Answering Questions Requiring Cross-passage Evidence

Kisuh Ahn
Dept. of Linguistics
Hankuk University of Foreign Studies
San 89 Wangsan-li, Mohyeon-myeon
Yongin-si, Gyeongi-do, Korea
kisuhahn@gmail.com

Hee-Rahk Chae
Dept. of Linguistics
Hankuk University of Foreign Studies
San 89 Wangsan-li, Mohyeon-myeon
Yongin-si, Gyeongi-do, Korea
hrchae@hufs.ac.kr

Abstract

This paper presents methods for answering, what we call, Cross-passage Evidence Questions. These questions require multiply scattered passages all bearing different and partial evidence for the answers. This poses special challenges to the textual QA systems that employ information retrieval in the "conventional" way because the ensuing Answer Extraction operation assumes that one of the passages retrieved would, by itself, contain sufficient evidence to recognize and extract the answer. One method that may overcome this problem is factoring a Cross-passage Evidence Question into constituent sub-questions and joining the respective answers. The first goal of this paper is to develop and put this method into test to see how indeed effective this method could be. Then, we introduce another method, Direct Answer Retrieval, which rely on extensive pre-processing to collect different evidence for a possible answer off-line. We conclude that the latter method is superior both in the correctness of the answers and the overall efficiency in dealing with Cross-passage Evidence Questions.

1 Distinguishing Questions Based on Evidence Locality

Textual factoid Question Answering depends on the existence of at least one passage or text span in the corpus that can serve as sufficient evidence for the question. A single piece of evidence may suffice to answer a question, or more than a single piece may be needed. By "a piece of evidence", we mean a snippet of continuous text, or passage, that supports or justifies an answer to the question posed. More practically, in factoid QA, a piece of evidence is a text span with two properties: (1) An Information Retrieval (IR) procedure can recognise it as relevant to the question and (2) an automated Answer Extraction (AE) procedure can extract from it an answer-bearing expression (aka an *answer candidate*).

With respect to a given corpus, we call questions with the following property *Single Passage Evidence Questions* or SEQs:

> A question Q is a SEQ if evidence E sufficient to select A as an answer to Q can be found in the same text snippet as A.

In contrast, we call a question that requires multiple different pieces of evidence (in multiple text spans with respect to a corpus) a *Cross-passage Evidence Question* or CEQ:

> A question Q is a CEQ if the set of evidence $E1$, ..., En needed to justify A as an answer to Q cannot be found in a single text snippet containing A, but only in a set of such snippets.

For example, consider the following question:

> Which Sub-Saharan country had hosted the World Cup?

If the evidence for the country being located south of Sahara dessert and the evidence for this same country having hosted the World Cup is not contained in the same passage/sentence, but are found

26th Pacific Asia Conference on Language,Information and Computation pages 371–380

in two distinct passages, the question would be a Cross-passage Evidence Question. This distinction between SEQs and CEQs lies only in the locality of evidence within a corpus. It does not imply that the corpus contains only one piece of text sufficient for a SEQ: Often there are multiple text snippets, each with sufficient evidence for the answer. Such redundancy is exploited by many question answering systems to rank the confidence of an answer candidate (e.g. including (Brill et al., 2002)) but the evidence is redundant rather than qualitatively different.

Now, as opposed to Single-passage Evidence Questions, which had been the usual TREC type questions (White and Sutcliffe, 2004), Cross-passage Evidence Question poses special challenges to the textual QA systems that employ information retrieval in the "conventional" way. Most textual QA system uses Information Retrieval as document/passage pre-fetch. The ensuing Answer Extraction operation assumes that one of the passages retrieved would, by itself, contain sufficient evidence to recognize and extract the answer. Thus the reliance on a particular passage to answer a question renders the task of question answering essentially a *local* operation with respect to the corpus as a whole. Whatever else is expressed in the corpus about an entity being questioned is ignored, or used (in the case of repeated evidence instances of the same answer candidate) only to increase confidence in particular answer candidates. This means that *factoid questions whose correct answer depends jointly on textual evidence located in different places in the corpus cannot be answered.* We call this *the locality constraint* of factoid QA. Thus special methods are needed to overcome this locality constraint in order to successfully handle CEQs. In the following sections, we explore two methods for answering CEQs, first, based on Question Factoring for conventional IR based QA systems, and second, based on what we call Direct Answer Retrieval method in place of conventional IR.

2 Solving CEQs by Question Factoring

While whether a question is a CEQ or not depends entirely on the corpus, it can be guessed that the more syntactically complex a question, the more likely that it is a CEQ, given that a complex question will have more terms and relations that need to be satisfied. For example, the above question is of the form "What/Which <NBAR> <VP>?" such as *Which* [NBAR *Sub-Saharan country*] [VP *had hosted the World Cup?*], and has at least two predicates/constraints that must be established, the one or more conveyed by the NBAR, and the one or more conveyed by the VP. These multiple restrictions might call for different pieces of evidence depending on the particular corpus from which the answer is to be found.

In database QA, CEQs correspond to queries that involve joins (usually along with selection and projection operations). The database equivalent of the afore-mentioned question about the certain World Cup hosting country might involve joining one relation linking country names with the requisite location, and another linking the names of countries with World Cup hosting history. Note that this involves breaking up the original query into a set of sub-queries, each answerable through a single relation. Answering a CEQ through sub-queries therefore involve the fusion of answers to different questions.

Analogously, we apply this method of joining sub-queries for database to the task of textual QA to deal with the CEQs. The solution we explore here can be adopted by any existing system with the conventional Information (Passage) Retrieval and Answer Extraction (IR+AE) pipeline architecture. It involves:

1. Dividing a CEQ into sub-questions SQ_1, ..., SQ_n, each of which is a simple question about the same question variable.[1]
2. Finding the answer candidate(s) for each sub-question,
3. Selecting the best overall answer from the answer candidates for the sub-questions.

2.1 Dividing a CEQ into sub-questions

Decomposing a CEQ into simpler sub-questions about the question variable involves:

[1] We have only considered sub-questions that are *naturally joined* on the question variable. Extending to the equivalent of joins on several variables would require an even more complex strategy for handling the answer candidate sets than the one we describe in Section 2.2.

- Resolving all co-referring expressions within the question;
- If the WH-phrase of the question is a complex phrase, making a sub-question asking the identity of its head noun phrase (e.g. "Which northern country is ..." → "What is a northern country?");
- Breaking up the question at clause boundaries (including relative clauses);
- Within a single clause, if there is a conjoined set of restrictors (e.g. " ... German physician, theologian, missionary, musician and philosopher .."), copying the clause as many times as the number of restrictors, so that each clause now contains only one restrictor;
- Finally, reformulating all the individual clauses into questions.

Some examples of CEQs which have been factored into sub-questions are as follows[2].:

- Which French-American prolific writer was a prisoner and survivor of the infamous Auschwitz German concentration camp, Chairman of the U.S. President's Commission on the Holocaust, a powerful advocate for human rights and a recipient of the Nobel Peace Prize?
    1. Who was a French-American prolific writer?
    2. Who was a prisoner and survivor of the infamous Auschwitz German concentration camp?
    3. Who was a Chairman of the U.S. President's Commission on the Holocaust?
    4. Who was a powerful advocate for human rights?
    5. Who was a recipient of the Nobel Peace Prize?

It should be clear that factoring a CEQ into a set of sub-questions is often tricky but doable. The intra-sentential anaphora resolution can be less than straight-forward. So often, it is a matter of choice as to how to break a question into how many sub-questions and at what boundaries. In the evaluation reported in Section 2.3, the CEQs were manually broken into sub-questions, based on the procedure outlined above, since our focus was on evaluating the viability of the overall method rather than individual components. Our results may thus serve as an upper-bound to what a fully automated procedure would produce. (For work related to the automatic decomposition of a complex question in order to deal with temporal questions, see (Saquete et al., 2004).)

2.2 Selecting an Answer to the CEQ from Sub-question Answer Candidates

From the answer candidates to the sub-questions, an answer must be derived for the CEQ as a whole. The most obvious way would be to pick the answer candidate that is the answer to the largest number of sub-questions. So if a CEQ had three sub-questions, in the ideal case there would be one answer candidate that would be the answer to all of these sub-questions at the same time and thus be the most likely answer to the CEQ. This is like a simple voting scheme, where each sub-question votes for an answer candidate, and the candidate with the most votes win.

Complications from this ideal situation arise in two ways: First, a sub-question can be very general because it results from separating away other restrictions from the question. Hence, the answer to a sub-question must be regarded instead as a set of answers as in list questions, several of which may be correct rather than being one correct answer to that subquestion. In other words, a sub-question can vote for multiple candidates rather than just one.

Second, simply maximising overlap (i.e. the number of sub-questions to which an answer candidate is the answer, which we call simply *votes* from now on) ignores the possibility that multiple answers can tie for the same maximum votes, especially if this number is small compared to the number of sub-questions. Thus, there is the need for a method to rank the answers with the same number of votes.

In summary, a sub-question can vote for multiple candidates and more than one candidates can receive the same largest number of votes from the sub-questions. This means we need an additional way to

[2]All the evaluation questions are from a pub-quiz site http://www.funtrivia.com

break the ties among the candidates by ranking the candidates according to some other criteria. The answer selection method we have chosen is described below.

Let's assume that a question has been factored into N sub-questions, and each sub-question is answered with a (possibly empty) set of answer candidates. So for the set of N sub-questions for the original question, there are N answer candidate sets. The most likely answer would be the answer candidate shared by all the N sub-questions (i.e. the answer candidate present in all the N sets of answer candidates.). To see if there is such a common answer candidate, these N sets of answer candidates are first intersected (via generalized intersection).

If the intersection of these N sets is empty (i.e., there is no one answer candidate that all the sub-questions share.), then it must be investigated whether there is a common answer candidate for N-1 sets of answer candidates (i.e. an answer shared by N-1 sub-questions.). There will be N cases to consider since there will be N cases of N-1 sub-question sets to intersect. If all these are empty, then all subsets of size N-2 are considered, and so on, until a non-empty intersection is obtained. This means considering the *power set* of the original set of answer candidate sets.

This process may result in one answer candidate or several with the same maximum number of votes. If there is only one, this is chosen as the answer. Otherwise, there is a need for a further way to rank these answer candidates to produce the most likely as the answer. Specifically, if there is more than one non-empty intersection with the same maximum votes, we do the expected thing of taking into account the original ranking of answer to the sub-questions (in most QA systems, the answer candidates are ranked according to their plausibility). If an answer is found to be the second ranked answer to one sub-question and the sixth ranked answer to another, its overall rank is taken to be the simple mean of the ranks. So in this case, the answer would have the rank four overall. This algorithm can be more formally stated as follows[3].

- Step 1: Let S be the set of the sets of answers, $A1$, .. , An, to the sub-questions, $Q1$ .. Qn respectively.
- Step 2: Produce the power-set of S, i.e. $P = POW(S)$.
- Step 3: Produce a set of ordered pairs, V, such that $V = \{\langle o, R\rangle \mid R \subset P \wedge \forall x \in R.\ |x| = o$ for every distinct $o = |y|$ of every $y \in P\}$
- Step 4: Pick the ordered pair, L such that $L = \langle o, R\rangle \in V$ with the largest o among the members of V, and do:
  1. Produce T from R such that $T = \bigcup\{x \mid x = \bigcap t$ for every $t \in R\}$. (Note that t is a set.)
  2. If R' is an empty set, repeat this step 4 for the ordered pair with the next largest o.
  3. Else if T has a unique member, then pick that unique member as the answer and terminate.
  4. Otherwise, go to the next step.
- Step 5: For each member, $x \in T$, get the ranks, r_1, .. , r_n in the sub-questions, Q_1 .. Q_n and compute the mean M of the ranks.
- Step 6: Pick the member of T with the lowest M score as the answer.

To illustrate the steps described above, consider a CEQ M that can be split into three sub-questions Q1, Q2 and Q3. Then:

- Step 1: Assume that the sets A1, A2, A3 are the answer candidate sets for sub-questions Q1, Q2 and Q3 respectively:

  A1 = {CLINTON, BUSH, REAGAN}
  A2 = {MAJOR, REAGAN, THATCHER}
  A3 = {FORD, THATCHER, NIXON}

  Let S = {A1,A2,A3}
- Step 2: P = POW(S) = {{A1,A2,A3},{A1,A2},{A2,A3},{A1,A3},{A1},{A2},{A3},{}}
- Step 3:

  V = $\{\langle 3, \{\{A1, A2, A3\}\}\rangle$,
  $\langle 2, \{\{A1, A2\}, \{A2, A3\}, \{A1, A3\}\}\rangle$,
  $\langle 1, \{\{A1\}, \{A2\}, \{A3\}\}\rangle, \langle 0, \{\{\}\}\rangle\}$

[3]This will be easier to understand if considered together with the example that follows.

- Step 4:

  First pick $L = \langle o, R\rangle = \langle 3, \{\{A1, A2, A3\}\}\rangle$ based on the largest o in consideration (i.e. 3).

  Then, get T such that $T = \bigcup\{x \mid x = \bigcap t$ for every $t \in R\} = \bigcup\{A1 \cap A2 \cap A3 = \{\}\}$

  Since T is an empty set, no answer can be picked. So repeat this step for the ordered pair with the next largest votes.

- So again Step 4:

  The second pick $L = \langle o, R\rangle = \langle 2, \{\{A1, A2\}, \{A2, A3\}, \{A1, A3\}\}\rangle$

  Now get T by $T = (A1 \cap A2) \cup (A2 \cap A3) \cup (A1 \cap A3) = \{\text{REAGAN}, \text{THATCHER}\}$

  Since R' is non-empty, which at the same time does not have a unique member, go to the next step.

- Step 5:

  REAGAN: 4th candidate for Q1 and 10th in Q2 – average rank 7

  THATCHER: 1st candidate for Q2 and 5th in Q3 – average rank 3

- Step 6:

  Take the candidate with the highest average rank as the answer – here, THATCHER.

2.3 Evaluation of the Sub-question Method

The evaluation has two purposes. The first is to see how well the sub-question method works with respect to CEQs. The second is to provide a baseline results for the performance of the Direct Answer Retrieval method introduced in the next section.

For the evaluation of this strategy, we chose 41 "quiz" questions (i.e. ones designed to challenge and amuse people rather than to meet some real information need, like the IBM Watson system doing Jeopardy questions (David Ferrucci, 2012)) as likely Cross-passage Evidence Questions (CEQs) with respect to the knowledge base corpus, AQUAINT corpus in this evaluation. They were filtered based on the following criteria:

- Did the question contain multiple restrictions of the entity in question?
- Did the answer appear in the AQUAINT corpus?
- Was the answer a proper name?
- Was it a difficult question? (Questions from the original site have been manually marked for difficulty.)

The questions, varying in length from 20 to 80 words[4], include:

> *What was the name of the German physician, theologian, missionary, musician and philosopher who was awarded the Nobel Peace Prize in 1952?*
>
> *Which French-American prolific writer was a prisoner and survivor of the infamous Auschwitz German concentration camp, Chairman of the U.S. President's Commission on the Holocaust, a powerful advocate for human rights and a recipient of the Nobel Peace Prize?*
>
> *He was born in 1950 in California. He dropped out of the University of California at Berkeley and quit his job with Hewlett-Packard to co-found a company. He once built a "Blue Box" phone attachment that allowed him to make long-distance phone calls for free. Who is he?*

The QA system we have used to test this method was developed previously and had shown good performance for TREQ QA questions (Ahn et al., 2005). In order to accomodate this method, a question factoring module must be added. But as we have previously mentioned, question factoring has been done manually offline as we have not implemented a fully automatic module for that as of yet. The joining of answers to sub-questions are done automatically, however, using a specially added post-processing module. Thus, this QA system is a simulation of a fully-automatic CEQ handling QA system.

The evaluation procedure ran as follows. First, a CEQ was factored into a set of sub-questions manually. Then each sub-question was fed into the QA

[4]This is long compared to TREC questions, whose mean length is less than 10 words.

QID	SQ	MaxV	ACI	Rank
1	2	1	0	0
3	4	1	0	0
6	3	1	0	0
7	4	1	0	0
8	3	1	0	0
9	5	2	6	3
10	4	1	0	0
12	3	1	0	0
13	2	2	2	1
16	3	1	0	0
17	2	2	1	1
18	4	1	0	0
20	2	2	4	1
21	3	2	7	1
22	2	1	0	0
23	3	3	1	1
24	2	1	0	0
25	3	3	1	1
26	5	3	1	1
27	2	2	23	1
28	2	1	0	0
29	2	2	86	1
30	3	3	3	1
31	2	2	9	5
32	2	2	1	1
36	9	1	0	0
37	2	1	0	0
38	8	1	0	0
40	9	3	1	1

Table 1: Questions with answer candidates identified for $\geq$1 sub-questions.

engine to produce a set of 100 answer candidates. The resulting sets of answers were assessed by an Answer Selection/Ranking module that uses the algorithm described in Section 2.2 to produce a final set of ranked answer candidates.

2.4 Results and Observations

Among the 41 questions, 12 questions are found to have no correct answer candidates by the QA engine, and so these questions are ignored in the rest of the analysis. For the remaining 29 questions, Table 1 shows the value of ranking (i.e., Step 5 above) when more than one answer candidate shares the same number of largest votes. Such answer candidates need to be ranked in order to pick the best answer. Here, **QID** indicates the question ID which did have at least one sub-question with answers, where **SQ** more specifically tells how many sub-questions each question was factored into.

A@N	**SQ-Combined**
1	0.317:13
2	0.317:13
3	0.341:14
4	0.366:15
5	0.366:15
6	0.366:15
7	0.366:15
8	0.366:15
9	0.366:15
10	0.366:15
15	0.366:15
20	0.366:15
ACC	0.317
MRR	0.341
ARC	1.333

Table 2: Results for Sub-question Method

In Table 1, column **MaxV** indicates the largest number of votes received by an answer candidate across the set of sub-questions. Column **ACI** indicates the number of answer candidates with this number of Votes. For example, CEQ 9 has 5 sub-questions. Its **MaxV** of 2 indicates that only the intersection of the answer candidate sets for two sub-questions (out of maximum 5) produced a non-empty set (of 6 members according to **ACI**). These 6 members are the final answer candidates that will be ranked.

The column labelled **Rank** indicates the ranking of final answer candidates by mean ranking with respect to the sub-questions they answer and identifies where the correct answer lies within that ranking. So for Question 9, the correct answer was third in the final ranking. Fifteen questions had no answer candidates common to any set of sub-questions (i.e., **ACI=**0). Of the 14 remaining questions, six had only a single answer candidate, so ranking was not relevant. (That answer was correct in all 6 cases.) Of the final eight questions with $\geq$1 final answer candi-

dates, Table 1 shows that ranking them according to the mean of their original rankings led to the correct answer being ranked first in six of them. In the most extreme case, starting with 86 ties between answers for two sub-questions (all of which have a set of 100 answers), ranking reduces this to the correct answer being ranked first. Table 1 also shows the importance of the proportion of sub-questions that contain the correct overall answer. For CEQs with 2 sub-questions, in every case, both sub-questions needed to have contained the correct overall answer in order for that CEQ to have been answered correctly. (If only one sub-question contains the correct overall answer, our algorithm has not selected the correct overall answer.) For CEQs with 3 sub-questions, at least two of the 3 sub-questions need to have contained the correct overall answers in order for the CEQ to be answered correctly by this strategy. This seems to be less the case as the number of sub-questions increases. However, the strategy does seem to require at least two sub-questions to agree on a correct overall answer in order for a CEQ to be correctly answered. It is possible that another sub-question ranking strategy could do better.

In sum, starting with 41 questions, the "sub-question method found 14 with at least one "intersective" answer candidate. Of those 14, the top-ranked candidate (or, in 6 cases, the only "intersective" candidate) was the correct answer in 12 cases. Hence the "sub-question" method has succeeded in 12 of 41 cases that would be totally lost without this method. Table 2 shows the final scores with respect to A@N, accuracy, MRR and ARC.

2.5 Discussion

The evaluation shows that the method based on sub-question decomposition and ranking can be used for answering Cross-passage Evidence Questions, but we also need to consider the cost of adopting this method as a practical method for doing real time Question Answering; clearly multiplying the number of questions to be answered by decomposing a question into sub-questions can make the overall task even more resource-intensive than it is already. Pre-caching answers to simple, frequent questions, as in (Chu-Carroll et al., 2002), which reduces them to database look-up, may help in some cases. Another compatible strategy would be to weigh the sub-questions, so as to place more emphasis on more *important* ones or to first consider ones that can be answered more quickly (as in database *query optimisation*). This would avoid, or at least postpone, very general sub-questions such as "Who was born in 1950?", which are like list questions, with a large number of candidate answers and thus expensive to process. The QA system would well process more specific sub-questions first by learning the appropriate weight as in (Chali and Joty, 2008) The second issue is for the need of a method that can reliably factor a CEQ into simple sub-questions automatically, which would not be trivial to develop. Direct Answer Retrieval for QA offers another alternative that does not require the factoring of a question in the first place, which is the subject of the next section.

3 Direct Answer Retrieval for QA

The answers to many factoid questions are named entities. For example, "Who is the president of France?" has as its answer a name referring to a certain individual. The basic idea of Direct Answer Retrieval for Question Answering is to extract these kinds of expressions off-line from a textual corpus as potential answers and process them in such a way that they can be directly retrieved as answers to questions. Thus, the primary goal of Direct Answer Retrieval is to turn factoid Question Answering into fine-grained Information Retrieval, where answer candidates are directly retrieved instead of documents/passages. For simple named entity answers, we have previously shown that this can make for fast and accurate retrieval (Ahn and Webber, 2007).

In addition, as the process gathers and collates all the relevant textual evidence for a possible answer from all over the corpus, it becomes possible to answer a question based on *all* the evidence available in the corpus regardless of its locality. Whether this is indeed so is what we are going to put to test here.

3.1 The Off-line Processing

The supporting information for a potential answer (named-entity), for the present research, is the set of all text snippets (sentences) that mentions it. With the set of all sentences that mention a particular potential answer put into one file, this can literally be

regarded as a document on its own with the answer name as its title. The collection of such documents could be regarded as the index of answers and the retrieval of documents as answer retrieval (Hence the name *Direct Answer Retrieval*).

The processes of generating the collection of answer documents for all the potential answers run as follows. First, the whole base corpus is run through POS tagging, chunking and named-entity recognition. Then, each sentence in each document is examined as to whether it contains at least one named-entity. If so, then whether this named-entity represents an already identified answer candidate and stored in the repository is examined. If so, then this sentence is appended to the corresponding answer document in the collection. If not, then a new answer entity is instantiated and added to the repository and a new corresponding answer document is created with this sentence. If more than answer candidate is identified in a sentence, then the same process is applied to every one of them.

In order to facilitate the retrieval of answer documents, this answer document collection is itself indexed using standard document indexing technique. At the same time, for each answer entity, its corresponding named entity type is stored in a separate answer repository database(using external resources such as YAGO (Suchanek et al., 2007), it is possible to look up very fine-grained entity type information, which we did in our evaluation system). Answer Index together with this repository database make up the knowledge base for Direct Answer Retrieval QA system.

3.2 The On-line Processing

With the knowledge base built off-line, the actual on-line QA processes run through several steps. The first operation is the answer type identification. For this, in our evaluation system, the question is parsed and a simple rule based algorithm is used that looks at the WH-word (e.g. "Where" means location), the head noun of a WH-phrase with "Which" or "What" (e.g. "Which president" means the answer type is of president), and if the main verb is a copula, the head of the post-copula noun phrase (e.g. for "Who is the president ..", here again "president" is the answer type. The identified answer type is resolved to one the base named entity type (PERSON, LOCATION, ORGANIZATION and OTHER) using ontology database such as the WordNet if the named entity type is derived from the head noun of WH-word or the noun after the copula, which do not have the corresponding entity type in the answer repository. The next operation is the retrieval of answers as answer candidates for a given question. This involves formulating a query, retrieving answer documents as answer candidates. If the index has been partitioned into sub-indices based on the NER type, as we have done, an appropriate index can be chosen based on the answer type identified, and thereby make the search more effiecient.

In the actual retrieval operation, there can be different ways to score an answer candidate with respect to a query depending on the model of retrieval used. The model of retrieval implemented for the evaluation system is an adaptation of the document inference network model for information retrieval (Turtle, 1991). For a more thorough description of the answer retrieval model, please refer to our previous work (Ahn and Webber, 2008).

When the search is performed and a ranked list of answers is retrieved, this ranked list is then run through the following procedures:

1. Filter the retrieved list of answers to remove any named-entity that was mentioned in the question itself if any.

2. Re-rank with respect to answer type, preferring the answer that match the answer type precisely.

3. Pick the highest ranking answer as the answer to the question.

The first procedure is heuristically necessary because, for example, "Germany" in "Which country has a common border with Germany", can pop up as an answer candidate from the retrieval. From the remaining items in the list, each item is looked up with respect to its answer type using the answer-type table in the answer repository. The re-ranking is performed according to the following rules:

- Answers whose type precisely matches the answer type are ranked higher than any other answers whose types do not precisely match the answer type.

- Answers whose type do not precisely match the answer type but still matches the base type traced from the answer type are ranked higher than any other answers whose types do not match the answer type at all.

Now using these rules, the answer which is ranked the highest is picked as the number one answer candidate.

This method, by itself, looks more like an answer candidate retrieval system rather than a full-fledged QA system with sophisticated answer extraction algorithm as found in most QA systems. However, the structured retrieval algorithm that we employ makes up for this answer extraction operation. Again for a full exposition of this structural retrieval operation, refer to our previous work.

3.3 Answering CEQs by Direct Answer Retrieval Method

Direct Answer Retrieval enables a direct approach to Cross-passage Evidence Questions: There is no need for any special method for question factoring. With respect to an answer document, a possible answer is already associated with all the distributed pieces of evidence about it in the corpus: In other words, CEQs are exactly the same as SEQs.

Here, for example, to the question, "Which US senator is a former astronaut?" the conventional approach requires different set of textual evidence (passages) to be assembled based on the decomposition of the question as we have presented in the previous section, whereas in the Direct Answer Retrieval approach, only one answer document needs to be located.

Thus there are three advantages for using Direct Answer Retrieval method over the conventional IR with the special sub-question method:

- No multiplication of questions.
- No need for question factoring.
- No need to combine the answers of the sub-questions.

Whether, despite these advantages, the performance would hold good, is evaluated next.

3.4 Evaluation and Comparison to IR+AE

The purpose of the evaluation is to see how well this method can deal with CEQs, particularly as compared to simulated IR+AE system with a sub-question method as presented in the previous section.

The implemented QA system had been previously evaluated with respect to the more simple TREC QA questions and found to have good performance. The particular configuration that we used for our test here utilizes naturally the same questions and the same textual corpus as the source data as in the sub-question method.

For each question, the system simply retrieves answer candidates and the top 20 answers are taken for the score assessment.

A@N	Sub-QA	Answering
1	0.317:13	0.317:13
2	0.317:13	0.512:21
3	0.341:14	0.585:24
4	0.366:15	0.634:26
5	0.366:15	0.659:27
6	0.366:15	0.659:27
7	0.366:15	0.659:27
8	0.366:15	0.659:27
9	0.366:15	0.659:27
10	0.366:15	0.659:27
15	0.366:15	0.659:27
20	0.366:15	0.659:27
ACC	0.317	0.317
MRR	0.341	0.456
ARC	1.333	1.889

Table 3: Comparison of the Scores

Table 3 summarises the results of the evaluation for the Direct Answer Retrieval system (**Ans-Ret**) compared to the Sub-question method based system (**Sub-QA**).

The Ans-Ret system has returned more correct answers than the **Sub-QA** system (in all cut-off points except having tied in number 1 rank). Also all the correct answers that were found by the **Sub-QA** system were also found by the **Ans-Ret** system irrespective of the ranks. Twelve correct answers that were found by **Ans-Ret** (at **A@5**) were missed by the **Sub-QA** system. Among those answers that

were found by both systems, **Sub-QA** produced better rank in 5 cases whereas in only one case the **Ans-Ret** produced a better rank. However, Table 3 shows that **Ans-Ret** in general found more correct answers (27 vs 15) in total, and more answers in higher rank (e.g. top 2) than **Sub-QA** system.

In order to verify the performance difference, we used a statistical test, *Wilcoxon Matched Signed Rank Test* (Wilcoxon, 1945), which is used for small samples and assumes no underlying distribution. According to this test, the difference is statistically significant (W+ = 17.50, w- = 172.50, N = 19, p = 0.0007896). Hence, it is possible to conclude that Direct Answer Retrieval method is superior to the simulated IR+AE method with special sub-question method for this type of questions. The fact that Direct Answer Retrieval Method has superior performance is no surprise considering that the more the evidence from the question for an answer, more information is available for the method in matching the relevant answer document of a candidate answer to the question. This is in contrast to the IR+AE based systems, which, without a special strategy such as the sub-question method discussed here, require that whatever evidence exist in a question must be found within one sentence in the corpus due to the locality constraint.

4 Conclusion

Cross-passage Evidence Questions are the kind of questions for which the locality constraint bear out its constraining effect. A special method is devised in order to overcome this constraint with the conventional QA with IR+AE architecture. This involves partitioning a question into a set of simpler questions. The results show that the strategy is successful to a degree in that some questions are indeed correctly answered but it also comes with a cost of multiplying the number of questions. On the other hand, Direct Answer Retrieval method is process-wise more efficient, has a better performance in terms of accuracy, and does not require tricky question factoring regarding this type of questions. Direct Answer Retrieval method, thus, clearly shows a clear advantage for CEQs.

References

Ahn, K., Bos, J., Clark, S., Curran, J., Kor, D., Nissim, M., and Webber, B. (2005). Question answering with qed at trec-2005. In *Proceedings of TREC'05*.

Ahn, K. and Webber, B. (2007). Nexus: A real time qa system. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference*.

Ahn, K. and Webber, B. (2008). Topic indexing and retrieval for qa. In *COLING '08: Proceedings of the Coling 2008 IR4QA Workshop*.

Brill, E., Dumais, S., and Banko, M. (2002). Analysis of the askmsr question-answering system. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, pages 257–264, Philadelphia PA.

Chali, Y. and Joty, S. R. (2008). Selecting sentences for answering complex questions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 304–313.

Chu-Carroll, J., Prager, J., Welty, C., Czuba, K., and Ferrucci, D. (2002). A multi-strategy and multi-source approach to question answering. In *Proceedings of the 11^{th} Text Retrieval Conference (TREC 10)*, National Institute of Standards and Technology.

David Ferrucci, Eric Brown, e. a. (2012). Building watson: An overview of the deepqa project. *AI Magazine, Volume 31, No 3*, 31(1).

Saquete, E., Martínez-Barco, P., Muñoz, R., and González, J. L. V. (2004). Splitting complex temporal questions for question answering systems. In *ACL*, pages 566–573.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In Williamson, C. L., Zurko, M. E., and Patel-Schneider, Peter F. Shenoy, P. J., editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada. ACM.

Turtle, H. R. (1991). *Inference Networks for Document Retrieval*. PhD thesis, University of Massachusetts.

White, K. and Sutcliffe, R. F. E. (2004). Seeking an upper bound to sentence level retrieval in question answering. In *Proceedings of the 23rd Annual International ACM SIGIR Workshop on Question Answering (SIGIR 2004)*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, (1):80–83.