Click the Search Button and Be Happy: Evaluating Direct and Immediate Information Access

Tetsuya Sakai
Microsoft Research Asia, China
tetsuyasakai@acm.org

Makoto P. Kato
Kyoto University, Japan
kato@dl.kuis.kyoto-u.ac.jp

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

ABSTRACT

We define *Direct* Information Access as a type of information access where there is no user operation such as clicking or scrolling between the user's click on the search button and the user's information acquisition; we define *Immediate* Information Access as a type of information access where the user can locate the relevant information within the system output very quickly. Hence, a *Direct and Immediate Information Access* (DIIA) system is expected to satisfy the user's information need very quickly with its very first response. We propose a nugget-based evaluation framework for DIIA, which takes nugget positions into account in order to evaluate the ability of a system to present important nuggets first and to minimise the amount of text the user has to read. To demonstrate the integrity, usefulness and limitations of our framework, we built a Japanese DIIA test collection with 60 queries and over 2,800 nuggets as well as an offset-based nugget match evaluation interface, and conducted experiments with manual and automatic runs. The results suggest our proposal is a useful complement to traditional ranked retrieval evaluation based on document relevance.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Experimentation

Keywords

evaluation, information access, nugget, test collection

1. INTRODUCTION

Information Retrieval (IR) evaluation methods need to evolve together with users' needs and with IR systems. Classical IR evaluated *set retrieval* by means of *recall* and *precision*; but when the exponential growth of searchable information made *ranked retrieval* a necessity, IR evaluation turned to metrics such as *Average Precision* and *normalised Discounted Cumulative Gain* (nDCG) [10]. Furthermore, as Web search studies have revealed that user's queries are often *ambiguous* and/or *underspecified*, new evaluation metrics suitable for selectively *diversifying* search results have begun to receive attention (e.g., [6, 20]). However, all of these evaluation efforts still focus on ranked retrieval: a list of documents is presented to the user, and the user is expected to choose from the list and then read the entire Web pages that have been chosen.

In modern Web search engines used in desktop and mobile environments, providing a simple ranked list of documents to the user is becoming increasingly insufficient. A Search Engine Result Page (SERP) typically consists of several components besides a list of URLs with snippets, depending on query type. Some of these components aim to satisfy the user needs directly: for example, weather, flight status, stock prices, currency conversion, machine translation and other types of "answers" can be embedded in the SERP [5].

We define *Direct* Information Access (DIA) as a type of information access where there is no user operation such as clicking or scrolling between the user's click on the search button and the user's information acquisition. Chilton and Teeven's "answers" [5] are examples that enable DIA; Li, Huffman and Tokuda [12] have referred to successful DIA cases as "good abandonements." Moreover, we define *Immediate* Information Access (IIA) as a type of information access where the user can locate the relevant information within the system output very quickly. For example, if a system can show a query-biased summary (a search engine snippet would be a trivial example) of a retrieved Web page to the user, this system may be more effective for IIA compared to one that shows the entire contents of that page. Hence, a *Direct and Immediate Information Access* (DIIA) system is expected to satisfy the user's information need very quickly with its very first response.

Suppose that a Japanese user needs to visit a particular Japanese hospital, and he enters the name of the hospital in the query box of a DIIA system. Figure 1 shows a possible DIIA system output (This is from our manual run which we shall discuss in Section 5.1) that the present study considers. As the English translation shows, our "imaginary" DIIA system outputs the contact details and opening hours of the hospital. The Japanese text is about 140 characters and may fit a mobile phone screen size. If the DIIA system is used on a PC, then the DIIA system may be able to return more information, such as the nearest train station, car park availability and so on.

This paper proposes and validates a new methodology for evaluating textual DIIA systems that respond to a given query as shown in Figure 1. Our framework is closely related to multi-document summarisation and Question Answering (QA) evaluations, but differs in the following aspects. First, while the summarisation task treats the set of documents to be summarised as a given input to the system, our DIIA framework considers both the phases of *knowledge source selection* (i.e., finding relevant documents by means of a Web search API etc.) and *knowledge compilation* (i.e., extracting relevant pieces of information or *nuggets* [7, 15] from documents

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...$10.00.

[Japanese output]

電話 046-223-3636。fax 046-223-3630。住所 243-8551 厚木市温水 118-1。email soumu@shonan-atsugi.jp。面会時間:一般病棟 月～金 15-20 時、土日祝日 13-20 時／集中治療室(ICU) 11-11 時半 15-15 時半 19-19 時半。

[English translation]

Phone:046-223-3636. Fax:046-223-3630.Address:118-1 Nurumizu, Atsugi, 243-8551. Email:soumu@shonan-atsugi.jp. Visiting hours:general ward Mon-Fri 15-20; Sat&Holidays 13-20 / Intensive Care Unit (ICU) 11-11:30,15:30,19-19:30.

Figure 1: An example Japanese DIIA output and its translation (query: "Shonan Atsugi Hostpital").

and organising them for effective presentation for a given output screen size). Second, while traditional summarisation and QA evaluations are based on similarities with gold standards or nugget recall and precision (See Section 2.2), the DIIA task we define require the following features of a system:

- Present important nuggets first;
- Minimise the amount of text that the user has to read.

This is a Direct IA task because the system has to return a text that contain pieces of relevant information in response to a given query (instead of returning a list of URLs); it is an Immediate IA task because the above two requirements enable the user to obtain the desired information quickly.

To evaluate the aforementioned requirements of DIIA systems, we introduce a simple evaluation metric called *S-measure* which is an extension of weighted nugget recall but takes the positions of nuggets within the system output into account. To demonstrate the integrity, usefulness and limitations of our DIIA framework, we built a Japanese DIIA test collection with 60 queries and over 2,800 nuggets as well as an offset-based nugget match evaluation interface, and conducted some experiments.

Clearly, DIIA is not just about textual responses: effective DIIA systems should output visual "answers" such as maps, images and so on. These are beyond the scope of our study. Moreover, even when we restrict ourselves to textual responses, aesthetic features such as highlighting and font sizes are important in practice. Our current framework can only handle plain text. Nevertheless, we claim that our evaluation framework is a useful departure from and complement to traditional ranked list evaluation.

Section 2 describes previous work related to the present study. Section 3 proposes our new evaluation framework for the DIIA task. Section 4 describes the DIIA evaluation environment we constructed, and Section 5 reports on a set of experiments using this environment. Finally, Section 6 concludes this paper.

2. RELATED WORK

2.1 Evaluating Search

In modern IR (particularly Web search) evaluation, nDCG [10] has become a popular evaluation metric. While essentially the same idea for evaluating ranked retrieval based on graded relevance existed back in the 1960s (i.e., Pollack's *sliding ratio* [11]), the simple ideas behind nDCG are appealing: a system is rewarded with a gain value for retrieving a relevant document, and the gain values vary according to the relevance levels; each gain is *discounted* according to the rank of the retrieved relevant document; and an *ideal ranked list* is used to normalise the overall system score.

Our proposed metric for evaluating DIIA was inspired by nDCG. However, while nDCG evaluates a ranked list of items, we evaluate a textual output, which is much more problematic. It should also be noted that, while the original nDCG incorporated the notion of *user's patience* as a parameter (i.e., the logarithm base), the *de facto* standard version of nDCG [4] lacks this parameter. In contrast, our proposed metric retains a similar parameter that represents the maximum amount of text the user is expected to read, or equivalently, the maximum amount of time the user is expected to spend. In this respect, Dunlop's work [8] that explored IR evaluation by *expected search duration* is also related to our work.

Recently, methods for evaluating *search result diversification* have been proposed (e.g., [6, 20]). Among them, *α-nDCG* and *Novelty- and Rank-Biased Precision* (NRBP) [6] are somewhat related to our study, as they view both information needs and documents as sets of nuggets. However, these metrics are for evaluating ranked lists, not text.

INEX (INitiative for the Evaluation of XML retrieval) 2011 has launched the Snippet Retrieval Track[1], which is also somewhat relevant to this study as the task involves generation of a textual snippet for each retrieved document. However, the INEX task is still based on document relevance, and the snippets are regarded as a means to judge the relevance of the original documents within the traditional ranked list evaluation framework [21].

Bailey *et al.* [3] proposed a frawework for evaluating the *whole page relevance* of a SERP, which typically consists of multiple components, not all of which are textual. While their approach and our DIIA framework both attempt to go beyond ranked list evaluation, the former does not aim at producing a reusable test collection. In contrast, while our DIIA framework focusses on a single textual response, we aim to provide a nugget-based test collection that is reusable to some extent.

2.2 Evaluating Textual Output

As our DIIA task is similar to summarisation and QA evaluations in that the systems return textual responses, prior art in these areas needs to be discussed.

ROUGE is a family of evaluation metrics for evaluating summaries *automatically* [13]. While several versions of ROUGE exist, the key idea is to compare a system output with a set of gold-standard summaries in terms of *recall* (or alternatively *F-measure*), where recall is defined based on automatically-extracted textual fragments such as N-grams and longest common subsequences. *POURPRE*, an automatic evaluation metric for complex QA, is essentially F-measure computed based on unigram matches between the system output and gold-standard nuggets [14]. Also, new automatic summarisation metrics are being explored at the recent AESOP (Automatically Evaluating Summaries of Peers) task at TAC (Text Analysis Conference)[2].

The above automatic methods assume that automatic string matching between the system output and the gold standard works well. While these methods reduce the evaluation cost dramatically and work for *extractive* approaches (i.e., summaries or answers are parts of source documents), they are probably insufficient for evaluating intelligent systems that can handle consolidation of information across multiple sources, paraphrasing, textual entailment and inference. Hovy, Lin and Zhou [9] discuss some challenges in automatic matching in the context of evaluating summaries using *Basic Elements*. Indeed, we believe that, in order to quickly satisfy the user's information need with a small piece of text, systems will ultimately have to employ *abstractive* techniques. While a few approaches to incorporating paraphrase matching into automatic evaluation exist [16, 25], the premise in our study is that a system output and

[1] https://inex.mmci.uni-saarland.de/tracks/snippet/

[2] http://www.nist.gov/tac/2011/Summarization/

a list of gold-standard nuggets need to be compared *manually*, at least until we fully understand exactly what kinds of abstractive techniques are useful and reliable for the DIIA task.

The *pyramid method* [17], a recently-proposed method for manually evaluating summaries, is similar to our new proposal. The pyramid method requires multiple gold-standard summaries, from which *semantic content units* (SCUs) are extracted. Each SCU is weighted according to the number of gold-standard summaries it matches with. An assessor manually identifies SCUs within a system's summary, and the pyramid method essentially computes SCU-based weighted precision or weighted recall.

A similar method has been used for evaluating complex QA at TREC (and "squishy-list" QA at TAC) [7]. As was mentioned earlier, the TREC QA systems were required to output a set of documentID-answer pairs rather than a single text. In this evaluation methodology, gold-standard nuggets are created by a single assessor, but nugget importance is determined based on multiple assessors, who assign either *vital* or *okay* to each nugget. Nugget-based precision, weighted recall and F-measure were used for evaluation. However, the computation of precision is problematic here, because the number of "incorrect nuggets" present in a system output is difficult to define [7, 14]. Hence, an *allowance* of 100 characters per nugget match was introduced: if there are k nugget matches and if the system output length l is smaller than $100 * k$, then precision was defined as 1; otherwise precision was defined as $100 * k/l$. This evaluation method was also used for QA tasks with Asian languages at NTCIR[3], with different allowance values for different question types [15].

Unlike the pyramid methods for summarisation and QA, we aim to evaluate the system's ability to present important nuggets first and to minimise the amount of text the user has to read. Hence we take into account the *position* of each nugget match within a system output. Moreover, while the pyramid QA evaluation relies on an arbitrarily-set, fixed-length allowance for computing precision, our methodology allows nugget constructors to define *vital strings* which are used for approximating a length lowerbound for each gold-standard nugget (See Section 3). Note also that the pyramid methods discussed above are *content selection* evaluation metrics: the assumption is that *linguistic quality* evaluation is done separately [17]. Our evaluation framework is similar: S-measure is a *content ranking* evaluation metric, and does not directly consider readability aspects such as coherence and cohesiveness. The challenging problem of evaluating the readability of summaries is being tackled at the aforementioned TAC AESOP task.

2.3 DARPA GALE Distillation Evaluation

Another relevant line of research that lies more or less in between the aforementioned document retrieval paradigm and the textual output paradigm is the nugget-based evaluation effort for the DARPA GALE *distillation* program which was completed in 2010. Babko-Malaya [2] describes a systematic way to define nuggets in a bottom-up manner from a pool of system output texts. This is done prior to determining whether each nugget is relevant or not: thus, even nonrelevant parts of text need to be "nuggetised." In contrast, our nuggets are created in a top-down manner by means of manual web search. However, Babko-Malaya's approach to defining the *nugget granularity* and handling *world knowledge* may also be useful for our task: we leave this to future work.

White, Hunter and Goldsten [22] defined several nugget-based metrics for the distillation task, but they are set retrieval metrics. Allan, Carterette and Lewis proposed a character-based version of *bpref* (binary preference) to evaluate a ranked list of passages [1][4]. Yang *et al.* [24] and Yang and Lad [23] have also discussed nugget-based evaluation metrics that are similar in spirit to the aforementioned α-nDCG, but their distillation tasks are quite different from, and more complex than, our DIIA task: they consider multiple queries issued over a period of time, and multiple ranked lists of retrieved passages. On the other hand, their proposal to explicitly incorporate the user cost (of reading nonrelevant text at the end of system output) deserves attention and may be considered in our DIIA framework in the future.

3. PROPOSED FRAMEWORK

3.1 Task Definition

We first define the DIIA task as follows. Given a query input by the user, return a textual response whose length is no more than X characters (or words, or bytes), excluding white spaces and punctuation marks. We assume that some pieces of information are more important than others, and that important ones should be presented first to the user. Moreover, we expect a DIIA system to try to minimise the amount of text that the user has to read in order to satisfy his information need. Thus, a piece of relevant information near the beginning of the output text (which we call the *X-string*) will be rewarded more than the same piece of information near the end of the text. As the output length is limited, redundancy (i.e., repetition of the same information within the text) is penalised.

In this paper, we consider a Japanese DIIA task as an example, and consider $X = 140$ or 500 in Japanese characters. The former is designed to represent a mobile phone screen size; the latter roughly corresponds to top five snippets in a typical SERP, which are usually visible without scrolling in a desktop environment;

3.2 User Model and Nugget Discounting

We propose a simple evaluation framework for DIIA. First, let us assume that the human reading speed is constant. For example, it is known that the average reading speed of a Japanese person is 400-600 characters per minute. For convenience, we assume that the speed is 500 characters per minute. Our arguments can be extended to other languages, for example, by replacing the constant with 250 words per minute for English, and so on.

As we have discussed earlier, DIIA aims to present important nuggets first and to minimise the amount of text the user has to read in order to obtain the desired information. To evaluate this, we propose to conduct nugget-based evaluation following the practices of QA and summarisation communities, *and* to evaluate each system output by taking into account the *positions* of nugget matches found in it. This is in contrast to traditional QA and summarisation evaluation, where it is only the *presence* of nugget matches that matters. More specifically, we use a *discount factor* with each nugget match based on its *offset* (i.e., distance from the beginning of text), so that a nugget match near the beginning of text receives more credit than one near the end. This is analogous to how normalised discounted cumulative gain (nDCG) [10] discounts relevant documents based on document ranks. However, while the log-based discounting function of nDCG is understood as a model of the user scanning a ranked list of retrieved items from top to bottom while his patience gradually runs out, we are to model a user reading a piece of text from beginning to end. Assuming that the reading speed is constant, applying a *linear* discount to nugget matches based on the offset values is probably one sensible approach.

[3] http://research.nii.ac.jp/ntcir/

[4] It is known that there are more elegant and reliable ranked retrieval metrics than bpref, the simplest being the average precision defined over a *condensed list* [18].

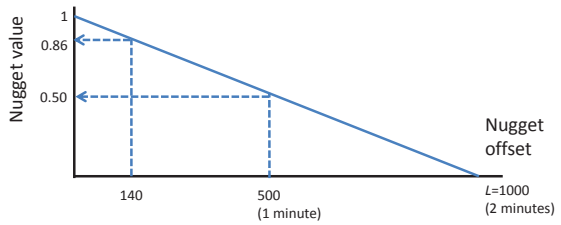

Figure 2: Discounting nugget values.

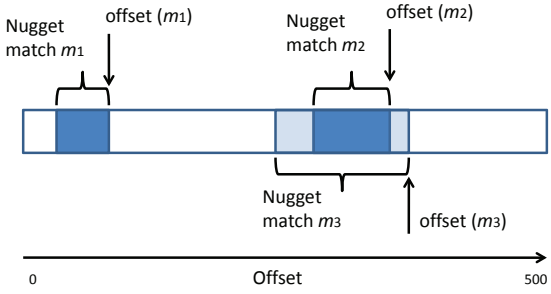

Figure 3: Nugget matches and offsets.

Let us therefore assume that the value of a nugget match decays linearly with respect to the offset. Moreover, let us assume that the value of a nugget wears out completely after (say) two minutes. This corresponds to a situation where the user cannot afford to spend more than two minutes (after pressing the search button) to gather information. For our average Japanese user, this translates to $L = 2 * 500 = 1,000$ characters.

Figure 2 shows how nugget values can be discounted in the above setting. For a *DESKTOP run* (where $X = 500$), the value of a nugget found at the very beginning of the text is 1, but that of one found at the very end is only 0.5. For a *MOBILE run* (where $X = 140$), the value of a nugget found at the very end is $1 - 140/1000$=0.86. Ideally, the parameter L should be determined based on a practical requirement.

How to define the offset of a nugget match deserves a discussion. We assume that an assessor uses an interface dedicated to evaluating an X-string by comparing it with a list of nuggets. Moreover, unlike traditional nugget-based evaluation, we assume that when the assessor identifies a nugget match, he records which part of the X-string corresponds to a nugget. As we shall describe in Section 4.3, the *nugget match area* can easily be recorded by means of a mouse drag. We thus assume that for every nugget match, we can obtain the start and end positions of the nugget match area.

Figure 3 depicts how we define the offset of a nugget match in this study. As the figure shows, we propose to use the end position of a nugget match area as the offset value, because the user probably has to read *through* the nugget match area in order to obtain the information conveyed in that nugget. Note also that, in Figure 3, there are three nugget matches, and one nugget match area subsumes another. In general, nugget match areas can overlap with one another, and also multiple nugget matches may share the same offset value, which is analogous to tied documents in IR.

Following previous work in QA and summarization, we define a nugget loosely as an atomic piece of information whose presence/absence in a text can be judged by an assessor. Moreover, we assume that we have a weight (i.e., importance) assigned to each nugget, just like *graded* relevance assessments are available in modern IR test collections. In practice, we let multiple assessors assign a grade to each nugget, and then take the sum of the grades to define *nugget weights*. An alternative approach would be to define "vital" and "okay" nuggets [7].

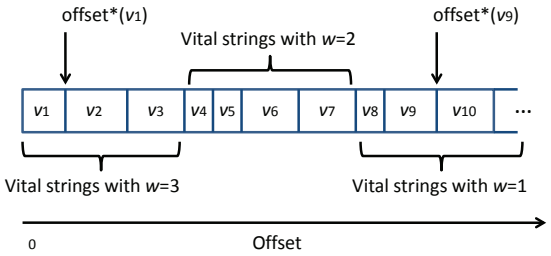

Figure 4: A pseudo minimal output composed of vital strings.

3.3 S-measure

We now formally define S-measure. Let N be a set of gold-standard nuggets constructed for a particular query, and let $M(\subseteq N)$ denote the set of *matched* nuggets, i.e., those manually identified in the X-string. For each $m \in M$, let $w(m)$ denote its nugget weight, and let $\mathit{offset}(m)$ denote its offset value. Then, by applying the aforementioned linear discounting scheme, we could evaluate an output for a DIIA task by computing:

$$\sum_{m \in M} w(m) \max(0,\ 1 - \mathit{offset}(m)/L)\ . \qquad (1)$$

As shown in Figure 2, L is the amount of text read that corresponds to the point where all information becomes useless to the user (set to 1000 in this paper). The max operator just means "ignore nuggets whose offsets are greater than L."

For evaluation with a set of queries, we should *normalise* the above metric so that it lies within the 0-1 range. In the case of IR evaluation based on nDCG, this is easily done by defining an ideal ranked list that exhaustively lists up all relevant documents in descending order of relevance levels. In our case, however, defining an ideal output is problematic, because our evaluation is based on offset-based discounting rather than rank-based one. Our proposed solution is to prepare a *vital string* $v(n)$ for each nugget n in order to approximate the minimal text length required to convey the meaning of a nugget within a textual output. For example, suppose that for query "Waseda university," we have two nuggets n_1 and n_2 representing the phone and fax numbers of this university, amongst other nuggets. That is, n_1 represents the fact: "Waseda university's phone number is YY-YYYY-YYYY" while n_2 represents the fact: "Waseda university's fax number is ZZ-ZZZZ-ZZZZ" (where X and Y represent any digit). Then we may define $v(n_1)$ and $v(n_2)$ as "YY-YYYY-YYYY" and "ZZ-ZZZZ-ZZZZ." Note that if an X-string contains "ZZ-ZZZZ-ZZZZ" but does not mention that this is a fax number, the user may not find it useful. Thus, a vital string represents a small piece of text that probably *needs* to be included in the output: it does not necessarily contain sufficient information. The idea is to provide a (possibly unreachable) length lowerbound for each nugget.

For normalising our evaluation metric, we define a *Pseudo Minimal Output* (PMO) by sorting all vital strings that correspond to all nuggets $n \in N$ for a query. The first sort key is the nugget weight $w(n)$ (larger the better), and the second sort key is the vital string length $|v(n)|$ (smaller the better). Figure 4 shows an example of PMO. Here, we assume that we have three different nugget weights $w = 3, 2, 1$, and the vital strings are sorted accordingly. Morever, within each set of nuggets with the same weight, the vital strings are sorted by length. For example, v_1 is the first vital string in this PMO because its nugget has the highest weight (namely 3) *and* v_1 is shorter than the other two vital strings with the same weight. Note that a PMO is a mere concatenation of vital strings and may completely lack cohesiveness and coherence.

For any vital string v, let $\mathit{offset}^*(v)$ denote its offset position within a PMO. As Figure 4 shows, we use the end position of each vital string. Then, we define *S-measure* as[5]:

$$S\text{-}measure = \frac{\sum_{m \in M} w(m) \max(0,\ 1 - \mathit{offset}(m)/L)}{\sum_{n \in N} w(n) \max(0,\ 1 - \mathit{offset}^*(v(n))/L)}$$

$$= \frac{\sum_{m \in M} w(m) \max(0,\ L - \mathit{offset}(m))}{\sum_{n \in N} w(n) \max(0,\ L - \mathit{offset}^*(v(n)))} . \quad (2)$$

It is clear that as L approaches infinity and the effect of offset-based discounting is reduced, S-measure approaches the traditional *weighted nugget recall*. The novel features of S-measure when compared to existing QA and summarisation metrics for the purpose of DIIA evaluation are: (1) It takes into account the position of nugget matches; and (2) It approximates the minimal length required *for each nugget*, while traditional nugget-based precision assumes that this length is a constant (i.e., the allowance). On the other hand, the above definition implies that S-measure is not theoretically bounded above by 1: there are two reasons for this.

The first reason is that the PMO as defined above does not necessarily maximise S-measure's numerator. Suppose that we have two nuggets n_1 and n_2, such that $w(n_1) = 2, |v(n_1)| = l_1$ and $w(n_2) = 1, |v(n_2)| = l_2$. Then it is easy to show that an X-string "$v(n_2)v(n_1)$" (i.e., one that presents the vital string of the less important nugget first) may receive an S-measure greater than one if $l_1 > 2l_2$. For example, if $l_1 = 3$ and $l_2 = 1$ (i.e. the important nugget requires a much larger space than the other one), the numerator for the above X-string would be $1 * (1000 - 1) + 2 * (1000 - (1 + 3)) = 2991$. (We repeat, however, that a simple concatenation of vital strings such as "$v(n_2)v(n_1)$" may be unreadable as vital strings do not necessarily contain *sufficient* information.) On the other hand, the denominator, which represents the PMO "$v(n_1)v(n_2)$", would be $2 * (1000 - 3) + 1 * (1000 - (3 + 1)) = 2990 (< 2991)$. Despite this shortcoming, we use the nugget weight as the first sort key and the vital string length as the second sort key for defining the PMO because (a) we consider presenting important nuggets first more important than presenting short nuggets first; and (b) the above simple definition of PMO makes S-measure computationally cheap.

The second reason that S-measure can exceed one arises from the basic assumption behind *all* nugget-based evaluation methods, namely, that nuggets are independent of one another. For example, consider the aforementiond example with a phone number and a fax number of a university. Suppose that these two numbers happen to be identical, and that we have defined the vital strings as "YY-YYYY-YYYY" (10 characters, excluding "-"s) for both nuggets. Then, a PMO will require at least 20 characters in total for these two nuggets. However, an intelligent DIIA system might express these pieces of information as "phone&fax:YY-YYYY-YYYY" (19 characters, excluding "-"s and ":"), which is in fact more concise than the PMO.

We later demonstrate that this lack of a theoretical upperbound for S-measure is not a serious problem in practice. One method for preventing S-measure from exceeding one is to define very short vital strings. We will provide real examples in Section 4.2. Nevertheless, we can also formerly define a version of S-measure that is guaranteed to range between 0 and 1 as follows:

$$S\flat\text{-}measure = \min(1, S\text{-}measure) . \quad (3)$$

We call this "S flat measure" because it "flattens" S-measure whenever it exceeds one.

[5]S stands for "the user Scanning a String at a constant Speed."

4. CONSTRUCTING AN EVALUATION ENVIRONMENT

To demonstrate the feasibility of our new evaluation framework, we constructed a Japanese DIIA test collection containing 60 queries and over 2,800 nuggets[6]. As DIIA systems are expected to perform both *knowledge source selection* and *knowledge compilation* (See Section 1), we allow DIIA systems to utilise any existing Web pages as their knowledge sources instead of requiring them to draw information from a closed document collection. We call this the *open knowledge source* evaluation.

Each of our nuggets is associated with a URL: this is the supporting document based on which the nugget was constructed. Thus, instead of making DIIA systems perform both knowledge source selection and knowledge compilation, we may optionally let the DIIA systems utilise the supporting URLs directly, thus isolating and evaluating the knowledge compilation component. We call this the *oracle knowledge source* evaluation.

In order to make our test collection as reusable as possible and to ensure successful matches between our gold-standard nuggets and system responses, we contructed queries that seek *established facts* rather than time-sensitive, controversial or subjective information. However, as we shall discuss in Section 4.2, even "established" facts may change over time, and some *truth maintenance* effort in the future may help prolong the life of our DIIA test collection.

4.1 Creating Queries

As we wanted our DIIA test collection to handle both DESKTOP and MOBILE situations (See Section 3.1), we referred to the work by Li, Huffman and Tokuda [12] that analysed mobile and desktop query logs from the United States, Japan and China, and identified frequent query types for potential "good abandonment" queries. They found that the query distributions over query types are quite different across the three countries: for Japan (which they considered to be a mature market in mobile search), the *LOCAL*, *QA*[7], *CELEBRITY* and *DEFINITION* query types were very popular in both mobile and desktop environments. We therefore decided to collect queries for these four query types. The *original* definitions of these query types by Li, Huffman and Tokuda were as follows:

CELEBRITY User seeks news or images of a celebrity.

LOCAL User seeks a local listing (address and/or phone number).

DEFINITION User seeks the definition of a term.

QA User seeks a short answer to a question.

While we basically followed the original definitions for the DEFINITION and QA query types, we interpreted CELEBRITY and LOCAL query types differently, as shown below:

CELEBRITY User wants to gather various facts about a celebrity: date/place of birth, real name, blood type, height, hobbies, profession, personal history, awards, publications, discography, films, TV series, favourite baseball team, favourite food etc.

LOCAL User wants to contact or visit a facility (school, shop, office, amusement park, hotel, train station etc.). Hence (s)he wants facts such as postal and email addresses, phone and fax numbers, opening hours, how to access the facility by train/bus/car, nearest stations, time required for the travel, whether the facility has a car park and its opening hours etc.

[6]We will make the test collection available through NTCIR.

[7]Li, Huffman and Tokuda [12] called it the "answers" category.

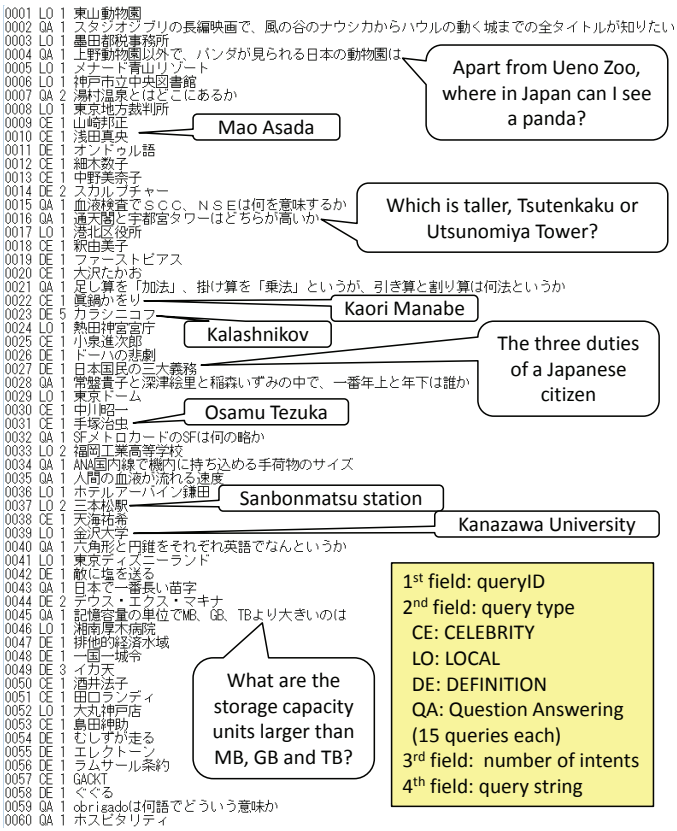

Figure 5: 60 DIIA queries.

Table 1: Number of nuggets per query type.

query type	#queries	mean	range
CELEBRITY	15	126.5	[38, 368]
LOCAL	15	30.9	[10, 125]
DEFINITION	15	26.3	[2, 174]
QA	15	5.6	[2, 26]
all	60	47.3	[2, 368]

Note that we had to replace the original definition of CELEBRITY completely to make the test collection as reusable as possible. These detailed specifications were formulated through repeated revisions during the nugget creation process.

After deciding on the four query types, we collected 15 queries for each type, thereby obtaining 60 queries in total. As we wanted to handle information needs that are as realistic as possible, we used two sets of real query data. For obtaining CELEBRITY and LOCAL queries, we used a proprietary Japanese mobile query log which covered two weeks in October 2009. It contained approximately 39,000 unique queries ranked by query frequency. From this data set, we manually selected celebrity names and local facility names while conducting trial Web searches and ensuring that good information sources exist on the Web. For obtaining DEFINITION and QA queries, we examined the Yahoo! Chiebukuro (Japanese Yahoo! Answers) data [19]. Natural language questions were manually selected from categories such as *education*, *local* and *health* categories, which we thought were more appropriate for factual information seeking than other categories such as *love*. For the DEFINITION query type, we used the look-up term as the query, while for QA, we formulated a short natural language sentence from each original question in the Yahoo! data.

Figure 5 shows the entire query set thus constructed, with some rough English translations[8]. For example, Query 0010 ("Mao Asada,"

[8]Query 0036 is misspelt, but we did not correct it. Current Web Search APIs can actually correct the spelling automatically.

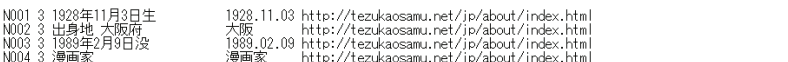

Figure 6: Some nuggets from 0031 "Osamu Tezuka" (CELEBRITY).

a figure skater) is a CELEBRITY query. We classfied 0023 "Kalashnikov" as a DEFINITION query, as the original question in the Yahoo! data set was "What is Kalashnikov?". As the second field for this query shows, it has five different interpretations in our data: a gun inventor's name, his son's name, the gun's name, and brands called "Kalashnikov vodka" and "Kalashnikov watch." Similarly, the LOCAL query 0037 has two interpretations, as there are two train stations named "Sanbonmatsu" in Japan. We intentionally included these ambiguous queries in the set, as we believe that handling such queries in DIIA evaluation will be important just as in traditional diversified ranked list evaluation [6, 20]. However, how to explictly encourage diversification in DIIA evaluation is beyond the scope of the present study.

4.2 Creating Nuggets

Through searching and browsing Web pages (mostly official pages or Wikipedia), the first author of this paper, who is a native Japanese speaker, created a total of 2,839 nuggets across the 60 queries. Statistics on the number of nuggets are shown in Table 1. The query with the highest number of nuggets (368) is 0031 "Osamu Tezuka" (See Figure 5), an extremely prolific cartoonist who died in 1989: about 300 nuggets correspond to comics and films that he created. A small portion of the nuggets for 0031 is shown in Figure 6: the first field is the nugget ID; the second field is the nugget grade (1, 2, or 3) assigned by the nugget creator (i.e., the first author); the third field is the *nugget semantics* in natural language, which represents an atomic piece of factual information and is used as a criterion for manually determining if a system output contains this nugget or not; the fourth field is the aforementioned *vital string* used for normalising S-measure; and the fifth field is the URL from which the nugget was extracted. Nuggets N001 and N003 say that Osamu Tezuka was born on November 3, 1928 and died on February 9, 1989, respectively; N002 says that he was born in Osaka; N004 says that he was a cartoonist; N009 and N013 say that he graduated from Osaka University in 1951 and that he got married in 1959; N014 says that his wife's name is Etsuko; and N015 says that he received a medical doctoral degree in 1961. The English translations of the corresponding vital strings would be: "Nov 3, 1928," "Osaka," "Feb 9, 1989," "cartoonist," "graduated," "married," "Etsuko" and "medical doctor."

The nugget grades assigned by the first author are subjective. For example, for 0031, all the comic book titles by Osamu Tezuka were given the grade of 1, as the first author judged that it is more important to include in the system output some basic facts about Osamu Tezuka (personal history etc.) than some of his comic titles selected from his lifetime's work. Note also that in Figure 6, the fact that he got married in 1959 (N013) is considered less important than the fact that he graduated from Osaka university in 1951 (N009). To remedy this subjectivity issue, the second author of this paper (also a native Japanese speaker) assigned his nugget grades (1, 2 or 3) independently, and the sum of these two grades were used as the final nugget weights in our experiments. The inter-assessor agreement was reasonable: 0.689 in terms of *quadratic-weighted kappa*. Note that, as in summarisation and QA evaluation, the nugget *creation* is still done by a single person.

Even though nugget creation depends on the view of a single person in our study, we devised a nugget creation policy document that can be shared by future nugget creators. An excerpt from the document is shown below:

(a) A nugget is a short factual statement such that an assessor can judge whether a given text shows or clearly implies that statement to be true.

(b) Information available on official Web pages is considered factual. Information available from other Web sources is considered factual provided that it does not contradict with the official information.

(c) Nuggets are built based on established facts as of December 31, 2010. Events that occur after this date will be ignored.

Item (b) was included based on the observation that information in official pages and that in Wikipedia are occasionally contradictory. For example, for the CELEBRITY query 0022, the official date of birth is different from the one in Wikipedia (as of December 31, 2010). In such a case, we assume that the *official* information is correct. Item (c) tries to free us from the burden of updating our nuggets indefinitely in response to future events. However, nuggets do become obsolete, sometimes unexpectedly quickly: on October 28, 2010, we finished constructing our *preliminary* nugget sets which contained a nugget representing a famous Japanese blog for the aforementioned query 0022 "Kaori Manabe" (who was once known as "Queen of Blogs"). However, to our surprise, the blog was shut down permanently on October 31, 2010! On the other hand, for the figure skater query 0010, we later had to *add* nuggets representing figure skating competitions that took place in November and December 2010. We thus re-examined all of the preliminary nugget sets and made revisions where necessary, to ensure that the final nugget sets represent facts as of December 31, 2010.

How efficient was the nugget creation process? The difficult part was devising the nugget creation policy document: deciding on what kinds of information to expect from DIIA systems in general as well as for each query type. Once this was done, the actual process of searching, browsing and collecting nuggets was not difficult. Although we did not measure the total time required for constructing the entire test collection (as the nugget creation policy writing and most of the actual nugget creation were done in parallel), we recorded the time spent for nugget creation for some typical cases. For example, it took about 68 minutes to construct 148 nuggets for a CELEBRITY query (0.46 minutes per nugget), and about 40 minutes to construct 31 nuggets for a LOCAL query (1.29 minutes per nugget). This includes the time to form nugget semantics and vital strings, to record URLs, and to decide on the first set of nugget grades as shown in Figure 6. In general, queries with many nuggets were created relatively efficiently, because they usually contained large lists (CDs released, books published, etc.) that are simple to handle. For DEFINITION and QA queries which generally have a very small number of nuggets, the time spent was typically 2-20 minutes per query. Our overall experience suggests that nugget creation for the DIIA task is just as feasible as that for QA and summarization or relevance assessments for traditional IR, provided that a clear nugget policy document and training material are given to the nugget creator. Note, however, that DIIA evaluation also requires a manual nugget match evaluation for each system, as will be described in the next section.

4.3 Nugget Match Evaluation Interface

To evaluate a system for the DIIA task, an X-string and the list of nuggets are compared manually. Figure 7 shows screenshots of the interface we have developed for this purpose. The left panel shows the X-string for a particular query. (This is an X-string from the manual DESKTOP run which will be described later). The nugget match evaluation interface can truncate each X-string into 500 (or 140) characters before evaluation. The right panel shows the list of nuggets for this query, in the PMO order (i.e., sorted first by the nugget weight and then by the vital string length). The assessor examines each nugget in turn, and decides whether it is covered by the X-string or not. If it is covered, he selects and records a nugget match area within the X-string by means of a mouse drag, as shown Figure 7 (a). In response to this user action, a Save button pops up, and the user can click on it. As a result, as shown in Figure 7 (b), the nugget match area is stored together with its start and end positions. The end position (44) is used as the offset for this nugget for computing S-measure later. The output from this interface is a list of nuggetID-offset pairs.

As was mentioned earlier, a nugget match area may overlap with or subsume another. We believe that this flexible feature is necessary for evaluating intelligent systems that go beyond extracting texts from source documents "as is". While the interface technically allows the assessor to record multiple matches per nugget, we use only the nugget match with the smallest offset (i.e., first nugget occurrence within the X-string) so that repetition is penalised when computing S-measure.

Given the output from the interface (i.e., a list of nuggetID-offset pairs), S-measure can easily be computed by automatically comparing it with the gold-standard that consists of nuggetIDs, nugget weights and the lengths of vital strings (See Eq. 2)[9]. In practice, it is probably useful to hire multiple assessors to evaluate each X-string, and to average the S-measure values across the assessors, which should serve two purposes: (a) trivial human errors (i.e., missing an existing nugget or locating a nonexistent nugget) can be detected by automatically comparing the assessment results; and (b) since recognition of nugget matches and selection of nugget match areas may vary across assessors, we can obtain more reliable results by averaging. In this study, however, only the first author performed nugget match evaluation. As the purpose of our DIIA experiments is to demonstrate the integrity, usefulness and limitations of our general approach, we argue that this is sufficient.

5. EXPERIMENTS

This section reports on a set of experiments using our DIIA test collection and S-measure. In addition to evaluating a pilot *automatic* run, we evaluated a *manual* run composed of X-strings constructed by hand for each query in order to estimate a *practical upperbound* for each query. This is because the maximum achievable S-measure value is often below one by design, as the denominator (i.e., normalisation factor) of S-measure is often unreachable by its numerator in Eq. 2. This normalisation issue arises from the fact that the Pseudo Minimal Output (PMO) is constructed using vital strings, which are probably necessary to be included in the system output, but probably not sufficient to convey the meaning of the nugget to the user. In addition, we compare the behaviour of S-measure with the traditional weighted nugget recall, which is equivalent to S-measure without offset-based discounting.

5.1 Manual Run

We devised a manual run in order to approximate S-measure values that are *actually* achievable if we have ideal information re-

[9]A software for computing S-measure is publicly available at `http://research.nii.ac.jp/ntcir/tools/ntcireval-en.html`.

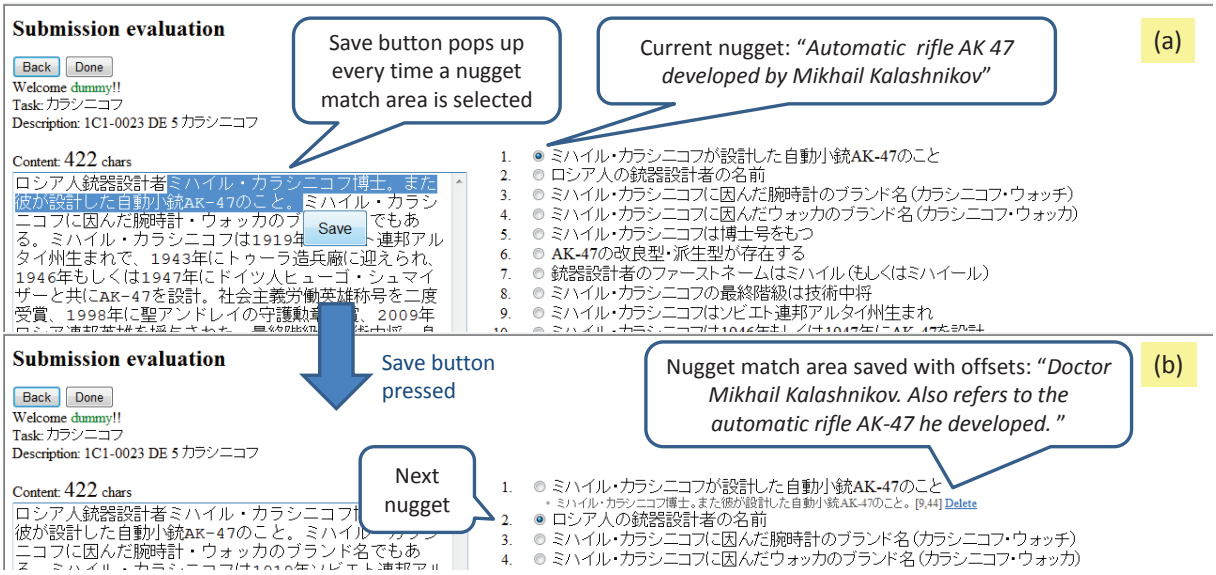

Figure 7: Nugget Match Evaluation Interface.

trieval and natural language processing techniques. Each output of the manual run is perfectly readable, in contrast to a PMO which may lack a lot of necessary information and may be defective as a natural language text.

Each X-string of our manual run was prepared as follows: the first author looked at the complete list of nuggets in the PMO order. Starting with an empty output text file, he added, for each nugget, a natural language text that concisely expresses the nugget semantics. (The text is often similar to the nugget semantics itself.) While he basically respected the original nugget order, he reordered some nuggets so that similar kinds of information are output close to each other. For example, even if a phone number is ranked at 1 and a fax number is ranked at number 5 in the PMO, he put the phone number and the fax number together in the output file (by basically respecting the rank of the first nugget). Similarly, for example, a list of book titles were output as one block even if they were spread across within the PMO, as the latter situation is not natural to the human eye. He also consolidated multiple nuggets where appropriate: recall the phone/fax number example discussed in Section 3.3. In short, he used common sense to re-arrange and merge the pieces of information conveyed in the PMO from the viewpoint of readability and conciseness. As the manual run was created by relying on the "right answers," it should be regarded as an oracle knowledge source run (See Section 4).

Figure 7 includes the first part of the X-string from our manual run for 0023 "Kalashnikov" (DEFINITION). The first sentence in this X-string means "Dr. Mikhail Kalashnikov, the Russian gun designer" and this sentence covers the second nugget "Kalashnikov is the name of a Russian gun designer" and the fifth nugget "Mikhail Kalashnikov has a Ph.D." The second sentence in the X-string means "Also refers to AK-47, the automatic rifle he designed" and this sentence covers the first nugget, "Kalashnikov means AK-47, an automatic rifle designed by Mikhail Kalashnikov."

The original manual run was created so that each X-string is around 700 characters long. Then, the nugget match evaluation interface truncated each X-string to 500 and 140 characters to produce DESKTOP and MOBILE runs, respectvely.

5.2 Automatic Run

We also generated an automatic run by assuming that the correct query types are known to the DIIA system[10]. Using the query type information and the Bing API, our automatic run adopted the simple heuristics shown in Figure 8 for each query q. Hence this is an open knowledge source run (See Section 4). Just like the manual runs, the automatic DESKTOP and MOBILE runs were created from the same 700-character run by truncation. Note that these runs are just *examples* for demonstrating the integrity of our DIIA evaluation framework.

CELEBRITY
1. Submit query "q", and obtain the snippet of the first retrieved URL that matches the pattern "*ja.wikipedia.org*";
2. Submit queries "q 公式[official]" and "q 名鑑[name encyclopaedia]", obtain the top-ranked snippets, and concatenate to the above.

LOCAL
Submit queries "q 住所[address]", "q 所在[location]", "q 電話[phone]", "q アクセス[access]", "q 時間[hour]", obtain the top-ranked snippets and concatenate.

DEFINITION
1. Submit query "qとは [what is q]" and obtain the top-ranked snippet;
2. Submit query "q", and obtain the snippet of the first retrieved URL that matches the pattern "*ja.wikipedia.org*". Concatenate to the above.

QA
Submit query "q", obtain the snippets of retrieved URLs that match the patterns "*chiebukuro.yahoo.co.jp*", "*oshiete.goo.ne.jp*", "*q.hatena.ne.jp*" and "*questionbox.jp.msn.com*", and concatenate.

Figure 8: The automatic run heuristics. (The URL pattern "ja.wikipedia.org" represents Japanese Wikipedia pages and the four URL patterns for the QA queries represent Japanese community QA sites.)

5.3 Results and Discussions

Table 2 shows the mean S-measure (in bold) and weighted nugget recall (in italics) values for our four runs (Manual/Automatic × DESKTOP/MOBILE). It can be observed that S-measure behaves similarly to weighted recall on average: the DESKTOP runs (500

[10] Another approach would be to provide the DIIA system with some training queries (which we lacked at this point), and make it determine the query type of each given query using an automatic classification technique.

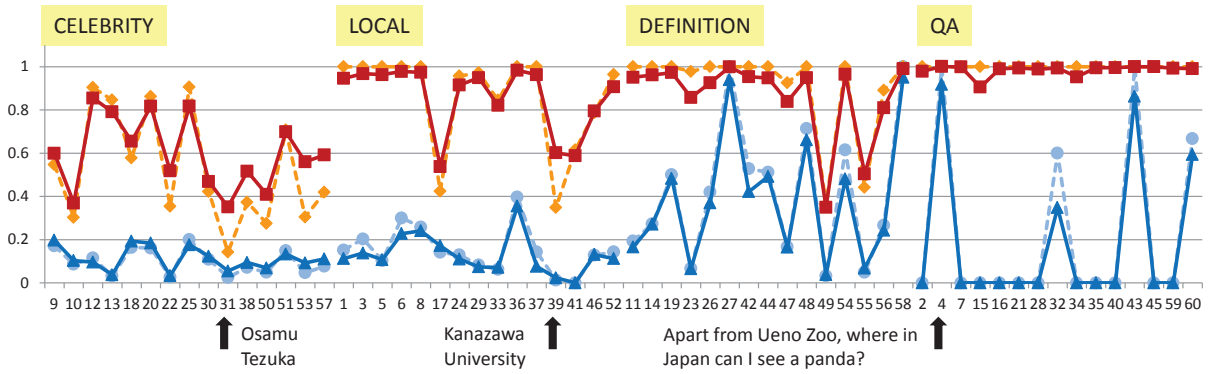

Figure 9: Per-query DESKTOP run performances: manual run S-measure / weighted recall (red solid / orange dotted lines); automatic run S-measure / weighted recall (blue solid / light blue dotted lines). The x-axis shows the query IDs.

Table 2: Manual and automatic run mean performances: S-measure (in bold) / weighted recall (in italics). D:DESKTOP; M:MOBILE.

	Manual-D	Auto-D	Manual-M	Auto-M
CELEBRITY	**.601**/*.530*	**.113**/*.099*	**.322**/*.244*	**.038**/*.032*
LOCAL	**.859**/*.861*	**.130**/*.150*	**.491**/*.460*	**.044**/*.043*
DEFINITION	**.865**/*.907*	**.388**/*.423*	**.702**/*.715*	**.294**/*.304*
QA	**.985**/*1.000*	**.182**/*.218*	**.954**/*.963*	**.079**/*.085*
all	**.828**/*.824*	**.203**/*.222*	**.617**/*.596*	**.114**/*.116*

characters) naturally outperform the corresponding MOBILE runs (140 characters); the automatic runs do better with DEFINITION queries than with others; and the manual runs substantially outperform the automatic runs, but are far from "perfect."

There two reasons why the mean S-measure across all queries does not reach one even for the manual DESKTOP run (.828). The first is simply because of the output widow size of 500: for some queries, it is simply impossible to list up all nuggets within this window, and this is exactly why the mean weighted recall is also below one (.824). The second reason is that the maximum possible S-measure value is often smaller than one.

Figure 9 visualises S-measure and weighted nugget recall for the manual and automatic DESKTOP runs per query. The performance of any DIIA DESKTOP run is expected to lie below the manual DESKTOP curves as the latter represents practical upperbounds. It can be observed, for example, that the practical upperbounds are actually 1 for the QA queries as they have few nuggets per query, while those for some CELEBRITY queries are indeed quite low, for exactly the reasons discussed above.

At the other extreme, there was exactly one query for which the S-measure only slightly *exceeded* 1, though this is not visible in Figure 9. The QA query 0004 "Apart from Ueno Zoo, where in Japan can I see a panda?" has the following four nuggets:

N001 Adventure World has pandas (vital string length = 11 for "Adventure World" in Japanese); weight = 6;

N002 Adventure World is in Wakayama prefecture (vital string length = 3 for "Wakayama" in Japanese); weight = 4;

N003 Oji Zoo has pandas (vital string length = 5 for "Oji Zoo" in Japanese); weight = 6;

N004 Oji Zoo is in Kobe city (vital string length = 2 for "Kobe" in Japanese); weight = 4.

Therefore, its PMO ranks the nuggets as <N003 N001 N004 N002> by using the weights as the first sort key and the lengths as the second sort key. Whereas, the manually composed X-string was "Oji Zoo (Kobe), Adventure World (Wakayama)" [English translation]. This covers all four nuggets, and is of the form <N003 N004 N001 N002> after removing punctuation marks etc., and it happens to outperform the PMO very slightly. As a result, the S-measure for this query was 1.001. Note that this is exactly one of the potential problems we discussed in Section 3.3. However, as was discussed earlier, this problem can be avoided by using the S♭-measure (Eq. 3) instead.

Let us now compare S-measure with weighted recall using Figure 9. It can be observed that, while S-measure generally resembles weighted recall, there are many cases where it is less than one even though weighted recall is one. This means that S-measure is more demanding for these queries. For example, see the manual DESKTOP performances for the LOCAL and DEFINITION queries. Also, for the aforementioned "panda" query 0004, our automatic DESKTOP run achieved a weighted recall of one but an S-measure of around 0.9: the first part of the X-string is shown in Figure 10. As the English translation shows, the red block in this figure covers all of the aforementioned four nuggets. However, because this X-string was automatically generated from search engine snippets, the part preceding the red block is completely irrelevant. That is, while this output achieves perfect nugget recall, it is still not satisfactory in terms of *Immediate* Information Access: the automatic run fails to minimise the amount of text that the user has to read in order to obtain the desired information. There were other similar cases in our experiments, which suggest that evaluation with S-measure will help us design our system output strategy carefully for DIIA.

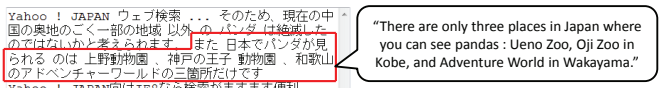

Figure 10: Part of the X-string from the automatic run for Query 0004.

Finally, let us discuss a few cases where S-measure is *greater* than weighted recall. In Figure 9, the S-measure of the manual DESKTOP run is higher than the weighted recall for 0031 "Osamu Tezuka" (CELEBRITY). Recall that this is the query with the largest number of nuggets (368). The manual DESKTOP run returned only 35 nugget matches for this query (unweighted recall= $35/368 = .095$). However, as the X-string basically showed the nuggets in order of importance (as this was the strategy used for creating the manual runs), the S-measure was .351, while the weighted recall was .142. Similarly, for 0039 "Kanazawa University" (LOCAL) which had 125 nuggets, the manual DESKTOP run

returned 31 nugget matches, and the S-measure was .602 while the weighted recall was .348. These examples show that S-measure can reward systems that present important nuggets first, even when their nugget recall values are low.

6. CONCLUSIONS

We defined the Direct and Immediate Information Access (DIIA) task, where the system is expected to satisfy the user's need very quickly with its very first textual output, and proposed an evaluation framework for it. We proposed a simple evaluation metric called S-measure that takes the positions of nugget matches into account, and built a real DIIA test collection and an offset-based nugget match evaluation interface. Our experiments have demonstrated the integrity and usefulness of our framework. In particular, we have verified that S-measure rewards systems that reflect nugget importance, and those that return nuggets near the beginning of the output string. We also demonstrated that S-measure usually lies within the 0-1 range, and showed a simple variant of S-measure called S♭-measure which is strictly bounded above by 1.

Clearly, there are limitations to the present approach. The first is that S-measure is purely a content ranking measure, and that it does not consider linguistic quality. As we discussed in Section 2.2, this is a problem inherent in all nugget-based approaches. One possible problem with S-measure in particular is that there may be cases where presenting important information first may actually hurt readability. We thus recommend that readability evaluation be done separately for evaluating DIIA. Also, note that while S-measure penalises presentation of nonrelevant information before relevant information as in Figure 10, it does *not* explicitly penalise presentation of nonrelevant information *after* relevant information [23, 24]. This is similar to IR evaluation that treats a list of nonrelevant documents and an empty list as equally useless.

The second limitation is that we can only evaluate plain text outputs, even though real DIIA systems will probably use highlighting, different font sizes, and even non-textual presentations.

The third limitation is that, as we have observed in Section 4.2, our nuggets may become obsolete relatively quickly even though we are already restricting ourselves to retrieval of "established" facts. To address this problem, building a semi-automated *truth maintenance system* for nuggets may be useful. Such a system could periodically monitor the URLs that the current nuggets rely on (See Figure 6), and notify the test collection builder if the contents of the URLs have been revised. Note, however, that while some queries may require frequent updating, others do not: for example, many DEFINITION and QA queries do not require frequent updating, and the same goes even for some CELEBRITY queries, particularly for people who have passed away.

Despite the above limitations, we believe that our framework is a useful departure from and complement to traditional ranked retrieval evaluation. Since we explictly encourage *concise* presentations of important information, the DIIA task may foster research in abstractive text presentation methods as opposed to simple extractive ones (e.g. sentence selection). As future work, we plan to extend our DIIA experiments to other languages such as English. Our framework may also be useful to other related applications, such as search engine snippets and Bing's hover previews.

7. REFERENCES

[1] J. Allan, B. Carterette, and J. Lewis. When will information retrieval be "good enough"? In *Proceedings of ACM SIGIR 2005*, pages 433–440, 2005.

[2] O. Babko-Malaya. Annotation of nuggets and relevance in gale distillation evaluation. In *Proceedings of LREC 2008*, pages 3578–3584, 2008.

[3] P. Bailey, N. Craswell, R. W. White, L. Chen, A. Satyanarayana, and S. M. M. Tahaghoghi. Evaluating search systems using result page context. In *Proceedings of IIiX 2010*, 2010.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML 2005*, 2005.

[5] 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.

[6] C. L. Clarke, N. Craswell, I. Soboroff, and A. Ashkan. A comparative analysis of cascade measures for novelty and diversity. In *Proceedings of ACM WSDM 2011*, 2011.

[7] H. T. Dang and J. Lin. Different structures for evaluating answers to complex questions: Pyramids won't topple, and neither will human assessors. In *Proceedings of ACL 2007*, pages 768–775, 2007.

[8] M. D. Dunlop. Time, relevance and interaction modelling for information retrieval. In *Proceedings of ACM SIGIR '97*, 1997.

[9] E. Hovy, C.-Y. Lin, and L. Zhou. Evaluating DUC 2005 using basic elements. In *Proceedings of DUC 2005*, 2005.

[10] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[11] R. R. Korfhage. *Retrieval Effectiveness Measures*, chapter 8, pages 191–218. Wiley Computer Publishing, 1997.

[12] J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and PC internet search. In *Proceedings of ACM SIGIR 2009*, pages 43–50, 2009.

[13] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL 2004 Workshop on Text Summarization Branches Out*, 2004.

[14] J. Lin and D. Demner-Fushman. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9(5):565–587, 2006.

[15] T. Mitamura, H. Shima, T. Sakai, N. Kando, T. Mori, K. Takeda, C.-Y. Lin, R. Song, C.-J. Lin, and C.-W. Lee. Overview of the NTCIR-8 ACLIA tasks: Advanced cross-lingual information access. In *Proceedings of NTCIR-8*, pages 15–24, 2010.

[16] H. Nanba, K. Hirahara, and T. Takezawa. Hiroshima city university at evaluation subtask in the NTCIR-8 patent translation task. In *Proceedings of NTCIR-8*, pages 415–419, 2010.

[17] A. Nenkova, R. Passonneau, 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.

[18] T. Sakai. Alternatives to bpref. In *Proceedings of ACM SIGIR 2007*, pages 71–78, 2007.

[19] T. Sakai, D. Ishikawa, N. Kando, Y. Seki, K. Kuriyama, and C.-Y. Lin. Using graded-relevance metrics for evaluating community QA answer selection. In *Proceedings of ACM WSDM 2011*, pages 187–196, 2011.

[20] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of ACM SIGIR 2011*, pages 1043–1052, 2011.

[21] A. Turpin, F. Scholer, K. Järvelin, M. Wu, and J. S. Culpepper. Including summaries in system evaluation. In *Proceedings of ACM SIGIR 2009*, pages 508–515, 2009.

[22] J. V. White, D. Hunter, and J. D. Goldstein. Statistical evaluation of information distillation systems. In *Proceedings of LREC 2008*, pages 3598–3604, 2008.

[23] Y. Yang and A. Lad. Modeling expected utility of multi-session information distillation. In *Proceedings of ICTIR 2009*, pages 164–175, 2009.

[24] Y. Yang, A. Lad, N. Lao, A. Harpale, B. Kisiel, M. Rogati, J. Zhang, J. Carbonell, P. Brusilovsky, and D. He. Utility-based information distillation over temporally sequenced documents. In *Proceedings of ACM SIGIR 2007*, pages 31–38, 2007.

[25] L. Zhou, C.-Y. Lin, D. S. Munteanu, and E. Hovy. ParaEval: Using paraphrases to evaluate summaries automatically. In *Proceedings of HLT-NAACL 2006*, 2006.