Wireless Personal Communications
https://doi.org/10.1007/s11277-019-06501-z

A Information Retrieval Based on Question and Answering and NER for Unstructured Information Without Using SQL

Partha Sarathy Banerjee[1] · Baisakhi Chakraborty[1] · Deepak Tripathi[2] · Hardik Gupta[2] · Sourabh S. Kumar[2]

Abstract
In today's world, the availability of information in the form of unstructured data is in abundance. The unstructured information received is more often than not in the form of natural language text. For any defense establishment, the spy data or any sensitive information received may be best utilized when the information can be extracted efficiently and easily. The proposed model is applicable wherever the influx of text-heavy (unstructured data) is high like the information from the world wide web, documents related to a particular domain, or any other source where the information is in the form of natural language. The proposed Natural Language Information Interpretation and Representation System (NLIIRS) accepts the information in the form of natural language text, processes the information and allows the user to retrieve information by rendering questions in natural language. The questions thus asked by the user are responded by NLIIRS in the form of factoid or phrase based answers. In comparison to the conventional question and answering systems the proposed NLIIRS uses the advantages of both named entity recognition as well as sequential pattern matching based answer search technique. The proposed technique helps us to avoid the use of structured query language (SQL) at the back-end for information processing, storage and extraction. The conversion of user query to SQL statements and also storing the unstructured text in the form of relation tables can be avoided by using NLIIRS. By using this approach in our novel text processing algorithm, after every execution step, the pattern matching and extraction process of the answers to the queries becomes concise and faster. The whole system has been designed on natural language tool kit of Stanford University which helped us to generate parts of speech tag, tokenize the data, and forming tree structure. The novel text processing algorithm utilizes the lemmatizer, stemmer and ne_chunker to prepare the text for information retrieval via Q&A. The advantage of this system is that it does not need training. This system will enable the user to retrieve any information of his/her choice from the available unstructured information.

Keywords Information retrieval · NLP · Unstructured data · NLTK · Question and answering · NER

✉ Partha Sarathy Banerjee
partha1010@gmail.com

Extended author information available on the last page of the article

Published online: 04 May 2019

1 Introduction

The data available or intercepted during the time of war is mostly in the form of unstructured text. Even the data available on the world wide web as well as of over various electronic communication channel is mostly in the form of unstructured data. This unstructured data is in the form of Natural Language Text. The best test of the fact that, whether the system has comprehended the input text or not is to perform a question and answering over the information that has been provided to it. Information retrieval using question and answering can be performed in three different situations. The first condition is when structured information in the form of table is available. The options for information retrieval in these cases are generally in the form of structured query language. The second condition is when no structured information is available but the question and answering system (Q&A system) is trained with a particular dataset. The third situation is when the system doesn't use any training data, and the asked questions are fired directly on the input text by use of the text processing algorithm. These types of systems are comparatively faster. When the available information in day to day life is increasing it is essential to have a system that can accept data in unstructured form and allows the user to retrieve information easily. In structured data the information can be retrieved very easily by the use of structured query language. While retrieving the information from a structured data the user has to be skilled enough to learn the query language. The proposed system provides the freedom to the user to ask a question in natural language without learning the complex syntax of structured query language. The Natural Language Information Interpretation and Representation System (NLIIRS) makes the process of information retrieval easier by allowing the user to ask the question in natural language. This NLIIRS accepts the input information in the form of any natural language text. This accepted text is processed by the proposed text processing algorithm and takes the user queries in the form of question asked in natural language. The NLIIRS provides the answer in factoid form. It answers to the queries of "Who", "Where" and "When" category. The system answers queries of "What" category when the sentences are simple.

2 Utility

This system finds utility where a robust framework is required to process the available information which is in natural language. It finds usage in mission critical situations like real-time processing of information and also retrieving the required information if it is available in the input text. For example, in case of defense utility when a user encounters any information in unstructured form, he or she can then extract any relevant information by simply putting a question in natural language. When sensitive information is to be extracted on real time, from unstructured data then this system finds its utility. The data gets processed and the information is extracted without any tampering hence enhancing the security of the system. In general, it can also be utilized for extracting the factoid information from a given unstructured data, for example to extract the birth place of Lord Rama from the given book of Ramayana. This kind of usage makes it a right candidate for the educational industry. It can also be used as browser extension to search and retrieve only the relevant information from a provided webpage or web blog.

3 Advantages

The main advantages of the proposed NLIIRS is that the system takes less memory then Structured Query Language (SQL) based system as the proposed model searches for relevant nodes rather than searching the entire column in case of a SQL system. The system can be updated with ease as more question forms or cases can be added easily. In case a better parts of speech (POS) tagger or named entity recognizer (NER) is available in future, the system could update its NER package. User training of the system is not required. The proposed work abstains from answering the question when the relevant information is not available in the input text. Generally other models give wrong answers as output, when the data is not available in the input text. Some systems use a trained system for Q&A. This training process involves training the Q&A system on a particular domain on which the Q&A is to be performed.

Many previously proposed modes had to store the input text but in this proposed model the data is not stored hence making the system much faster. One of the unique features of this system is that it supports text to speech conversion hence the user can just input the text on which the Q&A is to be done. After this the user has to ask a question and the system will automatically provide a voice output of the factoid answer. It can also answer for the passive form of natural language text like, if the input text is: "The apple was eaten by John" and the natural language query is who ate apple then the NLIIRS system will give correct answer. Another most important feature of the proposed model is that it can provide correct answer to query whose result have more than one proper noun.

4 Related Work

Information extraction concept via question and answering was attempted way back in 1961 when a module "Baseball" was proposed as stated in [1]. In this work, the proposed system has tried to provide information regarding a baseball tournament through Q&A. After the successful moon landing a system named LUNAR was developed as in [2]. The task of the LUNAR module is to provide information regarding the soil samples of the lunar missions and this work also provided information on the basis of Q&A. A model "MASQUE" proposed in 1993 [3] converted the questions into SQL queries and then the intended "Wh" category question is matched and is provided as answer. Later on, the same authors proposed a more advanced version of their own work as in [4]. In this work the author provided an interface where questions could be used for information extraction via an interface. Another Q&A system named "QUARC" was developed in 2000 which classified the various "Wh" type questions. The understandability of documents for answering the queries on the particular domain has always been a challenging task. The user-annotated data for training the systems is not helpful because of its scarcity. The basic Named Entity Recognition (NER) based Question and Answering (Q&A) uses the concepts of machine learning as stated in [5]. These systems are generally not so much reliable in expressing the semantics of the natural language text. The work stated in [5] uses the concept of skip gram to resolve the above stated problem. The NER problem is addressed using the semantic analysis which provides a faster solution. In this work an external data set is used which has a positive impact on the precision and recall values. The work in [6] proposes a model for generating to-the-point answers for an unstructured data which may

be from any domain. It states that the previous systems have been closed-domain and had lesser efficiency. The work in [6] classifies the questions in six different classes and proposes an open domain question and answering system. In the work described in [7], answer recapitulation is uprooted from the text with numerous features which is then sent to CNN with the question to acquire a brief, and systematic semantic representation. Dissimilar to the foregoing deep models, unrelated information is separated, and superior representations are produced for question and answer (Q&A), which is compulsory for question and answering of non-factoid ones. In the work illustrated in [8], we start by preparing a document recapitulation method to take out the passage level answers for queries which are non-factoid and mentioned as answer-prejudiced synopses. We come up with using exterior data or information from analogous Community Question Answering (CQA) content to finely recognize answer bearing sentences. We contribute a recommendation of the foremost use of our suggested approaches concerning the accessibility of different standard levels of analogous CQA content. In the work stated in [9], the author has used a model namely recursive neural network (RNN) model because text categorization methods for different jobs like factoid question and answering consistently use manually explained string matching regulations or bag of words description. These methods are fruitless when query text carries very little amount of individual words that are expressive of the answer. In the work described in [10], the author has discussed about Q&A, which is main foundation of NLP (Natural Language Processing), in which a system (or a machine) should be in a position to interpret the user language (or human language). This interpretation is at a level that an individual can perform queries to the system in appropriate language that can be understood. After that it should be able to get the answers in the same mode as that of the user. The question and answering on any corpus is now not just limited to a specific user as in [11] but to a very large number of people interacting on a social platform. Putting any natural language text on a social question and answering domain provides the advantage of collecting thousands of questions and lakhs of answers. Out of the answers provided, many answers come out to be highly valuable for a large number of users. Retrieving these answers when similar question is fired again in future involves huge complexity. In the work stated in [11], the author states that a score may be provided to each answer depending on its relevance and hence fetching the correct set of answers for a particular question becomes much efficient. Question and answering becomes difficult when structured data are not available, and the domain of the corpus is relatively large. Larger coverage and more accurate answer combination is a proper solution as proposed in [12]. In the work stated in [13] the training data set enhances the performance and this specific work focuses on training the system with data pertaining to that class of problem. The work discussed in [14] first puts emphasis on classification of question category then experimenting with the various question types. After the questions after classified then they were fired on a specific domain-based text which is in this case “Medical” data. Last but not the least one of the most efficient work on Q&A has been proposed in [15] which is also the comparative work with the proposed NLIIRS.

5 Prerequisites

Since the user gets the facility to ask question in natural language and there is no need for him or her to study the SQL hence there are some prerequisites for this system. The user must have some basic idea of the intercepted unstructured data so that the user can ask

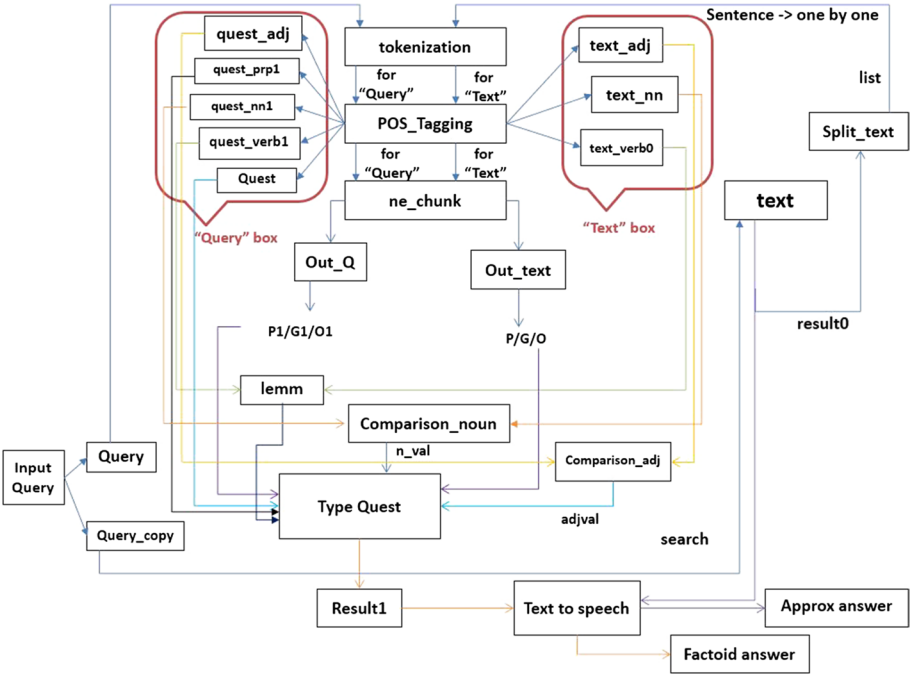


Fig. 1 System architecture NLIIRS

relevant questions. For performing question and answering the natural language query must be preceded by a "Wh" word like "Who", "Where", "When" and "What". It is expected that the users should use proper English language for better efficiency of the system. The proposed system does not support a question mark and this trait has no impact on the functioning of the logical part of the system.

6 System Architecture

The system architecture as described in Fig. 1, takes the input in the form of natural language text and stores it in a list, which is an infinite length of array, as string. When user puts a query related to a text, which is shown as "Query" in the diagram, the system also makes an additional copy of that query i.e. Query_copy. In the Query_copy section we separate out the questioning words like "who", "when", "where" and "what". Now the left behind text in the Query_copy is processed by "word.splitter". These remaining words are then stored in a list and are sequentially searched in the natural language text input. Let's say after the search process of the first word from the list in the input text, if it appears in "N" sentences then these sentences are separated out. The next word from the list is then searched in those "N" sentences and now the system will separate out "N-M" sentences, where "M" is the number of sentences in which the second word of the list does not exist. Now for the next word search, the value of "N" will be updated as "N-M". This process of searching the subsequent words will continue until no more words are left in the word list. The remaining sentences will be shown as approximate answer. Now the probability of

getting the accurate factoid answer to the generated query will be high in these sentences itself. These sentences will be stored in "result0" as a string.

Now the "query" with the questioning words like "who", "when", "where" and "what" is passed to the "tokenizer" for tokenization. The outcome of this tokenization is passed to the pos tagger for pos tagging purpose. The outcome of pos tagging in the form of a list is subdivided under the heads: "quest_adj" for adjectives, "quest_prp1" for prepositions, "quest_nn1" for common noun, "quest_verb1" for verbs and at last "quest" for question words like "who", "when", "where" and "what" lists, for classification. While classifying "quest_verb1" for verbs the system clubs up all the verb types classified by Natural Language Tool Kit (NLTK), under one head "quest_verb1". The same list, which is the output of pos tagging is also passed to "ne_chunker". The "ne_chunker" chunks/classifies the list into "PERSON", "GPE" and "ORGANISATION" as output shown as P1, G1 & O1.

Now result0 is passed to split_text function that splits the text of result0 into paragraphs and the paragraphs into sentences and stores it in a list. The system can now pass the sentences one by one for tokenizing. The outcome of this tokenization is passed to the pos tagger for pos tagging purpose. The outcome of pos tagging in the form of a list is subdivided under the heads: "text_adj" for adjectives, "text_nn" for common noun, "text_verb0" for verbs. The same list, which is the output of pos tagging is also passed to "ne_chunker". The "ne_chunker" chunks/classifies the list into "PERSON", "GPE" and "ORGANISATION" as output shown as P, G & O, as in the case of query processing described above. While classifying "text_verb0" for verbs the system clubs up all the verb types classified by Natural Language Tool Kit (NLTK), under one head "text_verb0". Now the "quest_verb1" as well as the "text_verb0" are passed to "lemm" for lemmatizing and stemming. This step finds out the root verb from the various verbs available like, from "go", "going", "gone" of the root verb "go". It will trace out the root verb "go". When the root verb of both question as well as text is same the system compares the noun in "comparison_noun" and adjective in "comparison_adj" of both i.e. for text as well as for query. Now the output of "comparison_noun", "comparison_adj" as well as "lemm" is passed to "type_quest". The "type_quest" will provide the output as factoid answer, if sufficient data is available, on the basis of logic described in algorithm 5. The factoid answer will be stored in "Result1" for displaying the factoid result and the same "Result1" segment will now be passed to text to speech for reproducing the machine spoken output.

7 Algorithms

Algorithm 1 – Searching of Related Text in unstructured data.
Input – User's query in natural language
Output – Approximate text related to user's query

```
1.def function → splitter(D_text,Q_text):
2.   def function →splitParagraphIntoSentences(Text):
3.      sentenceList←return
4.   def function →gh(x):
5.      call the function splitParagraphIntoSentences() by
        passing Text as an argument
6.   def function → split_line(pass user's query):
7.      words ← split the query into words using split()
8.      l ← find length of words array
9.      a← fetch the first element ie. words[0]
10.     for word in words :
11.        concatnate word of words in empty string txt
           and each word separated by whitespace.
12.     tt←txt.lstrip(a+" ")
13.     l2←find length of tt
14.     word_list←split tt into words using split()
15.     return →word_list
16.  x=D_text & y=Q_text
17.  in_bound←call the function gh() by passing
                  unstructured text as an argument
18.  a←call the function split_line() by passing query as
        an argument
19.  for
20.     sub=str(a[i])
21.     q= "".join(s for s in in_bound if sub.lower() in
                                              s.lower())
22.     in_bound= call the function gh() by passing q as an
                  argument
23.  for
24.     concatenate element of in_bound in an empty string
        A_ans and each word separated by whitespace.
25.  return →A_ans
```

Algorithm 2 – Query Processing
Input – User query in natural language
Output – Stores the similar POS tagged data in separate lists

1. **def** function → **query_single(pass user's query):**
2. tokenized ← sent_tokenize(pass user's query)
3. **for**
4. Tokenize the user's query and then do pos tagging using nltk.pos_tag method, store the output in a list named as tagged1.
5. **for**
6. under the tags → NNP, NN, VBZ, VBG, PRP, VBN, VBD, WRB, WP
7. The tagged POS are classified and append words of similar tags in various **List** → NNP1, NN1, VBZ1, VBG1, PRP1, VBN1, VBD1, WRB1, WP1
8. Apply ner chunking on tagged1 list using ne_chunk method and store the output in chunk1
9. **for**
10. Extract the output from tree in a list - ne_in_sent1
11. **for**
12. find PEROSN, GPE, ORGANAIZATION using ner_chunker
13. **EXTND** verb in a list **Verb1** under the tags→ **vbd1, vbg1, vbz1, vbn1, vbp1**
14. **EXTND** wp, wrb10
15. **return** → (verb1, nnp1, prp1, nn1, jj1, quest, person1, gpe1, organization1)

Algorithm 3 – Comparison of Verb, Noun, Adjectives
Input – Verb used in query and Verb in each sentence of text, Noun used in query and Noun in each sentence of text, Adjective used in query and Adjective in each sentence of text.
Output – Whether the adjective, verb & noun of query matches with that of adjective, verb & noun of any sentence.

1. **def** function → **lemm(verb0, verb1):**
2. Lemmatize the verb0: verb used in query and verb1:verb used in a sentence using nltk.stem.WordNetLemmatizer().lemmatize() and store its output in stem1 and stem2
3. **if** (if both stem1 and stem2 are eqal):
4. **return** (y,stem=1)
5. **else return** (n,stem=0)
6. **def** function → **com_noun(**nn1 , nn):
7. Compare the noun used in sentences of text and query
8. **if (nouns are common in both the list)**
9. **return** ptr=1
10. **else return** ptr=0
11. **def** function → **com_adj(jj1, jj):**
12. Compare the noun used in sentences of text and query
13. **if** (Compare the adj used in sentences of text and query)
14. **return** jjval=1
15. **else return** jjval

Algorithm 4 – Sequential Sentence Processing
Input – The Corpus Text
Output – The output of ne_chunker under the category of person , gpe, organization

1. **def** function → **para_to_list(txt):**
2. convert para to list named as inbound using **[Algorithm 1]**
3. **def** function → **chunker(x):**
4. chunk ←ne_chunk(x)
5. **for** subtree in **chunk**:
6. **if** type(subtree) == Tree:
7. Extract the output from tree in a list - ne_in_sent
8. **for** i in range(g):
9. find PERSON, GPE, ORGANIZATION using ner_chunker
10. **return** →(person , gpe, organization)

Algorithm 5 – Question Type Handling Algorithm

Input – person1, person, gpe1, gpe, quest, organization1, organization, prp1, jj1, jj_i, ptr, jjval, stem cd_i values for "Wh" category questions

Output – Processes answers on the basis of conditions

1. **def** function → **type_quest(**person1,person,gpe1,gpe, quest, organization1, organization, prp1, jj1, jj_i, ptr, jjval, stem, cd_i**):**
2. the function will identify the question type i.e. who, what etc.
3. **if (quest[0]=='Where'):**
4. Content matching for the entries present in both the question statement and raw text available takes place
5. If matching is successful, the adequate GPE, ORGANIZATION is used to answer the query as per requirement
6. **elif** (quest[0] ==**'Who'):**
7. entities and POS tagged words are matched with the entities from the raw text
8. **if** (matching is successful):
9. adequate PERSON is used to answer the query
10. **otherwise**:
11. "insufficient data or no match"
12. **elif (quest[0] =='When'):**
13. Content matching for the entries present in both the question statement and raw text available takes place
14. **if** matching is successful, then it will show output

Algorithm 6 – Main Invocation
Input – The Natural Language Text
Output – Factoid answer of user's query

```
1.def function → QPA(Text, Query):
2.   A_ans ←call Algorithm1 for finding the related text
              from raw text by passing raw text as well as
              query - splitter(raw_text, query) [Algorithm 1]
3.   find the outcome of POS tagging and NER_Chunking
     by query- query_single(query) [Algorithm 2]
4.   in_bound←convert paragraph to list using [Algorithm 4]
5.   for
6.      apply tokenization on each sentence.
7.      for
8.         tagged – apply POS tagging on each sentence
9.         for
10.           if tagged[i][1]!='.':
11.          The tagged POS are classified and append
                words of similar tags in various List →
                NNP1, NN1, VBZ1, VBG1, PRP1,
                VBN1, VBD1, WRB1, WP1, cd_i
12.     find person, gpe, organization by passing tagged
        output to chunker(tagged) [Algorithm 4]
13.     EXTND verb in a list Verb0A under the tags→
                vbd1_i, vbg1_i, vbz1_i, vbn1_i, vbp1_i
14.     EXTND wp, wrb10
15.     result0, stem←lemm(verb0A, verbQ) [Algorithm 3]
16.     if (result0=="stem are equal"):
17.        ptr←com_noun(nn1, nn_i)[Algorithm 3]
18.        jjval←com_adj(jj1, jj_i)[Algorithm 3]
19.        type_quest (person1, person, gpe1, gpe, quest,
              organization1, organization, prp1, jj1, jj_i,
              ptr, jjval, stem, cd_i)[Algorithm 5]
20.     else:
21.        ptr←com_noun(nn1, nn_i)
22.        jjval←com_adj(jj1, jj_i)
23.        type_quest (person1, person, gpe1, gpe, quest,
              organization1, organization, prp1, jj1, jj_i,
              ptr, jjval, stem, cd_i)[Algorithm 5]
24.a←QPA (Text, query)
```

7.1 Description of the algorithms used in NLIIRS

In this section we have provided an example that how the critical information can be processed by the algorithm of the system during the time of war or even intelligence by use of certain examples. The abbreviation used in the algorithm above is mentioned in Table 1.

Input Text → Jack lives in London. Tom is a good person. Paris is the capital of France.
Query → Who lives in London

[1] Main Invocation Algorithm

The system invokes the Main Invocation Algorithm. At first the system accepts the input as an unstructured text in natural language as well as a query from the user in natural

Table 1 Abbreviation for text processing algorithm

S. no.	Abbreviation	Description
1	D_text	Unstructured data on which Q&A will be performed
2	Q_text	Query which will be performed on D_text
3	POS	Part of speech
4	NNP	Proper noun
5	NN	Common noun
6	VBZ	Verb(3rd) person
7	VBG	Present participle verb
8	PRP	Pronoun
9	VBN	Past participle verb
10	VBD	Past verb
11	WRB	Wh-adverb
12	WP	Wh-pronoun
13	EXTND	Extend function
14	+	For concatenation
15	QPA	Query processing algorithm
16	GPE	Global positioning entity

language. The input and the query both in natural language are accepted by the function Query Processing Algorithm (QPA). The QPA makes a copy of Query as Query_copy. Now the remaining algorithms are invoked sequentially.

[1.1] Reducer Algorithm→The QPA passes the input text and Query_copy as a string to the Reducer Algorithm. It splits a given Input into individual sentences and stores each sentence as an element in a list. It removes the questioning words like 'who', 'where', 'what' etc. and splits the remaining sentence. It searches each word one by one and returns the output as an approximate answer. QPA invokes Algorithm 2.

[1.2] Query processing algorithm→QPA invokes query processing that tokenizes, POS tags and applies ne_chunker. It also stores the result in various List (infinite length of array). QPA invokes Algorithm 4.

[1.3] Algorithm 4→TQPA invokes algorithm 4 for converting paragraphs into list in which element of list are sentences of the paragraphs.

QPA tokenizes and POS tags each sentence of the input text stored in the List. The output of POS tagger is classified under the tags: NNP, NN, VBZ, VBG, PRP, VBN, VBD, WRB, WP and appends the words of similar tags in various Lists: NNP1, NN1, VBZ1, VBG1, PRP1, VBN1, VBD1, WRB1, WP1.

QPA invokes the algorithm 4 for chunking the output of tokenization. QPA "extend" (it appends each type of verbs in the list verb0) various types of verbs under the tags assigned by NLTK, in a list "Verb0" and again "extend" question words of various types in list: wp, wrb10 (both described in abbreviation table). Now QPA invokes the Algorithm 3.

[1.4] Comparison Algorithm→ TThis algorithm is used for comparing Verb, Noun, Adjectives of Query and each input sentence. This algorithm uses lemmatizer of nltk for comparing the Verbs. Then QPA Algorithm invokes Algorithm 5.

[1.5] Algorithm 5→ This Algorithm is used for getting factoid answer of user's query. It finds the type of query by differentiating it on the basis of question words and then, it uses the algorithm 5, to provide answer of query. At last it will provide the factoid answer.

8 Results

This section shows the test results of the proposed NLIIRS on unstructured text for various question types: Text→JJohn is playing football in America. Tom is living in America. Tom cruise is a good person. Paris is the capital of France. Paris is good place. Tata is a well-known person. The Output 1 shown below shows the output for the "Who" question "Who is playing":

```
Your question is =
Who is playing
Approximate answer is  =
John is playing football in America.
question is ['Who']
Person is  []
GPE is  []
Organization is  []
verb ['playing', 'is']
Question ['Who']
**********Text processing
starts.**********
John is playing football in America.
[('John', 'NNP'), ('is', 'VBZ'), ('playing', 'VBG'), ('football',
'NN'), ('in', 'IN'), ('America', 'NNP'), ('.', '.')]
nnp is ['John', 'America']
nn is ['football']
vbz is ['is']
vbg is ['playing']
prp is []
IN is ['in']
JJ is []
(S
  (PERSON John/NNP)
  is/VBZ
  playing/VBG
  football/NN
  in/IN
  (GPE America/NNP)
  ./.)

Person is  ['John']
GPE is  ['America']
Organization []
['playing', 'is']
['playing', 'is']
play
play
stem are equal
None
None
A who question
Person is ['John']
```

Output 1

The Output 2 shown below shows the output for the "Where" question "Where is Tom living":

```
Your question is =
Where is Tom living
Approximate answer is  =
Tom is living in America.
question is ['Where']
Person is  ['Tom']
GPE is  []
Organization is  []
verb ['living', 'is']
Question ['Where']
**********Text processing
starts.**********
Tom is living in America.
[('Tom', 'NNP'), ('is', 'VBZ'), ('living', 'VBG'), ('in', 'IN'),
('America', 'NNP'), ('.', '.')]
nnp is ['Tom', 'America']
nn is []
vbz is ['is']
vbg is ['living']
prp is []
IN is ['in']
JJ is []
(S (PERSON Tom/NNP) is/VBZ living/VBG in/IN (GPE
America/NNP) ./.)

Person is  ['Tom']
GPE is  ['America']
Organization []
['living', 'is']
['living', 'is']
live
live
stem are equal
None
None
A where question
Your answer is
Person is available
Person found
Organization []
GPE is ['America']
```

Output 2

The Output 3 shown below shows the output for the "What" question "What is the capital of france":

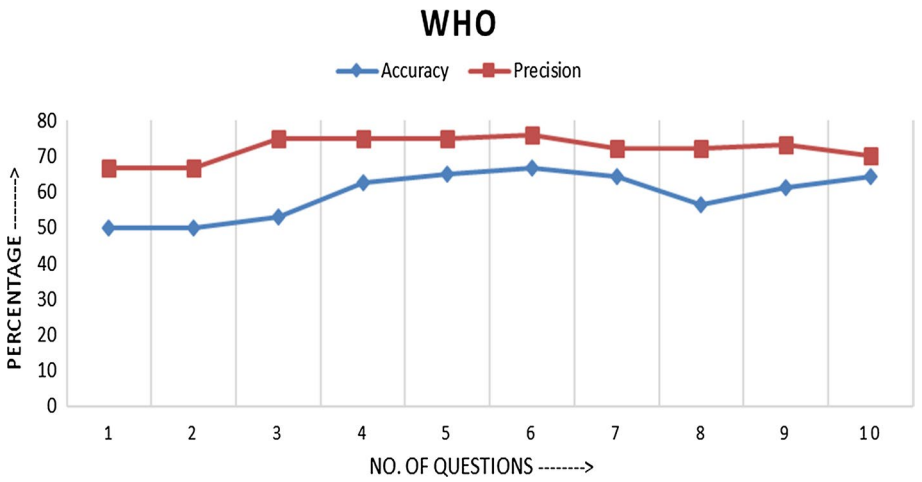
```
Your question is =
What is the capital of france
Approximate answer is  =
Paris is the capital of france.
question is ['What']
Person is  []
GPE is  []
Organization is  []
verb ['is']
Question ['What']
**********Text processing
starts.**********
Paris is the capital of france.
[('Paris', 'NNP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'),
('of', 'IN'), ('france', 'NN'), ('.', '.')]
nnp is ['Paris']
nn is ['capital', 'france']
vbz is ['is']
vbg is []
prp is []
IN is ['of']
JJ is []
(S (GPE Paris/NNP) is/VBZ the/DT capital/NN of/IN
france/NN ./.)

Person is  []
GPE is  ['Paris']
Organization []
['is']
['is']
be
be
stem are equal
None
```

Output 3

9 Comparison and Analysis of Results

This section shows the performance comparison of the proposed system for various question classes as well as its comparison with other works. The work with which the proposed model has been compared is sited in [14] and [15].

The Fig. 2 is to show the Accuracy and Precision (both in %) of the "Wh" question "WHO" for different no. of questions. Here, we have asked some questions (means Question Answering on our work (i.e. NLIIRS_FACTDqa) and noted the observations accordingly). Based on that observation, we have plotted this graph. On Y-Axis, we have taken the percentage (i.e. the accuracy and precision). On X-Axis, we have taken the number of questions



in a manner in which "1" denotes "4" questions and the number of questions increases in the multiple of 4, i.e. "2" denotes "8" questions (i.e. here, we have done our calculations on 8 questions), now the graphs progress when we increases the no. of questions in our work. So, "3" denotes "12" questions, "4" denotes "16" questions, this goes on till "9" denotes "36". Now due to some overloading of questions, "10" denotes "42" questions (i.e. here, we have done our calculations on 42 questions, rather on 40 questions).

The whole scenario is illustrated in the Table 2 below.

Figure 3 is to show the Accuracy and Precision (both in %) of the "Wh" question "WHERE" for different no. of questions. Again some questions were asked (means Q & A on our work (i.e. NLIIRS_FACTDqa) and noted the observations accordingly). Based on that observation, we have plotted this graph. On Y-Axis, we have taken the percentage (i.e. the accuracy and precision). On X-Axis, we have taken the number of questions in a manner in which "1" denotes "0" questions (means no questions was asked at this time), then from now onwards the number of questions increases in the multiple of 4, i.e. "2" denotes "4" questions (i.e. here, we have done our calculations on 4 questions), now the graphs

Table 2 The "WHO" table

Graph (X-axis)	No of questions	Accuracy	Precision
1	4	50	66.7
2	8	50	66.7
3	12	53	75
4	16	62.5	75
5	20	65	75
6	24	66.7	76
7	28	64.28	72
8	32	56.25	72
9	36	61.11	73
10	42	64.28	70

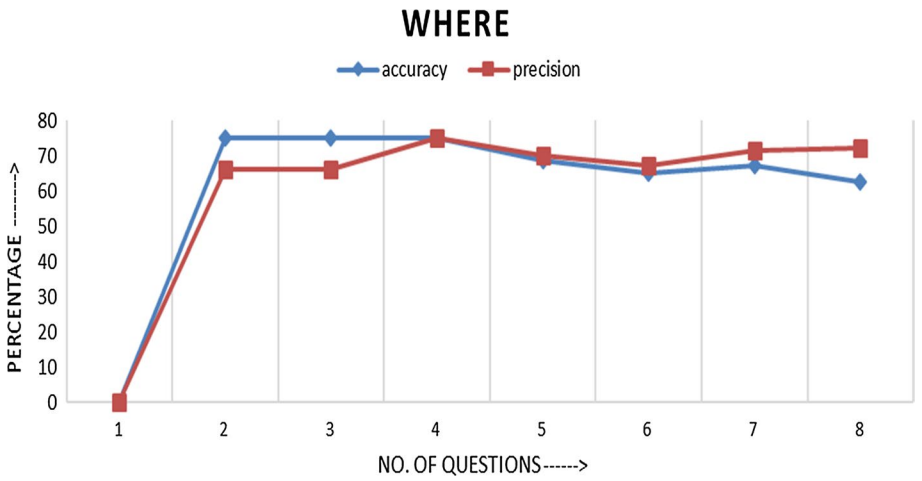


Fig. 3 The "WHERE" graph

progresses when we increases the no. of questions in our work. So, "3" denotes "8" questions. "4" denotes "12" questions, this goes on till "7" denotes "24". But from this point, due to the shape of the graph we have directly plotted the graph for 48 questions, i.e. "8" denotes "48" questions (i.e. here, we have done our calculations on 48 questions).

The whole scenario is illustrated in Table 3 below.

In Fig. 4, on Y-axis, we have the percentage of accuracy or precision and on the X-axis, we do have the no. of questions and in X-axis, "1" denotes "4" no. of questions that have been asked to the system, "2" denotes "8" no. questions (i.e. here, we have done our calculations on 8 questions) and "3" denotes "12" questions and so on which goes up to 32 questions. The results in Fig. 4 are up to 32 questions but later on we asked around 150 no. of questions. With "4" no. of questions, we are getting an accuracy of 75%, but it changes gradually with an increase in the number of questions that have been asked to the system. In the end we got an accuracy of 59.33% and Precision of 50% as shown in the Table 4 in which the whole scenario is illustrated.

Table 5 is a Comparison table which compares our work with other Research papers which are "Factoid Question Answering Over Unstructured and Structured Web Content" done by Microsoft and the other work one is "Question Classification for Medical Domain Question Answering System". This table includes the column namely "Total questions",

Table 3 The "WHERE" table

Graph (X-axis)	No of questions	Accuracy	Precision
1	0	0	0
2	4	75	66
3	8	75	66
4	12	75	75
5	16	68.7	70
6	20	65	67
7	24	67	71.4
8	48	62.5	72

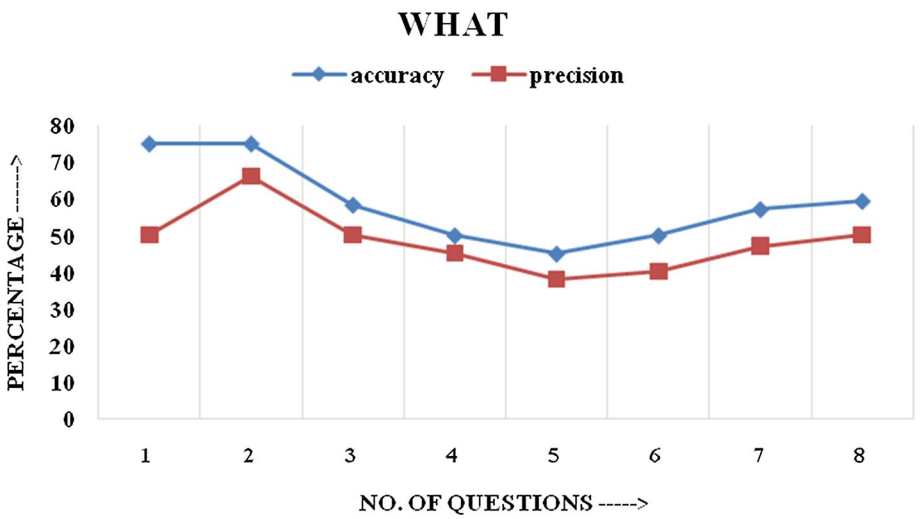


Table 4 The "WHAT" table

No. of questions	Accuracy	Precision
4	75	50
8	75	66
12	58.33	50
16	50	45
20	45	38
24	50	40
28	57.14	47
32	59.33	50

Table 5 Comparison table for "NLIIRS_FACTDqa", "QCMDqa" [14] and "MsoftFQA" [15]

Question type	Total que.	Correct answers	Incorrect answers	Accuracy (%)
Who (NLIIRS_FACTDqa)	51	32	18	62.74
Who (MsoftFQA)	51	6	45	11.80
Where (NLIIRS_FACTDqa)	80	50	30	62.50
Where (MsoftFQA)	40	12	28	30.00
Where (QCMDqa)	100	55	45	55
What (NLIIRS_FACTDqa)	150	89	61	59.33
What (MsoftFQA)	117	14	96	12.00
What (QCMDqa)	150	83	67	55.33
When (NLIIRS_FACTDqa)	54	35	19	64.81
When (MsoftFQA)	54	33	21	61.1
When (QCMDqa)	90	45	45	50

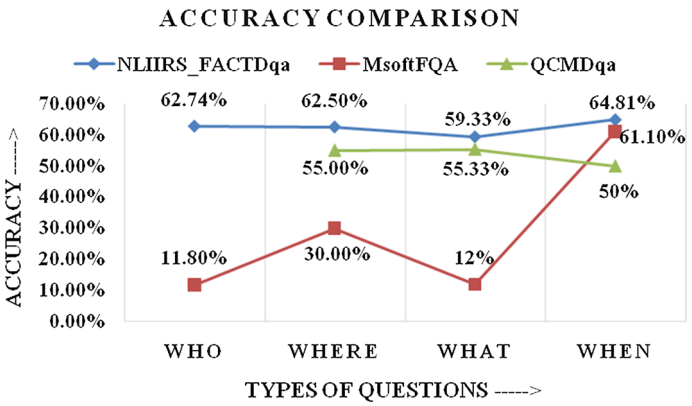


Fig. 5 the accuracy comparison graph

"Correct Answers", "Incorrect Answers" and "Accuracy %". Here, we have asked some questions (Question Answering on our work and noted the observations accordingly) and based on that observation, how many answers were correct and how many were wrong are plotted. Based on the above values, we have calculated the accuracy % (or Correct %) for each question word (like "Who", "Where", "What" and "When"). Then we have filled the table for each row (as shown) i.e. for Who, Where, What and When. Corresponding to each question word, there are three subdivisions, i.e. our work, MsoftFQA and QCMDqa, and corresponding to each row, suitable entry is filled. This table shows that our work is better than that of MsoftFQA and QCMDqa at some occasions.

The reason for the absence of "Who" row in the Table 5 and in the Fig. 5 is that the QCMDqa work hasn't done the work as well as experimentations on the question type "Who".

The Fig. 5 is the Line Chart which shows the comparison of our work with two already submitted Research papers which are "Factoid Question Answering Over Unstructured and Structured Web Content" work of Microsoft (which is abbreviated as MsoftFQA) and "Question Classification for Medical Domain Question Answering System" (which is abbreviated as QCMDqa). In this graph, we have shown the comparison of some "Wh" words like "Who", "Where", "What" and other type of questions like "When". We have made this graph to show where our work is better than "MsoftFQA" and "QCMDqa" and where not. On X-Axis, we have taken the "Types of questions like "Who", "Where", "What" and "When". On Y-Axis, we have taken the Accuracy (in %) which denotes the accuracy of all the types of questions.

Other than the comparisons made above there are a few more aspects or comparison that can be done when compared with other papers of the reference section. The work mentioned in [9] is domain specific and it performs well for history questions but fails for literature questions. In contrary to this our NLIIRS performs well for both type of questions.

10 Limitations

The proposed system is dependent on Stanford's POS tagger and ne_chunker, and as the POS tagger of Stanford University is unable to annotate the "regional" nouns: for example, the Indian names and Indian city names. When used for universal nouns like the defense

entities it works effectively. The system cannot judge the tense of the verb correctly, for example if the input text is, "John was eating apple last night" and the query asked is "Who is eating apple?" then the system will give an answer as "John". As we can see in the input text "John" was not eating the apple at the present moment still the system will give the answer as "John". The proposed system cannot function properly on compound sentences which comprises of "and" and other conjunctions. Last but not the least in the "When" type question the date format like "25-April-2019" is not supported by the system. It means the NLIIRS supports the date format in only numeric form like "25-04-2019", "25/04/2019", or only year like "1980" or "20/05".

11 Conclusion and Future Work

The proposed NLIIRS takes natural language text as input and the novel text processing algorithm, allows the user to retrieve information by asking any "Who", "Where" and "When" type of questions in the form of natural language. The NLIIRS with the help of Stanford's NER, provides a correct factoid answer to these questions. For these classes of question, the NLIIRS provides answers with an accuracy of more than 70% as shown in Sect. 9 above. The system performed well for almost all classes of "wh" questions. The systems performance is not so good for "What" & "When" category of questions. This problem is arising as the answers pertaining to "What" & "When" are generally of narrative type.

At present we have used 7 class NER but the algorithm can support for 3 class NER. In future, we can use 7 Class NER like Time, Money, Percent, Date (in addition to previously included tags like Person, Organization, GPE) to make the system better. The searching category may be extended up to the "Wh" question words like Which, Why and How. We will try to increase the efficiencies of the current "Wh" category questions. We are at present working on our own POS tagger and ne_chunker so that we become capable of annotating the "regional" nouns for any geographic location as stated in the limitations section above. We will improve our work to function with more accuracy on sentences having conjunctions. Work can be done so that the system works efficiently when tenses are encountered, like is, was, have been, etc.

Acknowledgements This publication is an outcome of the R&D work undertaken project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, MeitY, Government of India, being implemented by Digital India Corporation. This research work has been done at Research Project Lab of National Institute of Technology (NIT), Durgapur, India. Financial support was received from Visvesvaraya PhD Scheme, Deity, Govt. of India (Order Number: PHD-MLA/4 (29)/2014_2015 Dated- 27/4/2015) to carry out this research work. The authors would like to thank the Department of Computer Science and Engineering, NIT, Durgapur, for academically supporting this research work. The authors would also like to thank the Department of Computer Science and Engineering, Jaypee University of Engineering & Technology, Guna MP.

References

1. Green, B. F., Chomsky, C., & Laughery, K. (1961) Baseball: An automatic question answerer. In *Proceedings of the western joint computer conference*, New York: Institute of Radio Engineers (pp. 219–224). https://doi.org/10.1145/1460690.1460714.
2. Woods, W.A., & Bolt, B. (1973). Progress in natural language under-standing—An application to lunar geology. In Proceedings of the American Federation of Information Processing So-cieties (AFIPS) (Vol. 42, pp. 441–450). https://doi.org/10.1145/1499586.1499695.

3. Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1994). MASQUE/SQL—An efficient and portable natural language query interface for relational databases. In *Proceedings of the 6th international conference on industrial and engineering applications of artificial intelligence and expert systems* (pp. 327–330). Edinburgh: Gordon and Breach Publisher Inc.
4. Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases—An introduction. In Cambridge University Press 1995. J. Lang. Eng. 1 (1), 29–81, September 1.
5. Yan-hong, F. Hong, Y., Geng, S., & Xun-ran, Y. (2018). Domain named entity recognition method based on skip-gram model. In *First international conference on electronics instrumentation & information systems* (EIIS), China, 22 February 2018. https://doi.org/10.1109/eiis.2017.8298655.
6. Ranjan, P., & Balabantaray, R. C. (2016). Question answering system for factoid based question. In *2nd international conference on contemporary computing and informatics (IC3I), 2016*. https://doi.org/10.1109/ic3i.2016.7917964.
7. Ma, R., Zhang, J., Li, M., Chen, L., & Gao, J. (2018). Hybrid answer selection model for non-factoid question answering. In *2017 international conference on asian language processing (IALP)*, 22 February 2018. https://doi.org/10.1109/ialp.2017.8300620.
8. Yulianti, E., Chen, R.-C., Scholer, F., Croft, W. B., & Sanderson, M. (2018) Document summarization for answering non-factoid queries. In *IEEE transactions on knowledge and data engineering* (Vol. 30, no. 1). https://doi.org/10.1109/tkde.2017.2754373.
9. Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., Daumé, H. III. (2014). A Neural network for factoid question answering over paragraphs. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.3115/v1/d14-1070.
10. Jain, A., & Wasim, F. (2017). Answering SQuAD. In *Department of Computer Science*, Stanford University, Stanford, CA 94305-9020, USA, in 2017.
11. Bian, J., Liu, Y., Agichtein, E., & Zha, H. (2008). Finding the right facts in the crowd: Factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web* Beijing, China, April 21–25, 2008 (pp 467–476). https://doi.org/10.1145/1367497.1367561.
12. Angeli, G., Nayak, N., & Manning, C. D. (2016). Combining natural logic and shallow reasoning for question answering. In *Stanford University*, Stanford, CA 94305, in 2016. https://doi.org/10.18653/v1/p16-1042.
13. Chen, D., Bolton, J., & Manning, C. D. (2016). A thorough examination of the CNN/daily mail reading comprehension task. In *Department of Computer Science Stanford University*, Stanford, CA 94305-9020, USA, 2016. https://doi.org/10.18653/v1/p16-1223.
14. Dodiya, T., & Jain, S. Question classification for medical domain question answering system. In *IEEE international WIE conference on electrical and computer engineering (WIECON-ECE)* 19–21 December 2016, AISSMS, Pune, India. https://doi.org/10.1109/wiecon-ece.2016.8009118.
15. Cucerzan, S., & Agichtein, E. Factoid question answering over unstructured and structured web content. In *Microsoft research, one microsoft way, 2005*.

Partha Sarathy Banerjee had his B.E. and M.Tech degrees in Computer Science and Engineering both from Birla Institute of Technology Mesra, Ranchi. He is also a research fellow for Ph.D. in Computer Science & Engineering under the prestigious Visvesvaraya Ph.D. fellowship of Ministry of Electronics & Information Technology, MeitY, Government of India, being implemented by Digital India Corporation at National Institute of Technology, Durgapur, West Bengal. He started his career with Unit Trust of India Technology Services Limited, currently UTI Infrastructure Technology and Services Limited (UTI-ITSL), Govt. of India, Mumbai. After industry experience he decided to serve in academics and hence he joined the Department of Computer Science & Engineering, at Jaypee University of Engineering & Technology, Madhya Pradesh, (a part of the Jaypee Group) India. He is working in the field of Natural Language Processing. His areas of interest include Natural Language Processing, unstructured data handling using NLP. He has designed Natural Language Information Interpretation and Representation System (NLIIRS) for MeitY Govt.

of India. He has several publications in SCOPUS Indexed journals. He is a life member of Institution of Engineers India (IEI), Institution of Electronics and Telecommunication Engineers India (IETE), Computer Society of India (CSI).

Baisakhi Chakraborty received Ph.D. degree in 2011 from National Institute of Technology, Durgapur, India in Computer Science and Engineering. Her research interest includes knowledge systems, knowledge engineering and management, database systems, data mining, natural language processing and software engineering. She has several research scholars under her guidance. She has more than 30 international publications. She has a decade of industrial and 14 years of academic experience.

Deepak Tripathi is a sixth semester student of B.Tech. Computer Science & Engineering at Jaypee University of Engineering & Technology, Madhya Pradesh, India. He is among the top 1% student of his batch. He was also a summer intern at NIT Durgapur. His area of interest includes algorithm designing and is implementation using PYTHON. He designed the algorithm one and six.

Hardik Gupta is a sixth semester student of B.Tech. Computer Science & Engineering at Jaypee University of Engineering & Technology, Madhya Pradesh, India. He is among the top 1% student of his batch. He is current working in the area of NLP and had his second year summer internship at NIT Durgapur in unstructured data handling. He designed the algorithm two and three.

Sourabh S. Kumar is a seventh semester student of B.Tech. Computer Science & Engineering at Jaypee University of Engineering & Technology, Madhya Pradesh, India. He has job offers from Cognizant Technology and Infosys Limited. He will be joining soon in either of these two organizations. He has designed the various conditions that may arise while extracting information using Q&A in the implementation part.

Affiliations

Partha Sarathy Banerjee[1] · Baisakhi Chakraborty[1] · Deepak Tripathi[2] · Hardik Gupta[2] · Sourabh S. Kumar[2]

Baisakhi Chakraborty
baisakhichak@yahoo.co.in

Deepak Tripathi
deepak.tripathi93721@gmail.com

Hardik Gupta
ramramhdk@gmail.com

Sourabh S. Kumar
ssourabhskumar@gmail.com

[1] Department of Computer Science and Engineering, National Institute of Technology, Durgapur 713209, India

[2] Jaypee University of Engineering and Technology, Guna 473226, India