Proceedings of NTCIR-4, Tokyo, April 2003 - June 2004

Toshiba ASKMi at NTCIR-4 QAC2

Tetsuya Sakai Yoshimi Saito Yumi Ichimura
Makoto Koyama Tomoharu Kokubu
Knowledge Media Laboratory, Toshiba Corporate R&D Center
tetsuya.sakai@toshiba.co.jp

Abstract

*Toshiba participated in NTCIR-4 QAC2 Subtask 1: this is our first QAC participation. Using our newly developed Japanese QA system called ASKMi, we submitted two runs, one using Okapi/BM25 for document retrieval (*TSB-A*) and the other using Boolean AND constraints before applying Okapi/BM25 (*TSB-B*). We achieved the 5th best performance among the 17 participants (8th and 9th among the 25 submitted runs). This paper briefly describes ASKMi, and analyses the formal run and oracle run results using Reciprocal Rank, as well as a new performance metric called Q-measure which can handle multiple-answer questions and answer correctness levels.*

Keywords: *ASKMi, Q-measure.*

1 Introduction

Toshiba participated in NTCIR-4 QAC2 Subtask 1: this is our first QAC participation. Using our newly developed Japanese QA system called ASKMi ("*ask me*") [9], we submitted two runs, TSB-A (**T**O**S**HI**B**A **A**SKMi) and TSB-B (**T**O**S**HI**B**A ASKMi **B**oolean). The only difference between them is the document retrieval strategy: TSB-A used Okapi/BM25 [13], while TSB-B used a Boolean AND constraint before applying Okapi/BM25 (See Section 2.4). Table 1 provides a quick summary of our official results. TSB-A and TSB-B achieved the 8th and 9th best performance among the 25 submitted runs, which places us 5th among the 17 participants. (Our analyses are based on the answer file QAC2formalAnsTask1_040308 throughout this paper.)

The remainder of this paper is organised as follows. Section 2 briefly describes ASKMi. Section 3 briefly describes how *Q-measure*, an information retrieval performance metric for multigrade relevance, can be applied to QA evaluation. Using both Reciprocal Rank and Q-measure, Section 4 analyses our formal run results as well as some "oracle" runs (e.g. [14]). Finally, Section 5 concludes this paper. We report on our work for the NTCIR-4 CLIR task in a separate paper [10].

Table 1. TSB Formal Run Results based on QAC2formalAnsTask1_040308.

Run Name	MRR	Description
TSB-A	0.454	ASKMi (BM25)
TSB-B	0.440	ASKMi (BM25+Boolean)

2 ASKMi

ASKMi stands for "Answer Seeker/Knowledge Miner": Figure 1 shows its configuration. The Knowledge Miner consists of off-line components, namely, the *Semantic Class Recognizer* and the *Relation Extractor*. The Answer Seeker consists of on-line components, namely, the *Question Analyzer*, the *Retriever/Passage Extractor*, and the *Answer Formulator*. Sections 2.1-2.5 briefly describe each component: More details can be found in [9].

2.1 Semantic Class Recognizer

The basic function of the Semantic Class Recognizer is rule-based named entity recognition. For documents, it performs *candidate answer extraction* (or *predictive annotation* [5]): The strings extracted through this process are called *ex-strings*, which are associated with *answer extraction confidence* values. For questions, it performs *question abstraction* [9, 12]: For example, given Japanese questions such as "*Toshiba no shachō* (Toshiba's president)" and "*Maikurosofuto no fukushachō* (Microsoft's vice president)", question abstraction converts them into "COMPANY *no* POSITION", where *no* is the Japanese possessive particle.

Currently, the Semantic Class Recognizer maintains an Answer Type Taxonomy that consists of more than 100 answer types. In [4], we have studied how named entity recognition performance and answer type granularity affect QA performance using the QAC1 test collection.

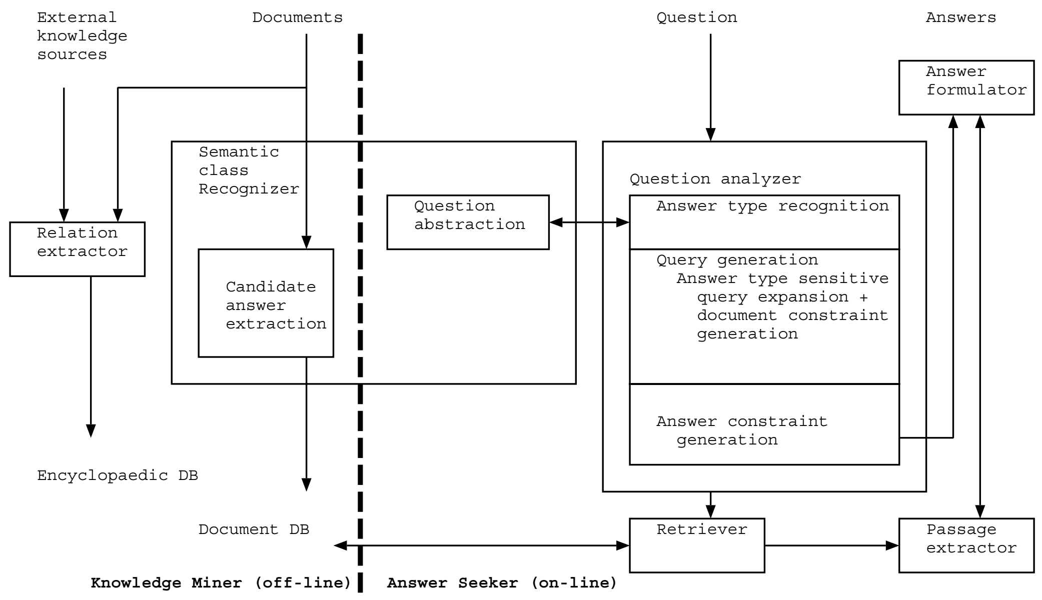
External knowledge sources
Documents
Question
Answers
Answer formulator
Semantic class Recognizer
Question abstraction
Question analyzer
Answer type recognition
Relation extractor
Query generation Answer type sensitive query expansion + document constraint generation
Candidate answer extraction
Answer constraint generation
Encyclopaedic DB
Document DB
Retriever
Passage extractor
Knowledge Miner (off-line)
Answer Seeker (on-line)

Figure 1. ASKMi configuration.

2.2 Relation Extractor

The Relation Extractor has been designed to collect encyclopaedic knowledge from free-text documents and other knowledge sources such as web pages. It creates an *encyclopaedic DB* for dealing with questions like "DVD *towa nani*? (What is DVD?)". However, for QAC2, the Relation Extractor managed to answer only one question, namely, QAC2-10036-01 "What does ADSL stand for?". As future work, we plan to improve the accuracy and coverage of the Relation Extractor by utilising the Semantic Class Recognizer for collecting encyclopaedic knowledge.

2.3 Question Analyzer

The Question Analyzer has several functions including *answer type recognition*, *query generation* and *answer constraint generation*. These functions rely on *Semantic Role Analysis (SRA)*, which can assign arbitrary labels to textual fragments based on pattern matching rules with priorities among them [6, 7, 9].

2.3.1 Answer Type Recognition

When ASKMi receives a question from the user, it goes through the following steps:

1. Question Abstraction: As mentioned earlier, both "*Toshiba no shachō* (Toshiba's president)" and "*Maikurosofuto no fukushachō* (Microsoft's vice president)" are transformed into "COMPANY *no* POSITION".

2. Question Type Recognition: Based on Japanese interrogatives found within the question string, a *question type* such as NANI (what), DARE (who), ITSU_KARA (from when) or DOKO_HE (to where) is assigned to it.

3. Answer Type Recognition (or *question classification* [2]): Each question class is mapped to one or more *answer category* (i.e. a group of possible answer types). Then, SRA pattern matching is applied to determine the final answer types. To each answer type, an *answer type recognition confidence* value is attached.

In [12], we have shown that Question Abstraction can be utilised for enhancing document retrieval performance for the purpose of QA.

2.3.2 Query Generation

Query generation refers not only to generating a set of search terms from a natural language question, but also to answer type sensitive *query expansion* and *document constraint generation*.

Answer type sensitive query expansion is designed to enhance document retrieval performance. For example, if the question string contains the Japanese expression "*umareta* (was born)", and if the expected answer type is "DATE", then words such as "*seinengappi* (date of birth)" are added to the original question. In contrast, if the expected answer type is "LOCATION", then words such as "*shusshinchi* (place of birth)" are added. Of course, answer type *insensitive* query expansion is also possible: If the question string contains expressions such as "*mottomo takai* (highest)" or "*mottomo ōkii* (largest)", then a single *kanji* word that conveys the same meaning ("*saikō*" or "*saidai*") can be added regardless of answer types.

Answer type sensitive document constraint generation creates a database restriction condition from the input question in order to perform high-precision search of the document DB. For INCIDENT/ACCIDENT-related questions such as "*2001 nen 9 gatsu 11 nichi ni naniga okitaka* (What happened on September 11, 2001?)", ASKMi can temporarily eliminate all newspaper articles published before September 11, 2001 from the database, thus reducing the search space. In contrast, for EVENT-related questions such as those concerning the Olympic games, then database restriction may not be applied, because such major events may be mentioned in newspapers well before they actually take place.

2.3.3 Answer Constraint Generation

Answer constraint generation includes "common sense" techniques that are similar to *expected numerical range* [3]. Answer constraints can also be applied based on Japanese particles: For example, if the question class is DOKO_HE (to where), answers which are immediately followed by "*he*" or "*ni*" (Japanese particles indicating destination) are preferred. Such answer constraints heuristically modify the original answer scores at the final answer formulation stage.

2.4 Retriever/Passage extractor

The ASKMi Retriever uses the same basic components as the BRIDJE system [7, 10]. For a given question q and each document d, it calculates the original document score ($origdscore(q,d)$) using Okapi/BM25 term weighting [13], and retrieves a fixed number of documents that contain candidate answer strings annotated with designated answer types. The Retriever can optionally perform Boolean search prior to document ranking, which may enhance retrieval precision at the cost of reducing retrieval recall. It can also perform Pseudo-Relevance Feedback (e.g. [10]), although it was not used for QAC2.

For QAC2, TSB-A used the Okapi/BM25 term weighting. Whereas, TSB-B used Boolean AND operators prior to term weighting if the number of query terms was less than four. If there were four or more query terms, or if the Boolean search was not successful, TSB-B fell back to the TSB-A strategy.

We did not use the Passage Extractor even though it has several static and dynamic passage selection functions [9]. This is because we have not found a passage selection method that is significantly more effective than using the whole document in terms of the final QA performance. Our comparison of several static passage selection methods in terms of *document retrieval* performance can be found in [8].

2.5 Answer Formulator

The Answer Formulator calculates the scores of candidate answers that are included in the passages.

An *ex-string* e is a quadruple <*document*, *answertype*, *string*, *position*> that has been extracted through candidate answer extraction. It represents a specific occurrence of a candidate answer within a document. A *candidate* c is a triple $< document, answertype, string >$, which is obtained by consolidating the ex-strings (i.e. multiple occurrences) within a document. An *answer string* a is obtained by consolidating candidates across documents and across answer types. Let $C(e)$ and $A(c)$ represent the mapping from ex-strings to the corresponding candidates, and that from candidates to corresponding answer strings, respectively.

Let e be an ex-string from a document d. Let t be a query term found in d, and let p be a passage extracted from d. Firstly, we define the *co-occurrence function* as follows:

$$distance(e,t,p) = \min_i |pos(e,p) - pos'(t,i,p)| \quad (1)$$

$$cooc(e,t,p) = \frac{1}{\sqrt{(1 + P_C * distance(e,t,p))}} \quad (2)$$

where

$pos(e,p) =$	position of ex-string e within p;
$pos'(t,i,p) =$	position of the i-th occurrence of t within p;
$P_C =$	constant called the *co-occurrence parameter* ($P_C \geq 0$).

For a given question q and a document d, the score of an ex-string ($escore$) and that of a candidate ($cscore$) are calculated as follows:

$$escore(q,e) =$$

$$cf_{atr}(q,e) * cf_{aex}(e) * \max_p\left(\frac{1}{|q|}\sum_{t \in q} cooc(e,t,p)\right) \quad (3)$$

$$cscore(q,c) = \max_{e, C(e)=c} escore(q,e) \quad (4)$$

where

$cf_{atr}(q,e) =$	answer type recognition confidence for the answer type of e, given q;
$cf_{aex}(e) =$	answer extraction confidence for e.

As for the BM25-based document score (See Section 2.4), it can optionally be modified as follows [9, 12]:

$$dscore(q,d) = \max\{0, P_D * (origdscore(q,d) - 1) + 1\} \quad (5)$$

where

$P_D =$	constant called the *document score parameter* ($P_D \geq 0$).

Let $D(c)$ represent the mapping from a candidate c to the corresponding document. We can now calculate the score of an answer string a:

$$ascore(q,a) = \sum_{c, A(c)=a} dscore(q, D(c)) * cscore(q,c) \quad (6)$$

Finally, as mentioned in Section 2.3.3, the above answer scores are heuristically adjusted based on answer constraints. Moreover, *answer string consolidation* is performed in order to prevent returning multiple answers that mean the same thing. Again, this process is answer type sensitive: For example, if the answer type is POLITICIAN, then an answer string "*Junichiro Koizumi*" absorbs other "less complete" candidates such as "*Koizumi*" and "*Junichiro*". On the other hand, if the answer type is COMPANY, then answer string consolidation is *not* performed, because, for example, "*Toshiba EMI*" and "*Toshiba*" are different companies even though the former contains the latter as a substring. Sakai [11] has shown the effectiveness of answer string consolidation using the QAC1 test collection.

3 Application of Q-measure to QA

Although NTCIR QAC uses Reciprocal Rank for Subtask 1 (i.e. "single-answer" task) and F-measure for Subtask 2 (i.e. "list" task), many seemingly "single-answer" questions do in fact have multiple answers and it is difficult to draw a line between these two tasks. Reciprocal Rank cannot evaluate multiple answers while F-measure (and TREC Accuracy [17]) cannot take answer ranking into account. Moreover, the above QA evaluation metrics cannot handle *answer correctness levels*, even though some answers may be "more correct" than others just as some documents may be more relevant than others in IR.

Sakai [11] has proposed a new evaluation metric called *Q-measure*, which can handle both single-answer and multiple-answer questions, *and* can handle answer correctness levels. His experiments using the QAC1 test collection suggests that Q-measure is a better QA metric than Reciprocal Rank. We therefore use Q-measure along with Reciprocal Rank for analysing the QAC2 results.

The mathematical definition of Q-measure (as an *information retrieval* performance metric) can be found in [10, 11]. Below, we briefly describe how to modify the answer data of a traditional "exact answer" QA test collection (such as QAC2) so that Q-measure can be applied to QA evaluation.

Step 1 Construct *answer synsets*, or equivalence class of answer strings, in order to avoid rewarding systems that return "duplicate" answers. In fact, the equivalence classes are *already included* in the QAC1 and QAC2 test collections for evaluation with F-measure based on answer *instances*. Therefore this is not difficult.

Step 2 For each answer string in each answer synset, assign a *correctness level*. For example, following the NTCIR document relevance levels, we can use "S-correct" (an excellent answer), "A-correct" (a good answer), and "B-correct" (an adequate answer) as answer correctness levels. This may be more difficult than constructing answer synsets, but if it is difficult for some questions, assigning "flat" correctness levels (e.g. treating all answer strings as A-relevant) for these questions would suffice.

Table 2 provides some examples of how we actually modified the QAC2 Subtask 1 answer data. For each question, the number of answer synsets is denoted by R, and the i-th answer synset is denoted by $AS(i)$.

QAC2-10001-01, 10074-01 and 10177-01 are the simplest examples: There is only one answer synset, but the answer strings vary in *informativeness* or *completeness*. For example, as "Yoshii" and "Mr. Sato" are relatively common Japanese surnames, they are considered to be B-correct. For 10031-01 "Where did Antonio Inoki's retirement match take place?", we constructed only one answer synset (even though the original QAC2 answer data treats the two answer strings as distinct instances), because "Bunkyo-ku, Tokyo" is merely the address of "Tokyo Dome", and may be a little awkward as an answer to the above question. 10049-01 is an example of assigning correctness levels from the viewpoint of how *accurate* the answer is: Since the truly correct answer is "thirty-two years", probably "over thirty years" is a good answer, but "thirty years" is probably inaccurate. For 10079-01 "What is the abbreviation for Deoxyribonucleic Acid?", "DNA (Deoxyribonucleic Acid)" is a "not exact" answer [17] although it is treated as correct in the QAC2 data. We therefore treated this string as B-correct.

We now look at the examples with multiple answer synsets. For QAC2-10135-01, there are two answer synsets, each representing a famous artist. As "Rodin" and "Klimt" are not common names in Japan, they are treated as A-correct here, in contrast to "Yoshii" for 10001-01 and "Mr. Sato" for 10074-01 which were treated as B-correct. (Note that, just as document relevance criteria may vary across topics, answer correctness criteria may naturally vary across questions.) For 10124-01 and 10157-01, there are as many as 7 and 10 answer synsets, respectively. Reciprocal Rank is clearly inadequate for dealing with these questions. The answer strings for 10157-01 are all A-correct rather than S-correct, as none of them contains the first name.

Constructing answer synsets and assigning correct-

Table 2. Examples of QAC2 answer synsets and correctness levels.

QAC2-10001-01 ($R = 1$) $AS(1) = \{<$“Masato Yoshii”,$S>$, $<$“Yoshii”,$B>\}$
QAC2-10031-01 ($R = 1$) $AS(1) = \{<$“Tokyo Dome”,$S>$, $<$“Bunkyo-ku, Tokyo”,$A>\}$
QAC2-10049-01 ($R = 1$) $AS(1) = \{\ldots, <$“thirty-two years”,$S>$, $<$“over thirty years”,$A>$, $<$“thirty years”,$B>\}$
QAC2-10074-01 ($R = 1$) $AS(1) = \{<$“former prime minister Eisaku Sato”, $S>$, $<$“Eisaku Sato”, $S>$, $<$“Mr. Sato”,$B>\}$
QAC2-10079-01 ($R = 1$) $AS(1) = \{<$“DNA”, $S>$, $<$“DNA (Deoxyribonucleic Acid)”, $B>\}$
QAC2-10124-01 ($R = 7$) $AS(1) = \{<$“Amalthea”, $S>, \ldots\}$ $AS(2) = \{<$“Adrastea”, $S>\}$ $AS(3) = \{<$“Io”, $S>$, $<$“Satellite Io”, $B>\}$ $\vdots$
QAC2-10135-01 ($R = 2$) $AS(1) = \{<$“Auguste Rodin”, $S>$, $<$“Rodin”, $A>\}$ $AS(2) = \{<$“Gustav Klimt”, $S>$, $<$“Klimt”, $A>\}$
QAC2-10157-01 ($R = 10$) $AS(1) = \{<$“Lenin”, $A>\}$ $AS(2) = \{<$“Former Prime Minister Stalin”, $A>$, $<$“Stalin”, $A>\}$ $\vdots$ $AS(7) = \{<$“Former President Gorbachev”, $A>$, $<$“Gorbachev”, $A>\}$ $\vdots$
QAC2-10177-01 ($R = 1$) $AS(1) = \{<$“New Delhi, India”, $S>$, $<$“New Delhi”, $A>$, $<$“India”, $A>\}$

ness levels require subjective judgment. However, experience from IR and QA evaluations suggests that differences in subjective judgments have limited impact on performance comparisons [15, 16], and this paper assumes that the above finding holds for answer synset construction and correctness level assignment. We plan to test this hypothesis in the near future.

Although it is not impossible to include *supporting document* information in the process of answer synset construction and correctness level assignment, we use the *lenient* evaluation methodology [17] even though QAC2 officially uses the *strict* one. This is because (a) While providing a supporting passage is probably of practical importance, forcing a system to name a supporting document from a closed document collection may not be the best way to evaluate open-domain QA; and (b) The “lenient MRRs” of TSB-A and TSB-B are actually identical to the “strict” (i.e. official) ones. Thus, given a correct answer, finding a good supporting document is not difficult with ASKMi.

We are now ready to apply Q-measure to QA evaluation, by treating any ranked output of a QA system as a *document retrieval* output. Figures 2 and 3 show how to calculate Q-measure (and R-measure [11]) for QA evaluation. The first algorithm identifies the correct answer strings, but ignores duplicate answers. The second algorithm calculates Q-measure exactly as defined in [10, 11]. Hereafter we use $gain(S) = 3, gain(A) = 2$ and $gain(B) = 1$ as the *gain values*. Ideally, the system output size L should be sufficiently large so that $L \geq R$ for all questions. However, we follow the TREC/NTCIR traditions and use $L \leq L' = 5$ [11]. Although the algorithms can properly handle NIL questions [11], we exclude QAC2-10198-01 and 10199-01 from our evaluation, following the official MRR calculation method used at QAC2.

4 Analysis

4.1 Formal runs

The Boolean AND constraints used for TSB-B were not at all successful: it improved only one question and hurt four questions. Although a *question focus* type of approach [1] may be useful for determining which of the query terms should be included in the Boolean expression, we have not found an effective way to do this. Hereafter, we analyse TSB-A only.

Table 3 provides a failure analysis for TSB-A, following our analysis method used for the QAC1 additional questions in [9]. Answer type recognition is responsible for Rows (b) and (e), the Retriever is responsible for Row (c), and the Semantic Class Recognizer

```
/* initialize flag for each answer synset.
The flags avoid marking multiple answers from
the same answer synset. */
for( i=1; i<=R; i++ ) flag[i]=0;

r=1; /* system output rank */
while read o(r){ /* system's r-th answer */
  if( there exists a(i,j) s.t. o(r)==a(i,j) ){
  /* o(r) matches with a correct answer */
    if( o(r)=="NIL" ){
    /* special treatment of NIL */
      if( r==1 ){ /* i.e. NIL at Rank 1 */
        print o(r), x(i,j);
        /* marked as correct */
      }
      else{
        print o(r);
        /* NOT marked as correct */
      }
    }
    else{ /* not NIL */
      if( flag[i]==0 ){
      /* AS(i) is a NEW answer synset */
        print o(r), x(i,j);
        /* marked as correct */
        flag[i]=1;
      }
      else{ /* i.e. flag[i]==1 */
        print o(r);
        /* duplicate answer from AS(i)
        NOT marked as correct */
      }
    }
  }
  else{ /* no match with a correct answer */
    print o(r);
    /* NOT marked as correct */
  }
  r++; /* examine next rank */
}
```

Figure 2. Algorithm for marking a system output [11].

```
rmax=max(L,R); /* L: system output size */
               /* R: #answer synsets */

/* obtain cumulative gains for the
IDEAL ranked output */
r=0; cig[0]=0;
for each X in (S,A,B) { /* X: correctness level */
  for( k=1; k<=R(X); k++ ){
  /* R(X): #answer synsets in which the
     highest correctness level is X. */
    r++;
    cig[r]=cig[r-1]+gain(X);
  }
}
for( r=R+1; r<=rmax; r++ ){ /* in case L>R */
  cig[r]=cig[R];
}

/* obtain cumulative bonused gains for
the system output */
r=0; cbg[0]=0;
for( r=1; r<=L; r++ ){
  if( o(r) is marked with X ){
    cbg[r]=cbg[r-1]+gain(X)+1;
  }
  else{
    cbg[r]=cbg[r-1];
  }
}
for( r=L+1; r<=rmax; r++ ){ /* in case L<R */
  cbg[r]=cbg[L];
}

/* calculation */
sum=0;
for( r=1; r<=L; r++ ){
  if( cbg[r]>cbg[r-1] ){
  /* i.e. correct answer at Rank r */
    sum+=cbg[r]/(cig[r]+r);
  }
}
Q-measure=sum/R;
R-measure=cbg[R]/(cig[R]+R);
```

Figure 3. Algorithm for calculating Q-measure/R-measure [11].

is responsible for Row (d). (Note that this classification is based on which module caused the *first* problem: For example, if both answer type recognition and retrieval were unsuccessful, the former is held responsible.) At the time of submission, we were fully aware that our answer type recognition rules and semantic class recognition rules were not as sophisticated as we wished them to be. As this is only our first QAC participation, we plan to do better in the next round of QAC by improving the rules.

Table 4(a) and (b) show the Q-measure values for TSB-A and TSB-B. (We will discuss (c)-(e) in Section 4.2.) Figure 4 shows the value of Q-measure *minus* Reciprocal Rank for each question with TSB-A in order to illustrate the advantages of Q-measure. Thus, the dots above zero imply that Q-measure values are higher than Reciprocal Rank values, and the dots below zero imply the opposite. (Recall that Reciprocal Rank is either $0, 0.2, 0.25, 0.333, 0.5$ or 1 given $L' = 5$.) The six "empty" dots near the top/bottom of Figure 4 represent the six questions discussed below.

The "emtpy" dots that are close to the top in Figure 4 represent QAC2-10001-01, 10001-03, 10017-1 and 10161-1: For all of these questions, the first correct answer was at Rank 3 and therefore Reciprocal Rank was 0.333, while Q-measure was as high as 0.667. This reflects the fact that the answer was S-correct, and that there was only one answer synset ($R = 1$). For example, for 10001-01, ASKMi returned the S-correct answer "Masato Yoshii" (See Table 2) at Rank 3, and therefore the *cumulative gain* sequence is $(cg(1), cg(2), \ldots) = (0, 0, \mathbf{3}, 3, \ldots)$ and the *cumulative bonused gain* sequence is $(cbg(1), cbg(2), \ldots) = (0, 0, \mathbf{4}, 4, \ldots)$ [11]. ($cg(r)$ and $cbg(r)$ values are shown in **bold** whenever $g(r) > 0$.) Whereas, the *cumulative ideal gain* sequence is $(cig(1), cig(2), \ldots) = (3, 3, 3, 3, \ldots)$, as there is only one answer synset and it contains an S-correct answer string. Therefore, $Q\text{-}measure = (\mathbf{4}/(3 + 3))/1 = 0.667$. Now, suppose that ASKMi returned the B-correct answer "Yoshii" instead at Rank 3: In this case, $(cg(1), cg(2), \ldots) = (0, 0, \mathbf{1}, 1, \ldots)$ and $(cbg(1), cbg(2), \ldots) = (0, 0, \mathbf{2}, 2, \ldots)$. Therefore, $Q\text{-}measure = (\mathbf{2}/(3 + 3))/1 = 0.333$. In this way, Q-measure can reflect the differences in answer correctness levels.

Table 3. Failure analysis of TSB-A.

#questions with a correct answer in top 5	111
First correct answer at Rank 1	75
First correct answer at Rank 2	17
First correct answer at Rank 3	7
First correct answer at Rank 4	6
First correct answer at Rank 5	6
# questions without a correct answer in top 5	84
(a) Out of answer type taxonomy	0
(b) Failure to capture the correct answer type	36
(c) Failure to retrieve a supporting document	11
(d) Failure to extract a correct answer from a retrieved supporting document	21
(e) Failure to rank a correct answer within top 5 due to noise in answer type recognition	5
(f) Others	11
(f1) First correct answer within Ranks 6-10	(4)
(f2) Other failures	(7)

We now turn to the "emtpy" dots at the bottom of Figure 4. For QAC2-10124, Reciprocal Rank was 1 as ASKMi returned an S-correct answer at Rank 1, but Q-measure was only 0.143. This is because there are as many as seven answer synsets (See Table 2), each representing a satellite of Jupiter. As none of the other returned answers were correct, $(cbg(1), cbg(2), \ldots) = (\mathbf{4}, 4, 4, \ldots)$. Whereas, $(cig(1), cig(2), \ldots) = (3, 6, 9, \ldots)$ as each answer synset contains an S-correct answer. Therefore, $Q\text{-}measure = (\mathbf{4}/(3+1))/7 = 1/7 = 0.143$. For 10157-01, ASKMi returned an A-correct answer ("Lenin") at Rank 1 and another ("Former President Gorbachev") at Rank 4. Thus, Reciprocal Rank is 1 *even though the question explicitly requests a list of 10 Russian politicians*. In contrast, Q-measure is only 0.150: $(cg(1), cg(2), \ldots) = (\mathbf{2}, 2, 2, \mathbf{4}, 4, \ldots)$ and $(cbg(1), cbg(2), \ldots) = (\mathbf{3}, 3, 3, \mathbf{6}, 6, \ldots)$. Whereas, as all of the correct answer strings for 10157-01 are only A-correct (See Table 2), $(cig(1), cig(2), \ldots) = (2, 4, 6, 8, 10, \ldots)$. Therefore, $Q\text{-}measure = ((\mathbf{3}/(2+1)) + (\mathbf{6}/(8+4)))/10 = 0.150$. It is clear from these examples that Q-measure can properly handle both "single-answer" and "multiple-answer" questions. Thus, the abundance of dots in the lower region of Figure 4 represents the fact that Reciprocal Rank *overestimates* the system's performance when there are several possible answers.

4.2 Upperbound Performance

We have conducted some additional experiments using the QAC2 answer data for estimating the upperbound performance of ASKMi, namely, "oracle answer type" (OAT) and "oracle supporting documents" (OSD) (or *oracle retriever* [14]) experiments, as well as the combination of the two. OAT means that the ASKMi Question Analyzer is provided with one correct answer type for each question (even though there may in fact be more than one possibility), and OSD means that the ASKMi Retriever searches only the supporting documents listed in the QAC2 answer file.

Table 4. Q-measure values for the Official/Oracle runs.

Run Name	MRR	Q-measure
(a) TSB-B	0.440	0.391
(b) TSB-A	0.454	0.396
(c) TSB-A+OAT	0.536	0.463
(d) TSB-A+OSD	0.567	0.513
(e) TSB-A+OAT+OSD	0.678	0.591

Table 4(c), (d) and (e) show the performance of our OAT run, OSD run and the combination of the two, respectively. As the highest official MRR at QAC2 was 0.607, improving a single component of ASKMi (e.g. Question Analyzer or Retriever) is clearly not sufficient for catching up with the top performers. Whereas, as the combination of two "oracles" (TSB-A+OAT+OSD) is very successful, it is probably possible to achieve over 0.6 in MRR by improving all the components, including the Semantic Class Recognizer and the Answer Formulator. In fact, our MRR for the *QAC1* formal question set is currently over 0.7.

5 Conclusions

Through our first participation at the QAC Japanese Question Answering task, we showed that our newly developed QA system, ASKMi, shows respectable performance. By improving each of ASKMi's components, namely, the Semantic Class Recognizer, the Question Analyzer, the Retriever and the Answer Formulator, we hope to catch up with the top performers at QAC soon. As all of the ASKMi components are rule-based, semi-automatic acquisition of "human-readable" rules (as opposed to "black-box" classifiers) will be one of our future research topics. At the same time, we plan to tackle the problem of Encyclopaedic QA by improving the accuracy and coverage of the Relation Extractor.

As a by-product of our experiments, it has been demonstrated that Q-measure is a useful QA evaluation metric. We would like to propose the use of Q-measure as an official evaluation metric at QAC, by introducing answer correctness levels to the QAC test collections where appropriate.

Acknowledgment

We would like to thank Toshihiko Manabe for his valuable suggestions regarding the development of ASKMi.

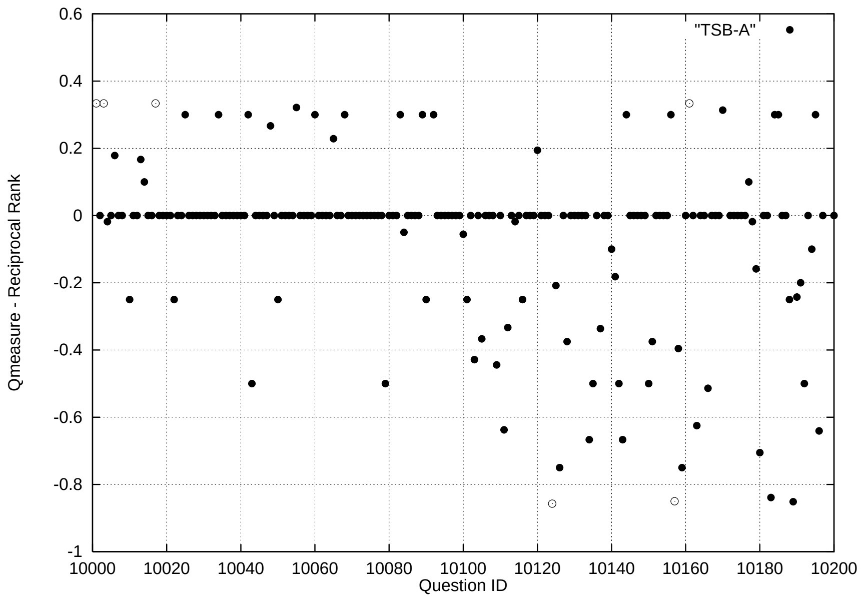

Figure 4. Q-measure minus Reciprocal Rank (TSB-A**).**

References

[1] Diekema, A. R. *et al.*: Question Answering: CNLP at the TREC-2002 Question Answering Track, *TREC-2002 Proceedings*, (2002).

[2] Hermjakob, U.: Parsing and Question Classification for Question Answering, *ACL 2001 Workshop on Open-Domain Question Answering* (2001).

[3] Hovy, E. *et al.*: Using Knowledge to Facilitate Factoid Answer Pinpointing, *COLING 2002 Proceedings* (2002).

[4] Ichimura, Y. *et al.*: A Study of the Relations among Question Answering, Japanese Named Entity Extraction, and Named Entity Taxonomy (in Japanese), *IPSJ SIG Notes*, NL–161–3, to appear (2004).

[5] Prager, J. *et al.*: Question-Answering by Predictive Annotation, *ACM SIGIR 2000 Proceedings*, pp. 184-191 (2000).

[6] Sakai, T. *et al.*: Retrieval of Highly Relevant Documents based on Semantic Role Analysis (*in Japanese*), *FIT 2002 Information Technology Letters*, pp. 67–68 (2002).

[7] Sakai, T. *et al.*: BRIDJE over a Language Barrier: Cross-Language Information Access by Integrating Translation and Retrieval, *IRAL 2003 Proceedings*, pp.65-76 (2003). `http://acl.ldc.upenn.edu/W/W03/W03-1109.pdf`

[8] Sakai, T. and Kokubu, T.: Evaluating Retrieval Performance for Japanese Question Answering: What Are Best Passages? *ACM SIGIR 2003 Proceedings*, pp. 429-430 (2003).

[9] Sakai, T. *et al.*: ASKMi: A Japanese Question Answering System based on Semantic Role Analysis, *RIAO 2004 Proceedings*, pp. 215-231 (2004).

[10] Sakai, T. *et al.*: Toshiba BRIDJE at NTCIR-4 CLIR: Monolingual/Bilingual IR and Flexible Feedback, *NTCIR-4 Proceedings*, to appear (2004).

[11] Sakai, T: New Performance Metrics based on Multigrade Relevance: Their Application to Question Answering, *NTCIR-4 Proceedings*, to appear (2004).

[12] Sakai, T. *et al.*: High-Precision Search via Question Abstraction for Japanese Question Answering, *IPSJ SIG Notes*, to appear (2004).

[13] Sparck Jones, K., Walker, S. and Robertson, S. E.: A Probabilistic Model of Information Retrieval: Development and Comparative Experiments, *Information Processing and Management* 36, pp. 779-808 (Part I) and pp. 809-840 (Part II), (2000).

[14] Tellex, S. *et al.*: Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering, *ACM SIGIR 2003 Proceedings*, pp. 41-47 (2003).

[15] Voorhees, E. M.: Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness, *ACM SIGIR '98 Proceedings*, pp. 315-323 (1998).

[16] Voorhees, E. M.: Building A Question Answering Test Collection, *ACM SIGIR 2000 Proceedings*, pp. 200-207, (2000).

[17] Voorhees, E. M.: Overview of the TREC 2002 Question Answering Track, *TREC 2002 Proceedings*, (2002).