However, for some classes of queries, the system should be able to gather and return relevant information directly to the user, to satisfy her immediately after her click on the search button (”one click access”).

In a normal search system, a user is expected to (1) enter a query, (2) click the search button, (3) scan a ranked list of URLs, (4) Click some URL and (5) repeat step 3 and 4 until all desired information is gathered. By contrast, in ”One Click Access” system after (2) clicking the search button, all desired information is returned in a single, concise textual result.

The goal is to design, implement and evaluate a One Click access system capable of participating in the 1CLICK-2@ NTCIR10 task.

1.1 Motivation

A *1CLICK* system is a search engine that requires no user operation between clicking the SEARCH button and seeing the output directly. Another essential aspect of this type of system is that the user has to spent to locate relevant information, as relevant information is generated and displayed in the result. In contrary, a regular search engine requires the user to visit one or several websites from the search results page, which could lead to information overload [19].

It can be concluded that such a system, given a query and an output window of size X, returns a text (Xstring) that:

- Presents important pieces of information (a.k.a. *nuggets*) first
- Minimizes the amount of text the user has to read (or the time spent reading it)

Huffman and Tokuda [10] identified the values of 1CLICK results, especially on mobile devices. First, opening web pages on a mobile device is often slow and clunky, with formatting/usability issues and omissions. Second, in order to quickly answer simple questions that had just come up in a conversation. Third, the ability to provide specific, localized information instead of general search results. Furthermore, it suggests that there is a big opportunity for search engines to directly address user’s information needs more often by providing the right information on the result page for certain types of information needs, especially in mobile search.

One click access mechanisms are currently implemented within major web search engines like Google and Microsoft Bing in addition to regular ranked search results. E.g. if a user searches for the query ”Beijing weather”, most search engines show Beijing’s local weather forecast information on top of the results. Google implemented a similar mechanism called ”Google Search Features” which displays a direct response to some query categories. *Answers*, as [1] refers to these direct answer mechanisms, have showed to be used repeatedly by some share of Bing web search users. However, [1] also shows that *answers* may not always address every user’s intent. One click systems should aim to satisfy the user with a single textual output, immediately after the user clicks on the search button.

1.2 Research Question

The main question of this research is ”What is the best approach to design a One Click Access Information Retrieval system?”. This main research question follows directly from the 1CLICK task goal.

1.2.1 Sub Research Question

For the 1Click task, systems are expected to present important pieces of information first, which are different for different types of queries. A classification of the query needs to be made in order to know what information is most relevant for a given query. After classification, relevant pieces of information (or *nuggets*) need to be collected for the query. As found by [2], an information nugget consists of a relevant property of a query subject together with the value of this property for the subject of a query. When all possible nuggets have been retrieved, the nuggets need to be ranked in order to select the most relevant nuggets to display in the results.

This leads to the following subquestions that need to be solved in order to design an One Click Access Information Retrieval system.

- How can query classification be accomplished in a one click environment?
- How can we extract nuggets for the query from the world wide web in a complete manner?
- How can we decide which nuggets need to be selected for display from the collection of nuggets?
- How can the selected nuggets of information be formatted as a single textual output (X-string) that contains important pieces of factual information in a minimum amount of text?

In section 2 we describe related work. As this is the second edition of the 1CLICK challenge, and no similar systems are commonly available, we will focus on the work of last year's participants ([12], [9] and[11]). Carefully analyzing the evaluation methods as described by [14] and [13] allows us to draw valuable conclusions regarding the desired specifications of our prototype system.

We then describe the iterative approach used to design the system. Once the 1CLICK system is designed and specified, a prototype implementation will be created. This will be used in a user study to validate our approach by comparing it with a naive baseline.

2. RELATED WORK

To gain insight in possible architectures and methods in One Click search engine design, we focus our literature research on the literature related to One Click Access systems condensed around The Virtual Sakai Laboratory and the experimental implementations designed by the participants of 1CLICK@NTCIR-9.

2.1 1CLICK@NTCIR-9 Participants

The 1CLICK Task aims to development of general purpose 1Click systems which are able to process different categories of queries, in contrary to existing research which mainly focused on Q&A query handling [6, 3, 18]. In order to properly analyze the literature of the previous participants in the 1CLICK Task of last year, the concept of nugget must be properly introduced. A *nugget* contains an atomic factual statement as well as some identification and bookkeeping attributes [14]. According to the presentation given by Dr. Sakai as overview of 1CLICK@NTCIR-9, [15] nugget record can also contain an id, a source URL, a *vital string* (i.e. the string in its minimal string representation) and an attached weight (i.e. importance).

The 1CLICK task organizers predefined four query categories: CELEBRITY, LOCAL, DEFINITION and QA. The first three query types are simple phrases, while the QA queries are single sentences [16]. Although 25 teams originally registered, only three actually submitted their runs, and a paper detailing their approach. The three teams are:

1. Kyoto University: *Information Extraction* based Approach [9];
2. Tokyo Institute of Technology: *Summarization* based System [11];
3. Microsoft Research Asia: *Passage Retrieval approach* [12].

Next we will analyze these approaches in retrospect by taking into account their obtained results and the relative scores achieved in the evaluation.

2.1.1 Information Extraction Based Approach

[9] proposes an approach in which query classification is done by generating an feature set of eight features using natural language processing tools and third-party services (e.g. Yahoo!Answers, Wikipedia and Yahoo!WebSearch). Using a a multi-class support vector machine the query is classified into one of the predefined categories. An IE (i.e. Information Extraction) method that is best suited for the particular category is selected and employed. Its output is then aggregated into a short text. The IE methods are predefined for each category and attempt to exploit empirical assumptions about each type of query. The system obtained an S-measure of 0.381 for the desktop run and 0.273 for the mobile run (on their best set of IE methods), which ranked them first of the three participants.

2.1.2 Summarization Based Approach

The approach described in paper[11] uses the integer linear programming technique to achieve text summarization. According to [11], two different implementations are attempted:

Abstractive summarization model adopts QSBP (Query SnowBall with word Pair) for query-oriented summarization and extends the method to abstractive summarization in order to recognize and extract the parts of a sentence related to the query. With an S-measure of 0.1585 on the desktop run and 0.0866 on the mobile run this approach scored lower than other participants.

Model for complex queries focuses on ensuring coverage of various predefined aspects of the users information need. The types of aspects are selected depending on the query category. It achieved 0.1484 S-measure on the desktop run and 0.0829 on the mobile run, just a bit lower that the first one.

2.1.3 Passage Retrieval approach

[12] describes two different techniques: a statistical ranking approach for queries of type and the utilization of semi-structured web knowledge sources. The system categorizes the query into one of the predefined types using a SVM and generated features involving natural language processing tools and online services. Depending on the discovered query type, a different method is employed:

Statistical ranking method A *Document Retriever* collects relevant text from various web pages. The*Candidate Generator* extracts candidate text from the retrieved documents, which are then ranked by a *Candidate Ranker* module using a machine learning approach and fed to a *Summarizer* which merges ranked information into textual output.

Utilization of semi-structured web knowledge sources Wikipedia and Yahoo!Answers are used in order in order to answer DEFINITION and QA queries by returning the Wikipedia abstract of the closest matching page for DEFINITION queries and a concatenation of the answers to the closest matching Yahoo!Answers question for QA queries. If no match is found in either sources, the system falls back to the statistical method.

Both methods described above have shown to increase recall and S-measure compared to a baseline system. The biggest improvement was noticed for QA queries, but disappointing results were achieved for DEFINITION queries, probably because of bad Wikipedia matches. The resulting average S-measure of the system was 0.329 for the desktop run which places it on second place, but very close to the IE based approach. This can be explained by their similar approach and usage of the same external resources.

2.2 Other Related Work

Chilton and Teevan [1] describe the following general challenges a search engine faces in trying to meet an information need directly on the page: Firstly, in a hybrid system, inline answers can cannibalize clicks from the ranked results. Secondly, they observed that in such a system, the user's repeated search behavior could be used to understand the provided information's value. Furthermore, it looks at the behavior of users of such systems and points out some advantages of the absence of interaction.

Li et al [10] introduce the concept of Good Abandonment as an abandoned query for which the user's information need was successfully addressed by the search results page, with no need to click on a result or refine the query. It goes on to explore how this phenomenon manifests itself in various classes of queries and serves as an inspiration for the query types predefined as part of the 1CLICK2@NTCIR-12 task.

3. DESIGNING THE PROTOTYPE SYSTEM

Our prototype system consists of a query classification and an information extraction module, which are described in detail in the following sub-sections.

3.1 Classifying Queries

In order to classify queries we decided to use a feature-based machine learning program, similar to the one designed by team KUIDL at the NTCIR-9 1CLICK Task [9]. In order to design the classifier, we opted for an iterative approach which allows us to identify the best algorithm and feature-set for the job. The following sub-sections describe the process.

3.1.1 First Version

A combination of features used by 1CLICK@NTCIR9 participants was used as a naive first try for our classifier feature-set. This feature set consisted of the following features.

Query length.

[12] showed that different query types show different query length distributions, e.g., where short queries generally occur for the person-like query types (athlete, politician, artist and actor), longer queries are generally used for the Q&A query type. Based on this idea we define a query length feature which takes the length of the query in characters with a minimum of five characters.

Has wikipedia.

[9] proposes a binary feature indicating whether a Wikipedia[1] article matching (this query exists.

Sentence pattern.

[9] introduces a feature checking for words like 'who', 'when', 'where', 'what', 'how', 'which' and the symbol '?' in the query. These words and symbols will occur more often in Q&A type queries compared to the other query types. For each of these words and the question mark symbol a binary feature indicating its presence in the query text is used.

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

Appearance of clue words.

[12] describes a feature where the appearance of some words that are often present in query of a particular type are used as binary feature. A binary feature for presence of the words 'west', 'north', 'east', 'south', 'street', 'road' is used. These words all occur often in *geo* queries.

3.1.2 Performance of the first version

A multi-class algorithm is needed to classify into the eight classes provided for the 1CLICK@NTCIR task. We used the WEKA suite [4] to test the accuracy of various such algorithms. The multilayer perceptron classifier was chosen, as it showed to give the best classification accuracy of the multi-class classification algorithms in WEKA. A stratified 10-fold cross validation method is used to validate performance of the classifier, taking a manually created query sample set consisting of thirty queries per type as input. The first try scores 38.3% correctly classified instances, which is insufficient. The Confusion matrix (table 1) shows how our queries were classified.

Table 1: Confusion matrix - first classifier version

a	b	c	d	e	f	g	h	<– classified as
0	7	4	1	0	3	8	0	a = ARTIST
2	7	6	4	0	3	8	0	b = ACTOR
1	7	7	5	4	2	4	0	c = POLITICIAN
1	9	4	4	1	3	8	0	d = ATHLETE
0	3	8	1	6	7	5	0	e = FACILITY
0	0	0	1	0	29	0	0	f = GEO
0	6	3	2	3	1	15	0	g = DEFINITION
0	0	0	0	0	6	0	24	h = QA

From analyzing the confusion matrix the following conclusions can be drawn.

- *Geo*, *Definition* and *Q&A* queries are classified well.
- *Person* query types are indistinguishable from each other.
- *Person* query types are often classified as *Definition.*
- *Facility* queries are often classified as one of the *Person* query types, or as *Definition* or *Geo.*

3.1.3 Second version

Based on the findings from Table 1, several new features have been designed and added to improve the weaknesses of the first version of the classifier.

Person/Facility classifier.

Two of the problems identified based on the confusion matrix of the first try classifier two of the conclusions were:

- *Person* queries are often classified as *Definition.*
- *Facility* queries are often classified as one of the *Person* queries, or as *Definition* or *Geo.*

In order to solve these problems the Person/Facility classifier feature has been introduced. The Person/Facility classifier feature retrieves the Wikipedia document that matches the query or retrieves the first Microsoft Bing search-result document when there is no Wikipedia page available for the given query. For each person-like query type and for the facility query type a small word database is constructed which contains words that often appear on pages for query of that specific query type, e.g., this database contains words like 'artist' and 'song' for the *Artist* query type. With Lucene [5], for each of the *Person* query types and the *Facility* query type, the amount of words in the retrieved document that are in the word database for this query type were counted. The amount of 'athlete'-words, 'facility'-words, etc., are turned into a ratio of the total amount of matched words and inserted as classifier features.

Percentage of real words in query.

The first try classifier had problems distinguishing *Geo* and *Facility* queries. To increase performance in this area the percentage of 'real words' in the query is used as feature. Here a 'real word' is defined as a word which is present in an English word list.

Yahoo! placeTypeName.

Distinguishing *definition* queries from *facility* queries failed in the first try classifier. These query types share a lot of characteristics: both query types often contain two- or three-word queries and both consist mainly of words that can be found in a dictionary. The Yahoo! GeoPlanet API return a placeTypeName for each query, indicating whether the query matches a 'Point of Interest', 'Town', 'County', etc. When the query does not match a location no placeTypeName is returned. Adding the placeTypeName from the Yahoo! GeoPlanet API a feature will help identifying *Facility* queries as these queries will often have a value of 22 (point of interest) for this feature.

3.1.4 Performance of the second version

Adding discussed features increased the classification accuracy from 38.3% to 78.8%. The confusion matrix (table 2) shows that classification improved for each query type, except for *Geo* queries for which the first try system already preformed well. The main classification problem are *Facility* queries, which is often classified as *Geo* queries. The difficulties in distinguishing between *Geo* and *Facility* queries can be explained by the difficulties that even human might have in distinguishing between those query types. The query 'san diego zoo' illustrates this; the creator of the query could either have meant the historical monumental zoo in San Diego (*Facility*) or could have asked for a zoo in San Diego (*Geo*).

Table 2: Confusion matrix - second classifier version

a	b	c	d	e	f	g	h	<– classified as
22	1	2	0	0	3	2	0	a = ARTIST
1	26	0	0	0	3	0	0	b = ACTOR
0	0	26	0	1	2	1	0	c = POLITICIAN
3	0	0	25	0	1	1	0	d = ATHLETE
2	0	1	0	16	7	4	0	e = FACILITY
1	0	0	0	0	29	0	0	f = GEO
2	0	1	2	3	0	21	0	g = DEFINITION
0	0	0	0	0	6	0	24	h = QA

3.2 Retrieving the relevant information

This section describes the methods used to extract relevant information for queries of the query types. An approach similar to[9] is used, the similarity being that we use a different set of Web resources and information extraction methods for each category.

3.2.1 Person *query type*

For person-like queries (i.e. *Artist*, *Actor*, *Politician* and *Athlete*) we follow a similar information extraction method:

- The first paragraph of the matching Wikipedia page is extracted and compared to the summary by the *Freebase TEXT API.* The Freebase API offers a summary of the first few paragraphs of the Wikipedia page. However, for some particular queries, the summary is too short, incomplete, or missing. Therefore, the longest of the two strings is chosen and appended to the output.
- The Wikipedia InfoBox of the matching Wikipedia page is analyzed and attribute-value pairs are extracted, formatted and appended to the result.

3.2.2 Facility *query type*

For queries that have been categorized as being of type *Facility* we have implemented a method that makes use of three web services: *Yahoo!GeoPlanet API*, *Google's Places API* (Place requests and Place Detail requests) and emph-WalkScore's Public Transit API in the following way:

1. First the full query is issued to the*Yahoo! GeoPlanet API.* The Yahoo API identifies the part of the query that describes a location and returns its latitude/longitude **coordinates**.
2. Coordinates are then issued in a *Google Place* request as location parameters, combined with the query as keywords. Returned place names are compared to the original query using Levenshtein distance and the **Google reference (ID)** of the most similar result is selected.
3. The ID number is then used as a parameter for a *Google Place Detail request.* This final request is used to gather as much relevant information about the particular facility (eg. full address, postal code, phone number, website, etc.).
4. Finally, the coordinates from step 1 are issued as location parameters in a request to the *Walk Score Public Transit* API to retrieve nearby public transit stops, their distance and lines services.

This approach was used as the Google API returns richer information about places, but requires lat/long coordinates, which the Yahoo API is great at finding. As neither Google nor Yahoo offer public transportation directions, we chose Walk Score's API to retrieve these.

3.2.3 Geo *query type*

Once a query has been categorized as *Geo* it will go through a process similar to the one describe above for *Facility* queries, only with a few key differences:

1. The query is issued to the *Yahoo! GeoPlanet API* to retrieve an approximate geolocation
2. The query together with that location's coordinates are issued as a *Google Place request.* Instead of comparing all nearby matches against the query string, we let Google relevance ranking create a list of places (limited to 15 results).

3. The Google reference (ID) of each of the previous results is passed on as a parameter for a *Place Details* request, this collects more information on more possible matching places nearby. All this information is then formatted and returned as output. Geo type queries do not get public transit information appended, as result quantity is prioritized over result quality.

3.2.4 Definition *query type*

Queries that have been categorized as being of type *Definition* are answered using two Web resources (the *Definitions.net API* and *Wikipedia*) in the following manner:

1. The query is forwarded to the Definitions API, which retrieves available definitions from multiple on-line dictionaries. The response is then formatted as a **definition**
2. The result from the previous step is further enriched by concatenating the **first paragraph** of the closest matching Wikipedia article.

3.2.5 QA *query type*

In order to answer queries that have been categorized as *QA* (Question & Answer) we use the Yahoo!Answers API. We simply forward the query to the Yahoo!Answers, select the highest rated response and return it as the answer.

4. VALIDATING THE PROTOTYPE SYSTEM

4.1 Validation Methodology

A user study has been conducted to verify the prototype systems performance of the prototype system. The purpose of the validation is to two-fold:

- To asses the utility of the 1-Click system compared to a popular search engine (e.g. Google)
- To get an impression on the performance of the system on the NTCIR 1CLICK task.

We chose to compare our system to a naive baseline. By means of a simple GUI, users were asked to enter a query, then they were shown both outputs (prototype and baseline) and asked to choose the best output. The outputs are randomly positioned on the left/right side of screen to avoid biased evaluations. This evaluation method is based on the concepts described by Thomas et al [17].

We chose to implement the baseline by simply retrieving the result snippets on the first Google result page for the query, to get an impression of the systems usefulness compared to more traditional web search tools. This naively simulates using Google as a One Click search engine and compares its utility with our implementation.

The system was designed with the 1CLICK task in mind, using the instructions issued by the organizers, therefore we tried to keep the evaluation close to the released NTCIR evaluation guidelines.

- Users were given a simple instruction sheet extracted from the 1CLICK task instructions and result evaluation criteria.
- Users were asked to try entering a query for each of the provided query categories.

The paired sign test will be used to draw conclusions on the prototype systems performance compared to the baseline system. It will be assumed that the data holds a binomial distribution.

- H0: $P = 0.5$
- H1: $P <> 0.5$

Where P is the chance of the prototype system performing better compared to the baseline system. The hypotheses are two-tailed. A statistical significance level of 0.05 will be used.

4.2 Validation Results

We conducted the study with 22 users, which entered a total of 169 queries. After analyzing the results of the evaluation, we conclude the prototype One Click system to perform better than the Google baseline in answering queries directly. For 68% of the queries users chose our system over the baseline. More detailed results (per category) are available in table 3.

Category	Preferred system				Total	
	Baseline		Our System			
	#	%	#	%	#	%
ACTOR	4	18.18	18	81.82	22	100
ARTIST	1	04.55	21	95.45	22	100
ATHLETE	4	20.00	16	80.00	20	100
POLITICIAN	6	28.57	15	71.43	21	100
FACILITY	10	47.62	11	52.38	21	100
GEO	6	28.57	15	71.43	21	100
DEFINITION	5	23.81	16	76.19	21	100
QA	18	85.71	3	14.29	21	100
TOTAL	54	31.95	115	68.05	169	100

Table 3: Evaluation results: prototype system compared to the baseline

4.2.1 *Significance Tests*

Paired sign test results are listed in Table 4 and conclusions on significant differences between prototype system and baseline are summarized in table 5

4.3 Discussion of Validation Results

It is observable from table 3 that some categories performed very well, while others did not. Conclusions on the systems performance can be drawn for each category.

- The system performed well on most *Person* categories (*Artist*, *Actor*, and *Athlete*), with the best results for *Artist* queries. This is mainly due to the fact that Wikipedia is an excellent source of human selected and synthesized information. The first paragraph of each Wikipedia page presents a concise summary of the entire article. Furthermore, the InfoBox is a great source for more structured information. Because of Wikipedia's review system, only the most relevant information for each topic is included in the summary and InfoBox. Thus, the system does not have to do any work on selecting, ranking or summarizing information for any of the person-type queries. Another observation is that even in case of a misclassification amongst *Person* queries, the returned result is still relevant because the method used for any of these is the same. However, as the length of the output is limited and most times the extracted information does not fit within this limit, we think there is still room for improvement by further trimming the outputs.

Category	System	N	Prop.	Test Prop.	Sig.
All	Prototype	115	0.68	.50	.000[a]
	Baseline	54	0.32		
	Total		169	1.00	
Actor	Prototype	18	.82	.50	.004
	Baseline	4	0.18		
	Total		22	1.00	
Artist	Prototype	21	.95	.50	.001
	Baseline	1	0.05		
	Total		22	1.00	
Athlete	Prototype	16	.80	.50	.012
	Baseline	4	0.20		
	Total		20	1.00	
Politician	Prototype	15	.71	.50	.078
	Baseline	6	0.29		
	Total		21	1.00	
Facility	Prototype	11	.52	.50	1.0
	Baseline	10	0.48		
	Total		21	1.00	
Geo	Prototype	15	.71	.50	.078
	Baseline	6	0.29		
	Total		20	1.00	
Definition	Prototype	16	.76	.50	.028
	Baseline	5	0.24		
	Total		21	1.00	
QA	Prototype	3	.14	.50	.002
	Baseline	18	0.86		
	Total		21	1.00	

Table 4: Paired sign test for actor queries

Category	Result
Overall	Prototype > baseline within significance level
Actor	Prototype > baseline within significance level
Artist	Prototype > baseline within significance level
Athlete	Prototype > baseline within significance level
Politician	No significant difference in performance
Facility	No significant difference in performance
Geo	No significant difference in performance
Definition	Prototype > baseline within significance level
QA	Baseline > prototype within significance level

Table 5: Comparison between prototype and baseline system

- The system performed average on *Facility* queries, with the baseline scoring approximately as many good hits as the prototype system. We believe there are three main reasons which caused these issues. First, some *Facility* queries tend to be mis-classified and thus return an empty or completely irrelevant result. Second, the API we use to retrieve nearby bus-stops and lines (i.e. WalkScore's Public Transit API) only works for places located in the US and thus returns nothing for any other locations. Finally, it seems that Google's geolocation services do not always cooperate well with Yahoo; They sometimes offer slightly different coordinates for the same location, sometimes resulting in weird or less relevant output.

- The prototype system performed very bad for *QA* queries. We suspect this is simply due to the low quality of the response retrieved by Yahoo! Answers API. This is due to the fact that some questions simply have not been asked recently, or have been but in a different form. Even if a closely matching question is found, the answers are sometimes missing, or plain wrong.

4.4 Fine-tuning the System

Using the conclusions drawn form the validation, we have concluded that small changes might create significant improvements in the behavior of the system. As such, we decided to further tune the prototype system by:

Improving classification by re-training the classifier using the queries gathered during the user study; also, we used the Weka suite to try out different classification models and configurations with the new training data.

Improving QA by using a better answering service (Evi.com) which searches multiple Q&A API's and keeping Yahoo Answers as a fall-back.

This increased our accuracy (in stratified 10-fold cross validation) to 89%. However, after implementing these improvements, the initially provided query structure was changed from only non-specific queries to both specific and non-specific queries, which resulted in a drastic accuracy drop. Furthermore, the new structure was also affecting the way our information extraction method worked (described in Section 3.2.1). One of the problems making our system incompatible with non-specific queries was located in the information extraction method that we used for Person queries. This information extraction method obtains the first wikipedia search result after searching for the full query text, however for non-specific queries such an article does not exist and thus no results were found. To overcome this problem the wikipedia search was changes from searching for the article with the title best matching the query text into searching for the article title having the largest sequence of words in common with the query text. As a result, a query as "harry potter actor", for which there is no fully matching wikipedia article will still be matched for the query part "harry potter". The residue part of the query which was not used for the match, "actor" in case of our example, is identified as specific part of the query and is used to extract relevant sentences within the "harry potter" article.

Another problem caused by the introduction of non-specific queries occurred in the classification phase of the system and again has to do with matching wikipedia titles with the query text. For specific Person queries the hasWikipedia feature is often false. As the hasWikipedia feature used to have a substantial role in discriminating Person queries from other query types a large decrease, specific Person queries are often misclassified. As a solution, a new hasWikipedia feature has been implemented which searches for the article title having the largest sequence of words in common with the query text and compares the Jaro-Winkler distance [20] of the partial query sequence and wikipedia article title with and minimum value, chosen to be 0.7. Intuitively this new hasWikipedia feature seems to return true for most Person queries, including specific queries. The downside is however that more non-Person queries seem to score true on the new hasWikipedia feature, therefore the added capability of identifying specific Person queries as Person queries came at the cost of our discriminatory power to distinguish between Person and non-Person queries. To restore some of the discriminatory power to distinguish between Person and non-Person queries the original hasWikipedia (as proposed in [9] has been re-introduced as additional feature under the name hasFullWikipedia. The official NTCIR test runs will be used to evaluate the added value of using both the hasFullWikipedia and the altered hasWikipedia features compared to only using the hasWikipedia feature, submitting a run using both features as first run (ut-E-D-OPEN-1) and a run using only the hasWikipedia feature as second run (ut-E-D-OPEN-2).

5. EVALUATION OF THE SYSTEM AT NTCIR

An official evaluation of our system was achieved by submitting two runs (runs described in section 4.4) to the NTCIR evaluation panel. Both submitted runs were Desktop runs (output limit of 1000 characters).

5.1 Evaluation methodology

A description of the task and evaluation methodology details are described in detail in [8]. A semi-automated method was put in place to extract vital strings for each query. These vital strings will then be ranked and each run will be manually examined to determine its relation to these strings. A manual evaluation was conducted by an evaluation panel by comparing the submitted runs to each other and to some baseline runs that make use of Microsoft Bing and Oracle. The metrics used were S, T, and S# measures, with S#-measure being a combination of the former. These metrics are designed to take the length of the output into account as well as the number of relevant pieces of information (vital strings or nuggets) and their position and order.

5.2 Evaluation Results

Figure 1 shows an overview of the official results across all desktop runs, ordered from high to low. It shows that our first run outperformed the second run and all other participant runs for the Desktop category and also two of the baseline runs. We can conclude on the results that an API-based approach is a promising way of designing a One Click information extraction system. Furthermore we can conclude that the addition of the original hasWikipedia feature (as proposed by [9] is beneficial to the performance of the system and therefore helps in the classification task.

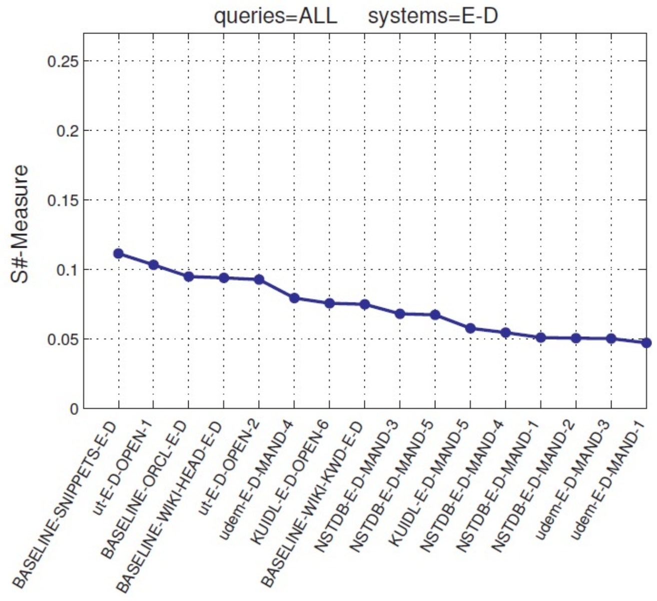
On a task-wide comparison (of both Desktop and Mobile runs) our two runs also turn up in the first half. Furthermore, in [8] we also see that our system performed best for *Person*, with our *Person* query responses being second only to a Mobile run from KUIDL. This shows that the keyword based full text search approach using wikipedia is a very good method for query classification for *Person* queries. Overall we can conclude an API-based One Click information extraction system to be feasible.

6. CONCLUSION AND DISCUSSION

Considering the results from the significance tests, we can conclude that the results we have obtained from the evaluation are not only successful, but also show a statistically significant difference between the two systems for most query types, except *Facility*, *Geo* and *Politician.* As a conclusion: the proposed prototype is a successful approach to designing a One Click Access Information Retrieval system. From our evaluation we also can also conclude that there is still room for improvement. A few such points of improvement are discussed in more detail below in section 6.1.

We believe that the new field of One Click access offers exciting possibilities for the future of web search engines,

Figure 1: S# score for all systems averaged across all queries (from [8])



with benefits in speed and simplicity. In a world where information is transforming from available to overwhelming, the key lies in fast access to relevant and precise information. We do not see One Click will ever replace *traditional* web search, but we strongly believe it to have the potential of becoming an integral part of such systems.

Furthermore, we conclude that exclusively using publicly available API's to answer most queries is a feasible approach in designing information retrieval systems. One Click results can dramatically enrich search results, while also significantly decrease the average time of retrieving the desired information.

6.1 Future Work

We see the following possible improvements to the system described in this paper worthwhile to experiment with in the future.

6.1.1 *Improving the Classifier*

One obvious place for improvement of the classifier is for *Facility* query types. This can be partly achieved by enriching the keyword database for the Person/facility classifier described in section 3.1. This database was for the experiment created based on our own intuitions of often occurring words in *Facility* query wikipedia articles, but higher performance is expected when actual textual analysis will be applied to a randomly selected set of *Facility* query wikipedia articles. Higher performances as a result of a more structured keyword database construction method can also be obtained for the *Person* query types, as the set of chosen keywords for the *Person* query types were also based on own views on often occurring words for respective articles. In our experience we have found that a proper collection of keywords can be efficiently used as a sole feature-set in classifying queries belonging to a certain information need.

A further improvement could also be achieved by introducing new features to the classifier that differentiate the *Facility* query type of queries from the others.

6.1.2 Improving the information extraction methods

The Web nowadays offers a plethora of information, accessible via API's or page scraping, which opens up almost unbounded possibilities for information extraction. We believe the methods described in this paper, while suitable for the particular query types implemented, are not optimal and should only be taken as a guideline or starting point for similar future projects. Only minuscule part of information API's have been taking into account in this research, with other API's having the potential of increasing the performance of the system. When combining these API's and scraping possibilities on the endless Web ([3]) with existing text-summarizing methods and techniques, which has not yet been done in system, the system can without doubt be improved even further ([11]).

6.1.3 Other Improvements

While the query types used as a base for the classifier cover most of the common information need a One Click access system might have to address, we feel that the list is definitely not exhaustive. For such a system to become a matured technology, a much more complete list of categories should be devised, relative to the scope of the search engine. During the evaluation we noticed most users did experience the categorization to be unintuitive and not self explanatory, which caused confusion in the sort of queries that belongs to each category. We feel that the eight query types used in this system are artificial and do not constitute a proper base for a public system. However, the information extraction methods used can be used for other/new categories with minor adaptations.

7. ACKNOWLEDGEMENTS

This publication was supported by the Netherlands national program COMMIT and the Netherlands Organisation for Scientic Research (NWO, project 639.022.809).

8. REFERENCES

[1] L. B. Chilton and J. Teevan. Addressing people's information needs directly in a web search result page. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 27–36, New York, NY, USA, 2011. ACM.

[2] H. Dang and J. Lin. Different structures for evaluating answers to complex questions: Pyramids won't topple, and neither will human assessors. In *Annual Meeting of the Association For Computational Linguistics*, volume 45, page 768, 2007.

[3] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298. ACM, 2002.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations*, 11(1):10–18, 2009.

[5] E. Hatcher, O. Gospodnetic, and M. McCandless. Lucene in action, 2004.

[6] A. Hoekstra, D. Hiemstra, P. Van Der Vet, and T. Huibers. Question answering for dutch: Simple does it. 2006.

[7] N. Kando, D. Ishikawa, and M. Sugimoto, editors. *Proceedings of the 9th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access.* National Institute of Informatics, 2011.

[8] M. P. Kato, M. Ekstrand-Abueg, V. Pavlu, T. Sakai, T. Yamamoto, and M. Iwata. Overview of the ntcir-10 1click-2 task. In *Proceedings of the 10th NTCIR Conference*, 2013.

[9] M. P. Kato, M. Zhao, K. Tsukuda, Y. Shoji, T. Yamamoto, H. Ohshima, and K. Tanakai. Information Extraction based Approach for the NTCIR-9 1CLICK Task. In Kando et al. [7], pages 202–207.

[10] J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and pc internet search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 43–50, New York, NY, USA, 2009. ACM.

[11] H. Morita, T. Makino, T. Sakai, H. Takamura, and M. Okumura. TTOKU Summarization Based Systems at NTCIR-9 1CLICK task. In Kando et al. [7], pages 202–207.

[12] N. Orii, Y.-I. Song, and T. Sakai. Microsoft research asia at the ntcir-9 1click task. In Kando et al. [7], pages 216–222.

[13] V. Pavlu, S. Rajput, P. B. Golbus, and J. A. Aslam. Ir system evaluation using nugget-based test collections. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 393–402, New York, NY, USA, 2012. ACM.

[14] T. Sakai, M. P. Kato, and Y.-I. Song. Click the search button and be happy: evaluating direct and immediate information access. In C. Macdonald, I. Ounis, and I. Ruthven, editors, *CIKM*, pages 621–630. ACM, 2011.

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

[16] T. Sakai, M. P. Kato, Y.-I. Song, R. Song, M. Zhang, Y. Liu, and N. Craswell. 1CLICK@NTCIR-9 Task description. 1CLICK@NTCIR-9 Home Page, Sept. 2011.

[17] P. Thomas and D. Hawking. Evaluation by comparing result sets in context. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 94–101, New York, NY, USA, 2006. ACM.

[18] K. Tjin-Kam-Jet, D. Trieschnigg, and D. Hiemstra. Free-text search versus complex web forms. In *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, pages 670–674. Springer Berlin / Heidelberg, 2011.

[19] O. Turetken and R. Sharda. Clustering-based visual interfaces for presentation of web search results: An empirical investigation. *Information Systems Frontiers*, 7(3):273–297, 2005.

[20] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, pages 354–359. ASA, 1990.