A Hybrid Question Answering System based on Information Retrieval and Answer Validation

Partha Pakray[1], Pinaki Bhaskar[1], Somnath Banerjee[1], Bidhan Chandra Pal[1], Sivaji Bandyopadhyay[1], Alexander Gelbukh[2]
Department of Computer Science and Engineering[1],
Jadavpur University, Kolkata – 700032, India[1]
Center for Computing Research[2],
National Polytechnic Institute, Mexico City, Mexico[2]
{parthapakray, pinaki.bhaskar, s.banerjee1980, bidhan.cstbesus}@gmail.com[1],
sivaji_cse_ju@yahoo.com[1], gelbukh@gelbukh.com[2]

Abstract. The article presents the experiments carried out as part of the participation in the main task of QA4MRE@CLEF 2011. We have submitted total five unique runs in the main task: two runs from systems based on Answer Validation (AV) machine reading techniques, one run from systems based on Question Answering (QA) techniques while the last two runs are hybrid systems where the decision is taken based on the outputs from the AV and QA based systems. In the AV system, we first combine the question and each answer option to form the Hypothesis (H). Stop words are removed from each H and query words are identified to retrieve the most relevant sentences from the associated document using Lucene. Relevant sentences are retrieved from the associated document based on the TF-IDF of the matching query words along with n-gram overlap of the sentence with the H. Each retrieved sentence defines the Text T. Each T-H pair is assigned a ranking score in the AV system that works on textual entailment principle. The answer option for which the T-H pair gets the maximum score is selected as the possible answer. The two unique runs differ in the way in which the relevant sentences are retrieved from the associated document. The second system is based on Question Answering (QA) technique. Each question along with each answer option generates the possible answer patterns. Each sentence in the associated document is assigned an inference score with respect to each answer pattern. The sentence that receives the highest inference score corresponding to the answer patterns is identified as the relevant sentence in the document and the corresponding answer option is selected as the answer to the given question.

Keywords: QA4MRE Data Sets, Named Entity, Textual Entailment, Question Answering technique.

1 Introduction

After the success of ResPubliQA 2009 [1] and ResPubliQA 2010 [2], the third evaluation campaign on Question Answering system is Question Answering for Machine Reading Evaluation (QA4MRE)[1] at CLEF 2011.

The main objective of QA4MRE [3] is to develop a methodology for evaluating Machine Reading systems through Question Answering and Reading Comprehension Tests. Machine Reading task obtains an in-depth understanding of just one or a small number of texts. The task focuses on the reading of single documents and identification of the correct answer to a question from a set of possible answer options. The identification of the correct answer requires various kinds of inference and the consideration of previously acquired background knowledge. Ad-hoc collections of background knowledge have been provided for each of the topics in all the languages involved in the exercise so that all participating systems work on the same background knowledge. Texts have been included from a diverse range of sources, e.g. newspapers, newswire, web, blogs, Wikipedia entries.

We have submitted total five unique runs in the main task: two runs from systems based on Answer Validation (AV) techniques, another one run from systems based on Question Answering (QA) techniques while the last two runs are hybrid system where the decision is taken based on the outputs from the AV and QA based systems.

Answer Validation Exercise (AVE) is a task introduced in the QA@CLEF competition. AVE task is aimed at developing systems that decide whether the answer of a Question Answering system is correct or not. There were three AVE competitions AVE 2006 [4], AVE 2007 [5] and AVE 2008 [6]. AVE systems receive a set of triplets (Question, Answer and Supporting Text) and return a judgment of "SELECTED", "VALIDATED" or "REJECTED" for each triplet. We have participated in the Paragraph Selection (PS) Task and Answer Selection (AS) Task [7] in QA@CLEF 2010 – ResPubliQA.

Section 2 describes the corpus statistics. Section 3 describes the Answer Validation based Machine Reading System Architecture. Section 4 details the Question Answering based Machine Reading System Architecture. Section 5 details the Hybrid Machine Reading System Architecture. The experiments carried out on test data sets are discussed in Section 6 along with the results. The conclusions are drawn in Section 7.

2 Corpus Statistics

The data set is made up of a series of tests. Each test consists of one single document (Test Document) with several questions and a set of choices per question. So, the task is a Reading Comprehension test of the given document. Each participating system is provided with:

- 6 Test Documents (2 documents for each of the three topics)
- 10 questions per document with 5 choices for each question.

[1] http://celct.fbk.eu/ResPubliQA/

Topics, documents and questions were made available in English, German, Italian, Romanian, and Spanish. We worked only with English language data. The Background Collections (one for each topic) are comparable (but not identical) topic-related collections created in all the different languages.

3 Answer Validation based Machine Reading System Architecture

The architecture of the Answer Validation (AV) [8] based machine reading system is described in Figure 1. The various components of the AV system are Pattern Generation Module, Hypothesis Generation Module, Document Parsing Module, Question Type Analysis Module, Named Entity Recognition (NER) Module, Textual Entailment (TE) Module, Chunk Boundary, Syntactic Similarity Module, Answer Scoring Module and Answer Ranking Module. Each of these modules is now being described in subsequent subsections.

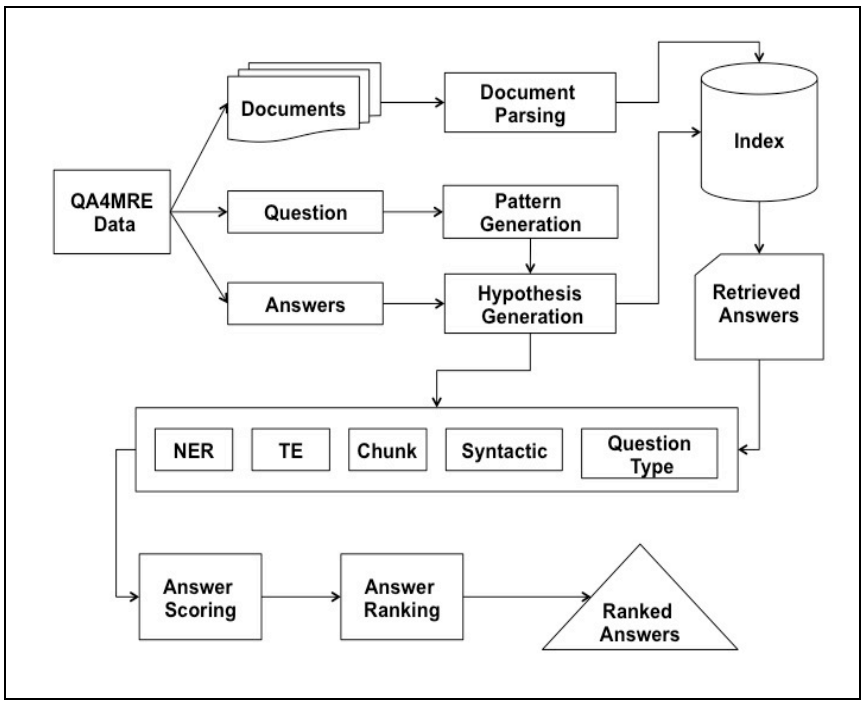

Figure 1. Answer Validation based Machine Reading System

3.1 Pattern Generation Module

At first we convert each question into an affirmative sentence that denotes the answer pattern and place the </answer> template in place of the appropriate answer. The pattern generation module is rule based.

For example, let us consider the question id 7 in doc id 2 of the QA4MRE train set,
Question: Where is the U.S. nuclear waste repository located?
The generated pattern is The U.S. nuclear waste repository is located </answer>.

3.2 Hypothesis Generation Module

After Pattern generation the </answer> template is replaced by each answer option string forming the generated Hypothesis. The generated hypothesis is termed as the query. For example, for question id 7 (QA4MRE Train set), the following hypotheses (or queries) are generated for each of the answer options:
H_1: The U.S. nuclear waste repository is located at Oklo.
H_2: The U.S. nuclear waste repository is located in Morsleben.
H_3: The U.S. nuclear waste repository is located in New Mexico.
H_4: The U.S. nuclear waste repository is located in a suitable geological formation.
H_5: The U.S. nuclear waste repository is located in the U.S. State of Nevada.

3.3 Document Processing and Indexing

The web documents are full of noises mixed with the original content. It is very difficult to identify and separate the noises from the actual content. First of all the documents had to be preprocessed. The document structure is checked and reformatted according to the system requirements.

The corpus is in XML format. All the XML test data has been parsed before indexing using our XML Parser. The XML Parser extracts the sentences from the document. After parsing the documents, they are indexed using Lucene, an open source full text search tool.

3.4 Query Word Identification

After indexing has been done, the queries have to be processed to retrieve relevant sentences from the associated documents. Each answer pattern or query is processed to identify the query words for submission to Lucene.

Certain key characters in the query cause implicit query handling during searching. For example, the dot character between two query words denotes AND of the two query words. Such key characters are thus removed from the question before submission to Lucene. For example, http://wt.jrc.it/ and doug@nutch.org are rephrased as “http wt jrc it” and “doug nutch org” respectively.

The Stop words (using the stop word list[2]) and question words (what, when, where, which etc.) are removed from each question. The remaining words in the question are identified as the query words. Query words may appear in inflected forms in the question. For English, standard Porter Stemming algorithm[3] has been used to stem the query words.

3.5 Sentence Retrieval

After searching each query into the Lucene index, a set of sentences in ranked order for each query is retrieved.

First of all, all query words are fired with AND operator. If at least one sentence is retrieved using the query with AND operator then the query is removed from the query list and need not be searched again. The rest of the queries are fired again with OR operator. OR searching retrieves at least one sentence for each query. Now, the top ranked relevant ten sentences for each query are considered for further processing In case of AND search only the top ranked sentence is considered. Sentence retrieval is the most crucial part of this system. We take only the top ranked relevant sentences assuming that these are the most relevant sentences in the associated document for the question from which the query has been generated.

Each retrieved sentence is considered as the Text (T) and is paired with each generated hypothesis (H). Each T-H pair identified for each answer option corresponding to a question is now assigned a score based on the NER module, Textual Entailment module, Chunking module, Syntactic Similarity module and Question Type module.

3.6 NER Module

It is based on the detection and matching of Named Entities (NEs) [9] in the Retrieved Sentence (T) - generated Hypothesis (H) pair. Once the NEs of the hypothesis and the text have been detected, the next step is to determine the number of NEs in the hypothesis that match in the corresponding retrieved sentence. The measure NE_Match is defined as NE_Match = number of common NEs between T and H/Number of NEs in Hypothesis.

If the value of NE_Match is 1, i.e., 100% of the NEs in the hypothesis match in the text, then the T-H pair is considered as an entailment. The T-H pair is assigned the value "1", otherwise, the pair is assigned the value "0".

3.7 Textual Entailment Module (TE)

This TE module [8] is based on three types of matching, i.e., WordNet based Unigram Match and Bigram Match and Skip-bigram Match.

---

[2] http://members.unine.ch/jacques.savoy/clef/

[3] http://tartarus.org/~martin/PorterStemmer/java.txt

a. WordNet based Unigram Match. In this method, the various unigrams in the hypothesis for each Retrieved Sentence (T) - generated Hypothesis (H) pair are checked for their presence in the retrieved text. WordNet synsets are identified for each of the unmatched unigrams in the hypothesis. If any synset for the H unigram match with any synset of a word in the T then the hypothesis unigram is considered as a successful WordNet based unigram match. If the value of Wordnet_Unigram_Match is 0.75 or more, i.e., 75% or more unigrams in the H match either directly or through WordNet synonyms, then the T-H pair is considered as an entailment. The T-H pair is then assigned the value "1", otherwise, the pair is assigned the value "0".

b. Bigram Match. Each bigram in the hypothesis is searched for a match in the corresponding text part. The measure Bigram_Match is calculated as the fraction of the hypothesis bigrams that match in the corresponding text, i.e., Bigram_Match=(Total number of matched bigrams in a T-H pair /Number of hypothesis bigrams). If the value of Bigram_Match is 0.5 or more, i.e., 50% or more bigrams in the H match in the corresponding T, then the T-H pair is considered as an entailment. The T-H pair is then assigned the value "1", otherwise, the pair is assigned the value "0".

c. Skip-grams. A skip-gram is any combination of n words in the order as they appear in a sentence, allowing arbitrary gaps. In the present work, only 1-skip-bigrams are considered where 1-skip-bigrams are bigrams with one word gap between two words in a sentence. The measure 1-skip_bigram_Match is defined as

1_skip_bigram_Match = skip_gram(T,H) / n,

where skip_gram(T,H) refers to the number of common 1-skip-bigrams (pair of words in order with one word gap) found in T and H and n is the number of 1-skip-bigrams in the hypothesis H. If the value of 1_skip_bigram_Match is 0.5 or more, then the T-H pair is considered as an entailment. The text-hypothesis pair is then assigned the value "1", otherwise, the pair is assigned the value "0".

3.8 Question-Answer Type Analysis Module

The original questions are pre-processed using Stanford Dependency parser [10]. The question type and its expected answer type are generally identified by looking at the question keyword. Table 1 lists the questions and the expected answer types. For example, if the question type is "When", the expected answer type is a "DATE/TIME". The answer string "<a_str>" is parsed by the RASP Parser [9]. If the RASP parser generates the tag "<timex type=date>" then the answer string is "1", otherwise it is "0". For "What" type questions we look for the keyword (e.g., Company) that is related to "What" through a dependency relation. If the keyword is "Company" the expected answer type is "Organization". If the corresponding answer string is tagged by the RASP parser as "Organization", the answer string is marked as "1", otherwise it is "0". If the question type is "How" and the answer string is tagged as "CD" by the RASP parser, the answer string is marked as "1", otherwise it is "0".

Table 1. Question Keyword and Expected Answer.

Question Type	Expected Answer
Who	PERSON
When	DATE / TIME
Where	LOCATION
What	OBJECT
How	MEASURE

3.9 Chunk Module

The question sentences are pre-processed using Stanford dependency parser. The words along with their part of speech (POS) information are passed through a Conditional Random Field (CRF) based chunker [11] to extract phrase level chunks of the questions. A rule-based module is developed to identify the chunk boundaries. The question-retrieved text pairs that achieve the maximum weight are identified and the corresponding answers are tagged as "1". The question-retrieved text pair that receives a zero weight is tagged as "0".

3.10 Syntactic Similarity Module

This module is based on the Stanford dependency parser [9], which normalizes data from the corpus of text and hypothesis pairs, accomplishes the dependency analysis and creates appropriate structures.

3.10.1 Matching Module

After dependency relations are identified for both the retrieved sentence and the hypothesis in each pair, the hypothesis relations are compared with the retrieved text relations. The different features that are compared are noted below. In all the comparisons, a matching score of 1 is considered when the complete dependency relations along with all of its arguments match in both the retrieved sentence and the hypothesis. In case of a partial match for a dependency relation, a matching score of 0.5 is assumed.

a. Subject-Verb Comparison. The system compares hypothesis subject and verb with retrieved sentence subject and verb that are identified through the *nsubj* and *nsubjpass* dependency relations. A matching score of 1 is assigned in case of a complete match. Otherwise, the system considers the following matching process.

b. WordNet Based Subject-Verb Comparison. If the corresponding hypothesis and sentence subjects do match in the subject-verb comparison, but the verbs do not match, then the WordNet distance between the hypothesis and the sentence is compared. If the value of the WordNet distance is less than 0.5, indicating a closeness of the corresponding verbs, then a match is considered and a matching score of 0.5 is assigned. Otherwise, the subject-subject comparison process is applied.

c. Subject-Subject Comparison. The system compares hypothesis subject with sentence subject. If a match is found, a score of 0.5 is assigned to the match.
d. Object-Verb Comparison. The system compares hypothesis object and verb with retrieved sentence object and verb that are identified through dobj dependency relation. In case of a match, a matching score of 0.5 is assigned.
e. WordNet Based Object-Verb Comparison. The system compares hypothesis object with text object. If a match is found then the verb corresponding to the hypothesis object with retrieved sentence object's verb is compared. If the two verbs do not match then the WordNet distance between the two verbs is calculated. If the value of WordNet distance is below 0.5 then a matching score of 0.5 is assigned.
f. Cross Subject-Object Comparison. The system compares hypothesis subject and verb with retrieved sentence object and verb or hypothesis object and verb with retrieved sentence subject and verb. In case of a match, a matching sc ore of 0.5 is assigned.
g. Number Comparison. The system compares numbers along with units in the hypothesis with similar numbers along with units in the retrieved sentence. Units are first compared and if they match then the corresponding numbers are compared. In case of a match, a matching score of 1 is assigned.
h. Noun Comparison. The system compares hypothesis noun words with retrieved sentence noun words that are identified through *nn* dependency relation. In case of a match, a matching score of 1 is assigned.
i. Prepositional Phrase Comparison. The system compares the prepositional dependency relations in the hypothesis with the corresponding relations in the retrieved sentence and then checks for the noun words that are arguments of the relation. In case of a match, a matching score of 1 is assigned.
j. Determiner Comparison. The system compares the determiner in the hypothesis and in the retrieved sentence that are identified through *det* relation. In case of a match, a matching score of 1 is assigned.
k. Other relation Comparison. Besides the above relations that are compared, all other remaining relations are compared verbatim in the hypothesis and in the retrieved sentence. In case of a match, a matching score of 1 is assigned.

API for WordNet Searching RiWordnet[4] provides Java applications with the ability to retrieve data from the WordNet database.

3.11 Answer Scoring Module

In this module, we have got the weight from Named Entity Recognition (NER) Module (Section 3.6), Textual Entailment (TE) Module (Section 3.7), Question Type Analysis Module (Section 3.8), Chunk Boundary (Section 3.9) and Syntactic Similarity Module (Section 3.10).

4 http://www.rednoise.org/rita/wordnet/documentation/index.htm

3.12 Answer Ranking Module

For each question has five hypothesis. Hypothesis are ranked by using NER Module, Textual Entailment Module, Chunking Module, Syntactic Similarity Module, Question type analysis Module. The highest weight hypothesis is the final answer.

4 Question Answering based Machine Reading System Architecture

4.1 Document Processing Module

4.1.1 XML parser

The given XML corpus has been parsed using XML parser. The XML parser extracts the document and associated questions. After parsing, the documents and the associated questions are extracted from the given XML documents and stored in the system.

4.1.2 Answer Pattern Generation

Each question has a number of answer options and the task is to identify the best answer to the question given an associated document. Each question in the system is identified as the **(**question, document) pair represented as $\{q_i, d\_id\}$ where i=1…10. There are 10 questions corresponding to each document. The "WH" word in the question is substituted by the given answer option to generate the answer pattern. The set of WH words include WH_P ={'Who', 'What', 'Where', 'Name', 'Which', 'Whom', 'Why'}. Each answer pattern is represented in the system as $\{d\_id, q\_id_i, a\_id_j\}$, where, d_id=document id, q_id_i= i th query, where i=1…10, a_id_j= j th answer option, where j=1…5.

Let us consider an example.
Question: *Who* is the founder of the SING campaign?
Answer Option: *Nelson Mandela*
WH_P: *who*
Generated Answer Pattern: Nelson Mandela is the founder of the SING campaign

Each answer pattern is stored in the system as the pair (PAT, KL) where,

PAT= ***P***robable ***A***nswer ***T***ext, which is the generated answer pattern and
KL= ***K***eyword ***L***ist, is a list of words after removing the stop words.
For example, the above generated answer pattern is stored as

PAT= "*Nelson Mandela is the founder of the SING campaign*"

KL= "*Nelson*", "*Mandela*", "*founder*", "*SIGN*", "*campaign*".

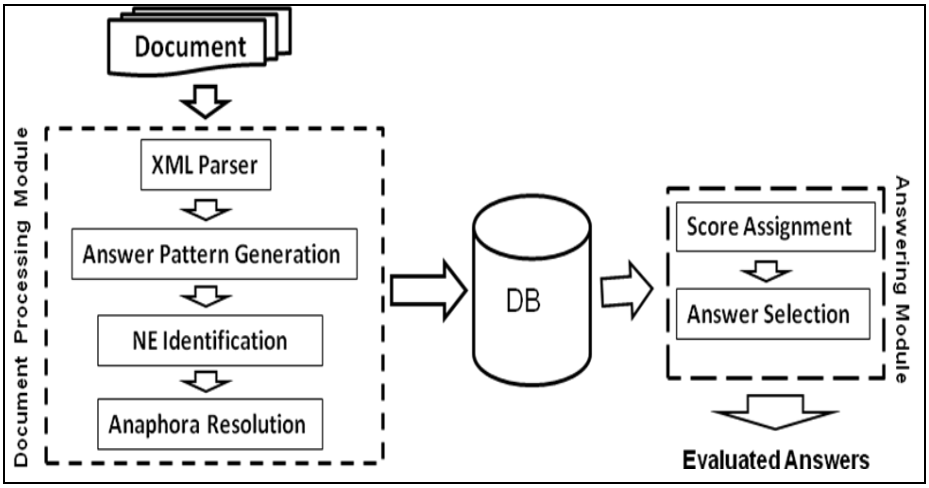

Figure 2. Question Answering based Machine Reading System Architecture

4.1.3 Named Entity (NE) Identification

For each question, the system must identify the correct answer among the proposed alternative answer options. Each generated answer pattern corresponding to a question is compared with each sentence in the document to assign an inference score. The score assignment module requires that the named entities in each sentence and in each answer pattern are identified. The CRF-based Stanford Named Entity Tagger[5] (NE Tagger) has been used to identify and mark the named entities in the documents and queries. The tagged documents and queries are passed to the lexical inference sub-module.

4.1.4 Anaphora Resolution

It has been observed that resolving the anaphors in the sentences in the documents improves the inference score of the sentence with respect to each associated answer pattern. The following basic anaphora resolution techniques have been applied in the present task.

(a) Each first person personal pronoun in the set PN_I = {'I', 'me', 'my', 'myself'} generally refers the author of the document as describer. For example, the anaphors in the following sentence can be resolved in the following steps:

I *am going to share with you the story as to how* **I** *have become an HIV/AIDS campaigner*.

[5] http://nlp.stanford.edu/ner/index.shtml

Step1: **< PN_I >** *am going to share with you the story as to how* **< PN_I >** *have become an HIV/AIDS campaigner.*
Step2: **< PN_I= "author" value= "Annie Lennox">** *am going to share with you the story as to how* **< PN_I= "author" value= "Annie Lennox" >** *have become an HIV/AIDS campaigner.*
Step3: **<NE=Person value= Annie Lennox>** *am going to share with you the story as to how* **<NE=Person value= Annie Lennox>** *have become an HIV/AIDS campaigner.*

In direct speech sentences PN_I refers to the first named entity (speaker) of that sentence. For example, the anaphors in the following sentence can be resolved in the following steps:
Frankie said, "**I** am number 22 in line, and **I** can see the needle coming down towards **me**, and there is blood all over the place.

Step 1: **<NE=Person value=Frankie>** said, " **< PN_I >** *am number 22 in line, and* **< PN_I >** *can see the needle coming down towards* **< PN_I >**, *and there is blood all over the place.*
Step 2: **<NE=Person value=Frankie>** said, "**< PN_I= "NE" value= "Frankie">"** *am number 22 in line, and* **< PN_I= "NE" value= "Frankie">** *can see the needle coming down towards* **< PN_I= "NE" value= "Frankie" >**, *and there is blood all over the place.*
Step 3: **<NE= "Person" value= "Frankie">** said, " **<NE= "Person" value= "Frankie">** *am number 22 in line, and* **<NE=Person value=Frankie>** *can see the needle coming down towards* **<NE=Person value=Frankie>**, *and there is blood all over the place.*

(b) Each second person personal pronoun in the set $PN_{He/She}$ = {'he', 'his', 'him', 'her', 'she'} generally refers the last NE of the previous sentence. For example, the anaphors in the following sentence can be resolved in the following steps:

I was invited to take part in the launch of Nelson Mandela's 46664 Foundation. That is his HIV/AIDS foundation.

Step 1: < **PN_I** > was invited to take part in the launch of **<NE= "Person" value= "Nelson Mandela">**'s 46664 Foundation. That is **<$PN_{He/She}$>** HIV/AIDS foundation.
Step 2: **< PN_I= "author" value= "Annie Lennox">**was invited to take part in the launch of **<NE=Person value= "Nelson Mandela">**'s 46664 Foundation. That is **<$PN_{He/She}$ = "PREV_NE" value= "Nelson Mandela">** HIV/AIDS foundation.
Step 3: **< PN_I= "author" value= "Annie Lennox">**was invited to take part in the launch of **<NE= "Person" value= "Nelson Mandela">**'s 46664 Foundation. That is **<NE= "Person" value= "Nelson Mandela">** HIV/AIDS foundation.

But, in indirect speech sentences, $PN_{He/She}$ refers to the first named entity (speaker) of that sentence. For example, the anaphors in the following sentence can be resolved in the following steps:

Alexander Graham Bell famously said that on his first successful telephone Call.
Step 1: **<NE= "Person" value= "Alexander Graham Bell"**> *famously said that on* < $\mathbf{PN_{He/She}}$ > *first successful telephone Call*.
Step 2: **<NE= "Person" value= "Alexander Graham Bell"**> *famously said that on* < $\mathbf{PN_{He/She}}$ = **"SEN_NE"** value= "**Alexander Graham Bell">** *first successful telephone Call*.
Step 3: **<NE= "Person" value= "Alexander Graham Bell"**> *famously said that on* **<NE="Person"** value="**Alexander Graham Bell">***first successful telephone Call*.

4.2 Answering Module

Each sentence in the associated document is assigned an inference score with respect to each generated answer pattern.

4.2.1 Scoring Assignment

This module takes query frame as input and returns score as output. The algorithm *AnswerScore* describes the scoring procedure.

Table 2. Algorithm AnswerScore (Sentence, PAT, KL)

Algorithm AnswerScore (sentence, PAT, KL)
Step 1: [*Initialization*] score = 0 keywordmatched = 0 // count no of matched keyword
Step 2: [*Check whether PAT matches in a sentence*] If **PAT** matches in a sentence then Score = 1 goto step 5
Step 3: [*Check each keyword in KL]* For each keyword in **KL** If keyword matches in a sentence then Score = score + 1 / (number of keywords -1) Keywordmatched = keywordmatched + 1
Step 4: [*Check whether all the keywords have matched*] If (keywordmatched = = total keywords – 1) then Score =1
Step 5: Return score
End

Now, for each given answer option a score is calculated and the answer option with highest score is taken as correct answer for the given query. The algorithm *SelectAnswerOption* describes the option selection procedure.

Table 3. Algorithm SelectAnswerOption (Answer Set)

Algorithm *SelectAnswerOption(answer set)*
Step 1: [*Initialization*]
correct_option= ∞ // not answered
Step 2: [*Calculate score for each sentence*]
For each sentence S_i € Sentences and answer pattern q_j€ Q
Where, j=1…5
A_{ji}=AnswerScore (S_i, PAT, KL)
End For
Step 3: [*Assign score to each option*]
For answer pattern q_j€ Q
AQ_i=maximum evaluated score for {S_1, S_2,…..S_n};
Where AQ_i is the score of i^{th} option
End For
Step 4: [Select the answer option]
correct_option= index of maximum AQ={ AQ_1, AQ_2, AQ_3, AQ_4, AQ_5 }
END

5 Hybrid Machine Reading System Architecture

We have got the ranking score for each answer option from the AV System and also from the QA system. Now we have used voting technique to select the final answer using the selected answer from our IR (Nutch) system. The Nutch system also provides a ranking score for each answer option. We have passed all the three answers from the different systems, i.e., the highest ranked answer option, to the voting module, which selects the final answer option based on the highest vote of the answers. The system architecture is shown in figure 3. When any two systems identify the same answer then this answer gets automatically selected as the final answer as it has at least two votes. But when three different answers have been selected from the three systems for a question, then this voting module has to give priority to a system. We have submitted two runs from our two different hybrid system based on this voting module. In the hybrid system of run id 4 we have given priority to the QA system and in the hybrid system of run id 5 we have given priority to the AV system.

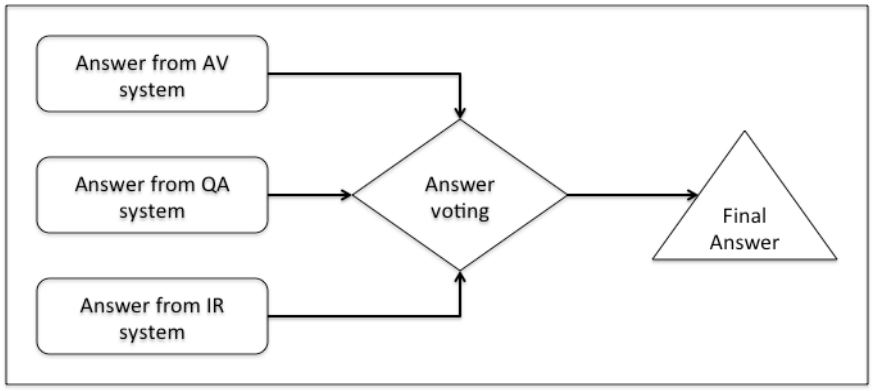

Figure 3. Hybrid Machine Reading System Architecture

6 Evaluation

In QA4MRE track, we have submitted total seven runs. But run 1 and run 2 are identical because same run has been submitted twice. Run 4 and run 5 also are identical because the same run has been submitted twice. So we have five unique run from three different systems. Evaluation results are shown in table 4.

For System 1: Run 1, Run 2 (Based on Answer Validation System)
For System 2: Run 3 (Based on Question Answering System)
For System 3: Run 4, Run 5 (Hybrid system)

The main measure used in this evaluation campaign is c@1, which is defined in equation 1.

$$c@1 = \frac{1}{n}\left(n_R + n_U \frac{n_R}{n}\right) \quad (1)$$

Where, n_R: the number of correctly answered questions, n_U: number of unanswered questions and n: the total number of questions

Evaluation at question-answering level:

Table 4. Overall Evaluation Results

Run ID	C1	C2	C3	C4	C5	C6	C7	c@1
1	120	0	19	101	0	0	0	0.16
2	120	0	25	95	0	0	0	0.21
3	120	38	82	0	0	0	0	0.32
4	**98**	**22**	**58**	**40**	**0**	**0**	**22**	**0.57**
5	109	11	52	57	0	0	11	0.47

C1: Number of questions ANSWERED
C2: Number of questions UNANSWERED
C3: Number of questions ANSWERED with RIGHT candidate answer
C4: Number of questions ANSWERED with WRONG candidate answer
C5: Number of questions UNANSWERED with RIGHT candidate answer
C6: Number of questions UNANSWERED with WRONG candidate answer
C7: Number of questions UNANSWERED with EMPTY candidate

Evaluation at reading-test level:

Table 5. Evaluation Result for Median, Average and Standard Deviation.

Run Id	Median	Average	Standard Deviation
1	0.15	0.16	0.13
2	0.20	0.21	0.12
3	0.30	0.32	0.15
4	**0.74**	**0.58**	**0.37**
5	0.45	0.48	0.28

Overall QA4MRE campaign statistics: lowest c@1: 0.02, highest c@1: 0.57, Average c@1: 0.21

7 Conclusion

The question answering system has been developed as part of the participation in the QA4MRE track as part of the CLEF 2011 evaluation campaign. The overall system has been evaluated using the evaluation metrics provided as part of the QA4MRE 2011 track. The evaluation results are satisfactory considering that this is the second participation in the track. Future works will be motivated towards improving the performance of the system.

References

1. Anselmo Peñas, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, Corina Forăscu, Iñaki Alegria, Danilo Giampiccolo, Nicolas Moreau, Petya Osenova.: Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In Working Notes for the CLEF 2009 Workshop, 30 September-2 October, 2009, Corfu, Greece.
2. Anselmo Peñas, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, Corina Forăscu and Cristina Mota.: Overview of ResPubliQA 2010: Question Answering Evaluation over European Legislation. In Working Notes for the CLEF 2010 Workshop, Padua, Italy, 20-23 September 2010.
3. Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, Corina Forascu, Caroline Sporleder. Overview of QA4MRE at CLEF 2011: Question Answering for Machine Reading Evaluation, Working Notes of CLEF 2011. (2011)
4. Peñas, A.,Rodrigo, Á. , Sama, V., Verdejo, F.: Overview of the answer validation exercise 2006. Working Notes of CLEF 2006. (2006)
5. Peñas, A., Rodrigo, Á, Verdejo, F.: Overview of the Answer Validation Exercise 2007. Working Notes of CLEF 2007. (2007)
6. Rodrigo, Á., Peñas, A., Verdejo, F.: Overview of the answer validation exercise 2008. Working Notes of CLEF 2008. (2008).
7. Partha Pakray, Pinaki Bhaskar, Santanu Pal, Dipankar Das, Sivaji Bandyopadhyay and Alexander Gelbukh: JU_CSE_TE: System Description QA@CLEF 2010 – ResPubliQA. CLEF 2010 Workshop on Multiple Language Question Answering (MLQA 2010).
8. Pakray, P., Gelbukh, A., Bandyopadhyay, S.: Answer Validation using Textual Entailment. 12th CICLing, Lecture Notes in Computer Science, 2011, Volume 6609/2011, 353-364, DOI: 10.1007/978-3-642-19437-5_29. (2011)
9. E. Briscoe, J. Carroll, and R. Watson.: The Second Release of the RASP System. In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions.
10. Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning.: Generating Typed Dependency Parses from Phrase Structure Parses. In 5th International Conference on Language Resources and Evaluation (LREC) (2006)
11. Xuan-Hieu Phan.: CRFChunker: CRF English Phrase Chunker. PACLIC 2006. (2006)