International Research Journal of Engineering and Management Studies (IRJEMS)
Volume: 03 Issue: 04 | April -2019 ISSN: 1847-9790 || p-ISSN: 2395-0126

Improving the performance of a Watson based QA system

Prabhat Ranjan, Aditya Raj

School of Computer Science, SRM Institute of Science And Technology, Kattankulathur

Abstract: Since the early Question-Answering (QA) system developed in the 1960s, extensive research has been done in this field. Still, it's one of the most researched topic, owing to its vast application. The objective is to build an efficient and intelligent system that can assess and verify the answer generated by the system with a confidence score. We have developed a system using IBM Watson in its core. Using Watson, we are able to work on a large set of structured as well as unstructured documents of various formats. The system based on Watson presents the answer in the form of passages. Although these passages are ranked by relevance, the top-ranked passage may not always be the answer to the question. To get the precise answer for a given question, we have integrated NLP and IR techniques on top of the Watson system. This improved the performance of Watson based QA system. Our system is a unique and advanced system as it combines the best features of tradition (IR technology) as well as advanced technologies (NLP and Machine Learning). We have evaluated the performance of our final system on four different domains (Software Architecture, Naval Weapons, BhagwadGita, and Radar) and achieved Mean Reciprocal Rank (MRR) score of 0.80.

Keywords: Question Answering, NLP, IR, Watson, Knowledge QA system, Hybrid QA system, Answer Extraction

1. INTRODUCTION

Information is power only if we can take action using it. Then, and only then, it represents knowledge and consequently, power. If we can access accurate information when the need arises, then we can take evidence and fact-based decisions. Our evidence/fact-based decisions will depend on the ease of availability and accuracy of the information. In any organization, one of the biggest challenges is to manage the information so that it can be used effectively when the need arises. The problem of data management to a large extent has been resolved courtesy database management technologies. Still a lot can be done to improvise the management and access of information. However, the most challenging task is of immediate and specific extraction of the information when the need arises. The process of information extraction even for a relatively small database is still a major challenge. To address these problems, lots of work have been done in the recent past, various techniques like Natural Language Processing, Information Retrieval, Chatbots, Question-Answering system etc, have been used in various fields to address the issue.

- The Question Answering (QA) system has been one of the most extensively researched topics in recent years. The (QA) system aims at extracting the answer from a collection of documents for any question posed in the natural question and Information Retrieval (IR) techniques to process the database to perform this task, IR system used statistical and probabilistic methods to retrieve the answer from the data-set. The retrieved answer was presented as the “most likely” answer and the QA system had no way of assessing whether the retrieved answer has positively answered the posed question. The answer assessment with a confidence score and verification requires additional intelligence.

- An efficient QA system should assess and verify that the retrieved text has certainly answered the posed question and post verification the answer should be presented to the user. Further, the QA system should present the specific information which was asked in the question and not the chunk of irrelevant text. To meet these objectives, the QA System cannot rely

only on classification and syntactical analysis of the question. It must employ a semantic analysis along with the classification and syntactical analysis of the question and possible answer. Further, the system should assess and verify the answer before it is presented to the user.

Information Retrieval based QA

The IR-based question answering system relies on the information available as a text document in the database or knowledge base. Given a user question, IR based system extract passages from these documents. The passage extraction

is governed by the keywords and entities present in the question. The system processes the question to determine the likely answer type (often a named entity like a person, location, or time) and formulates queries to send to a search engine. The search engine returns ranked documents which are broken up into suitable passages and re-ranked.

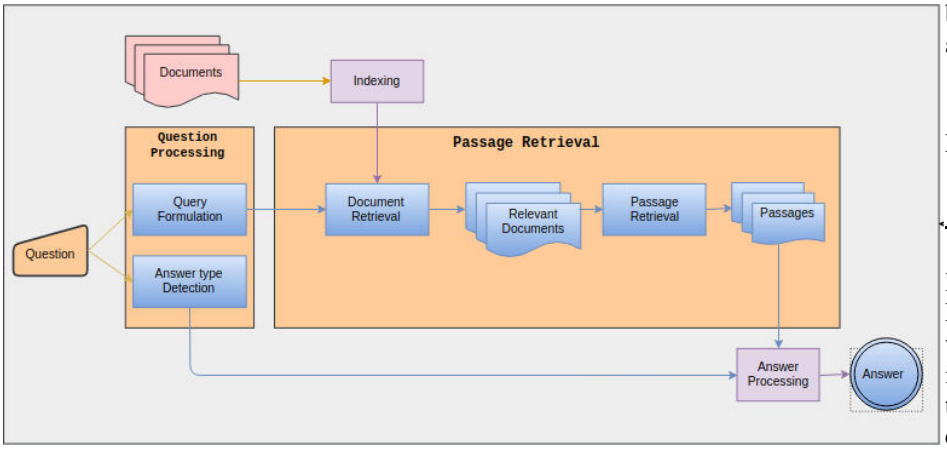

Figure-1

Question Processing

In Question Processing phase, we extract a set of information from the question. The question specifies the keyword which the IR system used for searching the documents [9]. It includes the following steps:

Focus Extraction

In this step, we identify the words in question that should be replaced by the answer. A focus word is usually the first word after the interrogative word. We can identify the focus word using Part-Of-Speech (POS) tagging.

Answer Type Detection/Question Classification

The answer type specifies the type of entity the answer consists of. In this phase, we determine the answer type and establish the entity categorizing the answer.

For example, for questions like,

Who is the head coach of Barcelona?

Expects an answer of entity type PERSON.

Which state has the largest population in India?

Expects an answer of entity type CITY.

who is — was — are — were: PERSON

Where is—was—are—were: LOCATION

In case of supervised machine learning, question classifiers are trained on question sets that are labelled with an answer type. We can use the following features for classification: words in the questions, the part-of-speech of each word, entities present in the questions and semantic information about the words in the questions.

Query Formulation

In this part we create a list of keywords form a query which we sent to an in-formation retrieval system. Using these keywords, the IR system can narrow down the possible answer sets by keyword matching. Keywords can be formed from the terms found in the noun phrases in the question, question Focus Words and using parts of speech.

Passage Retrieval

Before using the information retrieval system to extract the passages from the documents, we first index all the documents in the collection. These indexed documents are used to obtain the relevant passages. The passages are extracted using the following steps:

Extracting Relevant Document

In this step, we use the query terms that we created in the question-processing phase. These query terms can be "Keywords" and "Focus word". The information- retrieval engine search for these query term over a proprietary set of indexed documents. This search yields a set of documents. Although the set of documents is generally ranked by relevance, the top-ranked document may not always be the answer to the question. Therefore, the next step is to extract a set of candidate answer passages from the retrieved set of documents.

Extracting Answer Passages

The relevant set of retrieved documents are segmented into smaller passages, something like paragraphs. We can run a paragraph segmentation algorithm on all the returned documents and treat each paragraph as a segment and first filter out passages in the returned documents that do not contain potential answers.

Ranking the Answer Passages

The answer passages obtained in the previous steps are ranked using the ranking algorithm. We can rank the passages on the basis of following features:

Number of named entities of the right type in the passage.

Number of question keywords in the passage.

Longest exact sequence of question keywords that occurs in the passage.

Rank of the document from which the passage was extracted.

Answer Processing

In the final stage of question answering, we extract a specific answer from the passage to be able to present the user with an answer like 29,029 feet to the question "How tall is Mt. Everest?" We can use the answer type pattern extraction algorithm for these tasks. In the pattern- extraction methods, we use information about the expected answer type together with regular expression patterns.

Question 1: Who is the prime minister of India?

Question 2: How tall is Mt. Everest?

In these examples, the entities are HUMAN and DISTANCE QUANTITY. In the candidate answer passages, we extract potential answers by using these entities or patterns.

Knowledge-based QA System

The knowledge-based QA system computes answers to natural language question posed by the user based on the structured knowledge base. The database is curated in such a way that given the logical form of the question, it will generate the answer by matching the question with the responses in the database. The QA system first transforms or map the input question into its logical form using a semantic parser. Then the answer is retrieved from the structured database using the generated logical form as queries. The Semantic parser in knowledge-based QA maps the natural language question to a logical form, either some version of predicate calculus or a query language like SQL as depicted in the example in the table2.1. The database can be a structured database like

Knowledge-based QA system:

Question	Logical form
When was Subhash Chandra Bose born?	birth-year (Shubhash Chandra Bose,? x)
What states border Gujrat?	λ x.state(x) ∧ borders(x,Gujrat)
What is the largest state in India?	(λ x.state(x), λ x.size(x))
How many people died in 2011 Mumbai terror attack	(count (!FB:event.disaster.survivors) attack

Table-1

RDF triple is a 3-tuple, a predicate with two arguments, expressing some simple relation or-or proposition. The simplest form of the knowledge-based question answering task is to answer factoid questions that ask about one of the missing arguments in a triple. Consider an RDF triple as illustrated in table2.2. This triple can be used to answer text questions like 'When was Subhash Chandra Bose born? ' or 'Who was born in 23 January 1897? '.

RDF triple:

Subject	Predicate	Object
Subhash Chandra Bose	birth-year	1897
Gujrat	states	Madhya Pradesh,Rajasthan
India	largest state	Rajasthan

Table-2

Rule-based Methods

If the relations are frequently used, then hand-written rules can be used to extract relations from the question, as it was illustrated (in subsection Query processing).

Supervised Methods

A supervised data consists of a set of questions paired with their correct logical form like the examples in table. If we have set of supervised data, then we can use these pairs of training tuples and produce a system that maps from new questions to their logical forms. Most supervised algorithms for learning to answer

these simple questions about relations first parse the questions and then align the parse trees to the logical form.

Hybrid QA Systems

The Hybrid QA system integrates the advantage of both Information Retrieval based system as well as the Knowledge-based system. We will discuss the working and architecture of two hybrid systems (Yoda QA and IBM Watson) in subsequent sections.

Yoda QA

It stands for 'Yet another deep answering pipeline'. Yoda QA architecture is based on hybrid architecture. The system is designed as an open domain QA system. The system has been implemented using state-of-the-art methods of information extraction and natural language understanding. The Yoda QA is inspired by the Deep QA architecture of IBM Watson. Developers of Yoda QA system have proposed a model question answering system which can combine structural and non-structural knowledge which is domain-flexible and more liable. The proposed pipeline of the QA system mainly consists of:

Question analysis

Answer production

Answer analysis

Answer merging and scoring

Successive refining of answers

The performance of the QA system has evaluated by the developers on "Open Domain Factoid Question Answers". They have observed that the system and observed that the system could correctly answer one-third of the test Question Answers and over half of the questions have answers in the top five suggested answers.

Watson

The Watson system from IBM that won the Jeopardy challenge in 2011 is an example of a hybrid QA system. It relies on a wide variety of IR and Knowledge-based resources to answer questions. The system runs semantic parsing the way it is done in the case of a knowledge-based system to tag entity and to extract relations. Then like the IR based systems, this system extracts the focus and answer type (also known as lexical answer type) and performs question classification. In a hybrid system, these lexical answer types are again extracted by rules.

Watson is built using the following technologies:

Advanced natural language processing

Information Retrieval

Knowledge Representation

Automated Reasoning

Advanced Machine Learning

To answer a question, Watson first parses and classifies the question. It then looks for clues in the question to determine the answer type. If the clue has more than one statement, it is broken down into sub-clues. It uses a large database of web documents, to further analyse the clues using NLP techniques. After

completion of this step, the system obtains a list of candidate answers. The candidate answers are assigned a confidence score, and the answer with the highest confidence score is presented as a correct answer.

Question Analysis

The first step in the question-answering process is question analysis. In this step, the system attempts to understand what the question is asking and performs the initial analysis that determines how the question will be processed by the rest of the system.

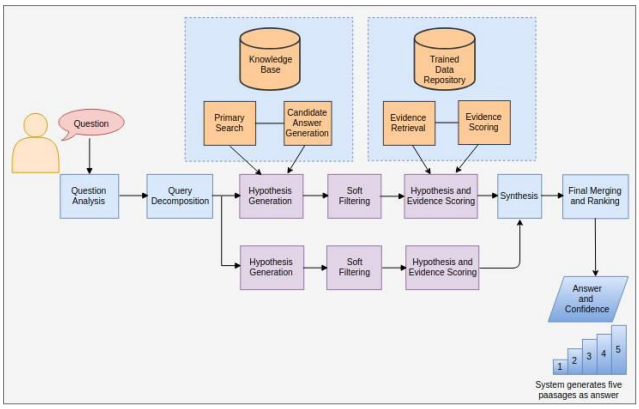

Figure-2

Question Classification

In this step, the system identifies the types or parts of questions that require special processing. Question classification may identify a question as a puzzle question, a math question, a definition question, and so on.

Focus and LAT Detection

The focus is the part of the question that is a reference to the answer. The focus is used by the system to align the question with a potential supporting passage; for proper alignment, the answer in the passage should align with the focus of the question. LATs are terms in the question that indicate what type of entity is asked in the question. Watson uses LATs to determine whether a candidate answer is an instance of the answer types.

Relation Extraction

The role of relation extraction is to detect the entities in the question and to identify the relationship between these entities. Entities can be Person, Organizations, etc. Watson uses relation detection throughout the QA process, from focus and LAT determination to passage and answer scoring. Watson also uses detected relations to query a triple store and directly generate candidate answers from the curated database.

Question Decomposition

In this step, the system recognizes whether questions should be decomposed and to determine how to break them up into sub-questions. Watson uses rule-based deep parsing and statistical classification methods to achieve this step.

Hypothesis Generation

This step takes the results of question analysis and produces candidate answers by searching the system's sources and extracting small answer snippets from the search results. Each candidate answer generated by the system for the posed question is considered as a hypothesis. Watson refers to the search performed in hypothesis generation as "primary search" to distinguish it from the search performed during evidence gathering.

Primary Search

The aim of the primary search is to find the potential answer-bearing content form the system resources based on the results of question analysis.

Candidate Answer Generation

The search results of a primary search are feed into the candidate answer generation module. The task of this module is to generate candidate answer form the search results. If the correct answers are not generated at this stage as a candidate, the system has no hope of answering the question.

Soft Filtering

The candidate answers set generated in the previous steps are pruned down to a smaller set of candidate answers using scoring algorithms. The scoring algorithms compute the likelihood of a candidate answer to be an instance of LAT. This step is known as soft filtering. Candidate answers that pass the soft filtering threshold proceed to hypothesis and evidence scoring, while those candidates that do not pass the filtering threshold are routed directly to the final merging stage.

Evidence Retrieval

To effectively evaluate each candidate answer that passes the soft filter, the system gathers additional supporting evidence. The supporting evidence is routed to the deep evidence scoring components, which evaluate the candidate answers in the context of the supporting evidence.

Scoring

Majority of deep content analysis is performed in this step. Scoring algorithms determine the degree of confidence that retrieved evidence supports the candidate answers. The DeepQA framework supports and encourages the inclusion of many different components, or scorers, that consider different dimensions of the evidence and produce a score that corresponds to how well evidence supports a candidate answer for a given question.

Final Merging and Ranking

The goal of final ranking and merging is to evaluate the hundreds of hypothesis based on potentially hundreds of scores to identify the single best-supported hypothesis given the evidence and to estimate its confidence-'the likelihood it is correct'.

Ranking and Confidence Estimation

After merging, the system ranks the hypotheses and estimate confidence based on their merged scores. Watson works on a machine-learning approach that requires running the system over a set of training questions with known answers and training a model based on the scores. Ranking and confidence estimation is performed in two separate phases in Watson. In both the phases sets of scores are grouped according to the domain (for example type matching, passage scoring, and so on) and intermediate models are trained using ground truths and methods specific for that task.

Word Embedding

Word embedding allows individual words to be represented as real-valued vectors in a predefined vector space. In this representation for text, words that have the same meaning have similar representation. The important feature of word embedding is that it preserves the contextual similarity of words.

Need of word Embedding

Strings or plain text in their raw form cannot be processed by many Machine Learning algorithms and almost all Deep Learning algorithms. They require numbers as inputs to perform any type of text processing, be it a classification, regression, etc.

Simple vector representation of words - One hot encoding

Consider the sentence: "The field of Question answering system is very challenging". Words in this sentence are "question" or "answer" or "challenging" etc. We can create a dictionary which contained a list of all unique words in the sentence. So, a dictionary may look like ['The', 'filed', 'of', 'question', 'answering', 'system', 'is', 'very', 'challenging'] We can represent the words in sentences using a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else [16]. The vector representation of "challenging" in this format according to the above dictionary is [0, 0, 0, 0, 0, 0, 0, 0, 1] and of "field " is [0, 1, 0, 0, 0, 0, 0, 0, 0].

Count Vector

Count vector model develops a vocabulary from all of the documents, then using this vocabulary, models each document by counting the number of times each word appears in that document. For example, consider we have D documents and W is the number of distinct words in our vocabulary then the size of the count vector matrix will be given by D*W. Consider the following two sentences; we can assume that these two sentences represent two documents:

Document 1: "India is a great country"

Document 2: "We are very proud of our country"

From these two documents, we can construct our vocabulary as follows. Vocabulary consists of distinct words of two documents: India, is, great, country, we, are, very, proud, of, our, country. Here D=2, W=10. Now, we count the number of times each word occurs in each document. In document 1, "India", "is", "great", and "country" each word appear once, so the feature vector for documents is: India, is, great, country, we, are, very, proud, of, our so the count vector matrix is:-

	India	is	great	country	we	are	very	proud	of	our
Dcoument 1	1	1	1	1	0	0	0	0	0	0
Document 2	0	0	0	1	1	1	1	1	1	1

Table-3

A column can also be understood as a word vector for the corresponding word in the matrix M. For example, the word vector for 'country' in the above matrix is [1,1] and so on. Here, the rows correspond to the documents in the corpus and the columns correspond to the tokens in the dictionary.

TF-IDF Vectorization

TF-IDF vectorization method takes into account not just the occurrence of a word in a single document but in the entire corpus. In large text documents, some words will frequently be present (e.g. "the", "a", "is", "are", "and", "am"). These words carry very little meaningful information about the actual contents of the document. We assigned a weight to the count features using the TF-IDF transformation.

For example, document A on "programming" is going to contain more occurrences of the word "programming" in comparison to other documents. But common words like "the", "is", "are", etc. are also going to be present in higher frequency in almost every document. For text processing, we must down weight the common words occurring in almost all documents and give more importance to words that appear in a subset of documents.

How TF-IDF works?

Term Frequency: Term Frequency calculates the number of times each word appeared a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than in the shorter documents. Thus, the term frequency is often divided by the document length as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

Inverse Document Frequency (IDF): Inverse Document Frequency estimates the rarity of a word in the whole document collection. While computing Term Frequency, all terms are considered equally important. However, certain frequent fewer importance terms need to be down-weighted and rare meaningful words are given higher weight. The IDF is calculated using the formula:

IDF(t) = log(Total number of documents / Number of documents with term t in it).

Let's consider we have two documents one document on the topic 'programming' and others on the topic of 'security'. Let's assume that table 2.3 and table 2.4 gives the count of terms(tokens or words) in document 1 and document 2 respectively.

Term	Count	Term	Count
This	1	This	1
article	1	article	1
is	1	is	1
about	1	about	1
Programming	1	TF-IDF	1

Table-4

So, using the table information,

TF(article, Document 1) =1/10

TF(article, Document 2) =1/8

IDF(article) = log(2/2)=0

As the word 'article' appears in both the document and the total no of documents are two.

Let us compute IDF for the word 'programming'.

IDF(Programming) = log(4/2) = 0.301

Now, let us compare the TF-IDF for a common word 'article' and a word 'programming' which seems to be of relevance to Document 1.

TF-IDF(article,Document1) = (1/10) * (0) = 0

TF-IDF(article, Document2) = (1/8) * (0) = 0

TF-IDF(Messi, Document1) = (4/10) * 0.301 = 0.12

Hence, the TF-IDF method heavily penalizes the word 'article' but assigns greater weight to 'programming'. So, this may be understood as 'programming' is an important word for Document 1 from the context of the entire corpus.

Prediction based Embedding-Word2vec

Prediction based word embedding models try to predict a word from its neighbours. Word2vec is a computationally efficient predictive model for word embedding. These models are trained to reconstruct linguistic contexts of words. Given an input text, Word2vec produces a vector space, typically of several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector representation. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located near one another in the space. It first constructs a vocabulary from the training text data and then learns the vector representation of words. Word2vec is a combination of two techniques - CBOW (Continuous bag of words) and Skip-gram model. Both of these are shallow neural networks which map words to the target variable which is also a word. Both of these techniques learn weights which act as word vector representations.

CBOW (Continuous Bag of Words)

CBOW algorithm predicts the probability of a word given a context. For example, consider a sentence: "The cat jumped over the puddle." So one approach is to treat "The", "cat", 'over", "the', "puddle" as a context and from these words predict or generate the centre word "jumped". This type of model is known as CBOW model.

Skip Gram

In Skip gram algorithm, the aim is to predict the context given the word. Skip gram is the opposite of CBOW. Skip gram is much slower than CBOW but considered more accurate with infrequent words.

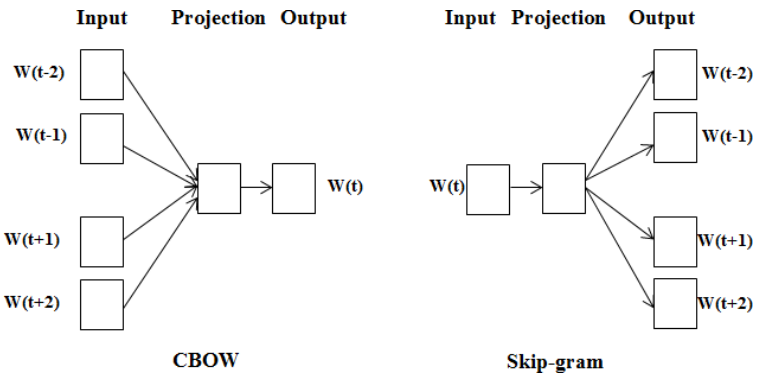

Figure-3

Pre-trained word vectors

Pre-trained vector models are used for text processing. Google offers its pre-trained word2vec model. It contains word vectors for a vocabulary of 3 million words trained on around 100 billion words from the Google news dataset.

Word Mover's Distance

Word Mover's Distance (WMD) is a method that allows us to assess the "distance" between two documents in a meaningful way, even when they have no words in common. The WMD distance measures the similarity between two text documents or passages or sentences as the minimum amount of distance that the embedded words of one document need to travel to reach the embedded words of another document. WMD compares two documents or sentences, by capturing the semantic meanings of the words. It is more powerful than BOW model as it captures the meaning similarities; it is more powerful than the cosine

distance between average word vectors, as the transfer of meaning using words from one document to another is considered.

WMD is illustrated below in figure 2.8for two very similar sentences.

sentence 1: Modi speaks to the media in Jakarta.

sentence 2: The Prime Minister greets the press in Indonesia.

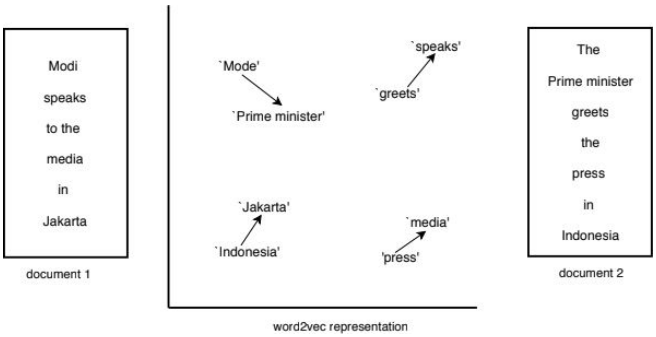

Figure-4

The sentences have no words in common, but by matching the relevant words, WMD is able to accurately measure the similarity between the two sentences. The intuition behind the method is that we find the minimum "travelling distance" between documents, in other words, the most efficient way to "move" the distribution of document 1 to the distribution of document 2.

Cosine Similarity

Cosine Similarity is used as a measure to find the similarity between two vectors. The cosine similarity between two vectors (or two documents on the Vector Space) calculates the cosine of the angle between them. Using cosine similarity is a good way to rank the document by finding which document is closer to the user query.

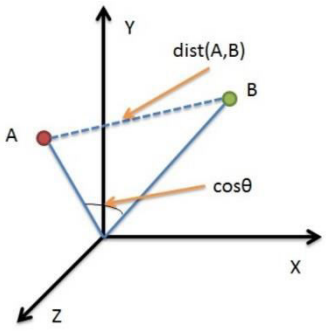

Figure-5

Given two vectors of attributes, A and B, the cosine similarity, cos θ, is represented using a dot product and magnitude as

$A \cdot B = "A""B" \cos \theta$

$\cos \theta = (A \cdot B)/("A""B")$

Euclidean Distance

Euclidean distance is the distance between two points in any number of dimensions. It is the square root of the sum of the squares of the differences between the respective coordinates in each of the dimensions.

If p = (p1, p2, ..., pn) and q = (q1, q2, ..., qn) are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula:

$$d(p, q) = d(q, p) = \sqrt{(q1 - p1)2 + (q2 - p2)2 + \cdots + (qn - pn)2}, .\Sigma n$$

Our Work

In this part, we will discuss in detail, the various approach that we used to develop the QA system. We will first explore the idea of using sentence matching or sentence similarity as a technique to answer fact-based questions for a single document. We will discuss the working and architecture of QA systems that can extract answers from a set of documents and can answer any question: factoid or non-factoid.

A fact-based question expects a straightforward fact-based answer in response. The answer to fact-based questions is normally a single line sentence containing the fact asked in the question. For example:

Question 1: Who is the President of India?

Question 2: Who is the CEO of Google?

The answer to fact-based questions is normally a single line sentence containing the fact asked in the question. For example:

Answer 1: Ram Nath Kovind took office as the 14th President of India.

Answer 2: Sundar Pichai, the CEO of Google is an Indian American business executive.

This motivated us to explore the idea of implementing sentence similarity to answer the fact-based question. Our idea was that based on a question posed by the user, our QA system should search for the correct sentences which contain the relevant information. To achieve this, we studied the different representation of word to represent the sentences in the vector space, viz bag-of-words, word2vec and doc2vec. We developed three models by integrating the word embedding, TF-IDF, LSI and NLP techniques. The working and architecture of these models are discussed below:

First QA Model

This first model was developed using the vector representations of words (bag of words), vector transformation (TF-IDF, LSI) and using NLP techniques.

Text Document Preprocessing

We created a document on the topic India's Nuclear program using the Wikipedia content. We split the document and converted into a list of sentences. The list of sentences was enhanced using various NLTK packages, we converted the text into tokens and removed the stop words. After that, we extracted all the distinct words from and stored them in a dictionary. The words in the dictionary were assigned a unique integer ID. This dictionary is used for transformation of sentences into a vector.

Text Vectorization - Bag of Word Representation

To transform the sentences into vector form we use the gensim functionality 'dictionary.doc2bow()'. This function of gensim, simply takes the reference of the dictionary and counts the number of occurrences of each distinct word of the sentence and converts the word to its integer word-id and returns the result as bag-of-words as a sparse vector, in the form of [(word-id, word-count), ...]

This vector representation of sentences was further transformed into precise and meaningful representation using TF-IDF and LSI transformation models. First, we use the gensim TF-IDF transformation model to convert bag-of-words integer counts into TF-IDF real-valued weights. Latent semantic indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured

collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings.

Answers Extraction

The question entered by the user is also transformed into vector form using the same gensim functionality that we used to transform the sentences into vectors. Now our question and sentences are transformed into vector form. To find the answer to a question, we calculated the similarity between the question and the sentences in the vector space using cosine similarity. Two vectors with the same orientation have a cosine similarity of 1. The output of cosine similarity produces a list of sentence indices and their similarity scores w.r.t the question. We sort the indices in decreasing order of their similarity score and present the top five sentences as the possible answers for a given question.

Second QA Model

In the previous model, we transformed the texts into vector representation using the bag-of-word technique. In this model, we use the open source pre-trained vectors of Google data. This model is developed using the pre-trained word2vec vectors, vector averaging, vector transformation and NLP technique.

Text Document Preprocessing

We use the same document created in the previous model (India's Nuclear pro- gram). We split the document and converted it into a list of sentences. The list of sentences was then enhanced by tokenizing and removing the stop words.

Text Vectorization

In this model, we transformed the text into vector representation by using the google's pre-trained vectors. This vector was again transformed using gensim TF-IDF transformation model to get TF-IDF real-valued weights. After that, we calculated the centroid of the vectors of words present in a sentence to get a single vector representation for a sentence. Centroid of a sentence is the sum of the embedding of the tokens divided by the number of tokens in the sentence. After this step, we get TF- IDF weighted centroids of word embeddings.

Answers Extraction

Given a question, we transformed the question into vector form and calculated the centroid of the question and compared the cosine similarity between the question and list of sentences using cosine similarity. We sort the results of cosine similarity in decreasing order of their similarity score and present the top five sentences as the possible answers to a given question.

Third QA Model

It uses word2vec vector embeddings of words. WMD can accurately measure the similarity between the two sentences or similarity between a sentence(question) against a group of sentences (answer sets).
In this model, we use the feature of Word Mover's Distance (WMD) to find the answer for a given query.

Text Document Preprocessing

We use the same document created for the previous models. We split the document and converted it into a list of sentences. The list of sentences is then enhanced by tokenizing and removing the stop words.

Text Vectorization

Since WMD works on vector representation of words, we use the pre-trained vectors of Google to transform the text into vectors.

Answers Extraction

We then calculate the similarity between the question and corpus of sentences using the WMD similarity function. WMD calculates the similarity between sentences and question using Euclidean distance. The output of WMD similarity yields

the distance score of the sentences w.r.t the question. We sort the most similar sentences in increasing order of their distance and present the top five sentences as the possible answers for a given question.

QA System using IR and NLP techniques

We developed this QA system using IR and NLP techniques. The specific technologies and framework that we have used to implement the system are discussed in this section.

Question Processing

In this step, we process the question to understand what exactly the question is asking. We have used 'Textblob' to process the question. Textblob is a python library that provides an API for text processing. In our system, we have used Textblob for 'Focus Extraction' of the question. Further, we created a list of query terms by removing all the stop words from the question.

Document Retrieval

To extract relevant documents, we use the list of query words created in the previous step. The IR search engine looks for these query terms over a set of indexed documents. We have done indexing on the documents and saved on disk. For a given query, our system loads the index file for document retrieval. The indexing minimizes the latency and narrows down the search.

Top relevant documents are extracted using the Okapi BM 25 model. To use this model effectively, we removed all the stop words from the documents. We implemented a BM25 algorithm using the standard Okapi BM25 formula.

Passage Retrieval

For passage extraction, we selected the top five relevant documents. These five documents are segmented into smaller passages. We have used two different methods to extract the passages: the first one using Okapi BM25, the second one using word embedding.

BM25 Model: Our first passage extractor model was built using BM25. To retrieve the passages, we scored all the passages in the document using BM25 model. Then the score's are index-sorted using the argsort function in python library. The top ten ranked passages are retrieved, and they form a set of candidate answer passages.

Word Embedding: We built our second passage extractor model using word embedding. We first convert the query and the passages into vector form. For vector transformation, we used TF-IDF as a feature. After that, we computed cosine similarity between the query vector and passage vector. Then the score's are index-sorted using the argsort function in python library. The top ten ranked passages are retrieved, and they form a set of candidate answer passages.

Answer Passage Extraction

In the previous step, we extracted ten relevant passages. To extract the answer passage, we re-ranked these passages. We implemented a ranking algorithm that includes both BM25 and WMD with appropriate weight. The output of this step is a single answer passage that can be processed to extract the answer.

Answer processing

To process the answer, we need to classify the questions. The questions are classified as follows:

Who: Person
Where: Location and place
When: Date and time
What: Descriptive, seeks explanation and definition.
Why: Descriptive, seeks reason.
How: Descriptive, seeks solutions.

The 'Wh questions' like who, where and when seeks one or two sentences as an answer. To answer these questions, we first tokenized the answer passage extracted in the previous step into sentences. After that, we computed the distance between the question and the sentences using WMD. The sentences are then sorted, and top-ranked sentences are

presented as the answer to the question. The Wh questions words like who, where and when are replaced from the question and appended with a person, location and date respectively to improve the performance of similarity function.
The descriptive questions like what, why and how seeks a descriptive answer. These questions are answered by filtering the answer passage. Using the similarity score between the question and sentences, we filter the answer passage.

QA System using IBM Watson APIs

We develop this QA system using the IBM Watson APIs. The specific technologies and framework that we have used to implement the system are discussed in this section.

Environment Creation

We created our knowledge base of documents in the Watson cloud platform. To store our documents in the Watson cloud, we created an environment using the POST method. An environment in a cloud is like a warehouse where we can store our collection of documents. The Watson REST API used to create the environment is [6]:

curl -X POST -u "{username}":"{password}" -H "Content-Type: application /json"-d '{ "name":environment_name", "description":" optional"}' "https://gateway.watsonplatform.net/discovery/api/v1/environments? version=2017-11-07"

'Replace username and password with the service credentials'

We are required to check the environment status periodically until we see an active status. When the status of the environment becomes active then only we can use the cloud environment. The API used to check the environment status is [6]:

curl -u "{username}":"{password}"https://gateway.watsonplatform.net/ discovery/api/v1/environments/{environment_id}?version=2017-11-07

Creating a Collection

Once the environment is active, we can use that for storing our document. Before storing the document, we have to create a collection in the environment. A collection is like a container where we store our documents in the cloud environment.
The Watson REST API used to create the collection is:
curl -u "{username}":"{password}" https://gateway.watsonplatform.net/discovery/api/v1/environments/{environment_id}/configurations?version=2017-11-07
The API returns information such as our collection ID, collection status, and how much storage our collection is using. Check the collection status periodically untilwe see an active status for our collection.
The API used to check collection status is:
curl -u "{username}":"{password}"https://gateway.watsonplatform.net/discovery/api/v1/environments/{environment_ id}/collections/{collection_id}?version=2017-11-07

Configuration

When we upload our documents to the Watson cloud platform, the Watson services convert and enrich the documents. These enrichments are necessary to extract the important features of the document like entities, keywords, semantic, relation, sentiment, etc. To enrich the documents, we have to configure the enrichment. These enrichments add cognitive metadata to the documents. To enrich the documents, we have to configure the enrichment. There are a total of nine Watson enrichment available; which are used for document configuration

1. Entity Extraction
2. Relation Extraction
3. Keyword Extraction
4. Category Classification
5. Concept Tagging
6. Semantic Role Extraction
7. Sentiment Analysis
8. Emotion Analysis
9. Element Classification,

Entity Extraction

Entity Extraction returns items such as persons, places, and organizations that are present in the input text. Entity extraction adds semantic knowledge to content to help understand the subject and context of the text that is being analysed. Watson entity extraction techniques are based on sophisticated statistical algorithms and natural language processing technology.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Entities:
"text": "IBM Watson" "relevance": 0.98389 "type": "Company",

"text": "Kanpur" "relevance": 0.532754 "type": "Location",

"text": "India" "relevance": 0.469643 "type": "Location",

Relation Extraction

The Relation extraction recognizes when two entities are related and identifies the type of relationship between the entities. The relation between entities is scored in the ranges from 0.0 to 1.0. The higher the score, the more relevant the relation.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Relations:
"type": "located At" "score": 0.989245,
"Entities":
"type": "GeopoliticalEntity" "text": "Kanpur",
"type": "GeopoliticalEntity", "text": "India"

Keyword Extraction

Keywords are those important topics or words in our content that are typically used when indexing data, generating tags, or when searching. The Watson service automatically identifies the keyword from the input content and then ranks the keywords in that content. The relevance score ranges from 0.0 to 1.0. The higher the score, the more relevant the keyword.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Keywords:
"text": "IBM Watson" "relevance": 0.985203,
"text": "new office" "relevance": 0.821033
"text": "software developers" "relevance": 0.66497,
"text": "Kanpur" "relevance": 0.307723,
"text": "India" "relevance": 0.306485

Category Classification

It categorizes input text into a hierarchical taxonomy up to five levels deep. Deeper levels allow us to classify content into more accurate and useful sub-segments. The categories are scored in the range from 0.0 to 1.0. The higher the score, the greater the confidence in that category
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Categories:
"score": 0.361614
"label": "/business and industrial",
"score": 0.329377
"label": "/business and industrial/company/merger and acquisition",
"score": 0.154254
"label": "/business and industrial/business operations/ business plans"

Concept Tagging

It identifies concepts with which the input text is associated, based on other concepts and entities that are present in that text. Concept tagging understands how concepts relate and can identify concepts that are not directly referenced in the text. Concept tagging enables higher level analysis of input content than just basic keyword identification.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
concepts:
"text": "IBM Watson", "relevance": 0.91136,
"text": "office", "relevance": 0.886784,

Semantic Role Extraction

It identifies the subject, action, and object relations within the sentences in the input content. Relation information can be used to automatically identify key events, key facts, and other important actions. The subject, action, and object are extracted for every sentence that contains a relation.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Semantic-roles: "subject":
"text": "The software developers",
"object":
"text": "were pleased that IBM Watson plans to build a new office in Kanpur, India",
"action":
"verb":
"text": "were",
"tense": "past"

Sentiment Analysis

The purpose of sentiment analysis is to identifies attitude, opinions, or feelings in the content that is being analyzed. The Watson service can calculate overall sentiment within a document, passages, sentence or even of a single word.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Sentiment:
"score": 0.759813,
"label": "positive

Emotion Analysis

Emotion analysis detects anger, disgust, fear, joy, and sadness implied in our text. Emotion Analysis can detect emotions that are associated with targeted phrases, entities, or keywords, or it can analyze the overall emotional tone of our content.
Input text: The software developers were pleased that IBM Watson plans to build a new office in Kanpur, India.
Emotion:
"disgust": 0.002578,
"joy" : 0.726655,
"anger" : 0.02303,
"fear" : 0.018884,
"sadness": 0.016802

Element classification

It parses elements in our documents to classify important types and categories. Element Classification makes it possible to rapidly parse through governing documents to convert, identify, and classify elements of importance. Using state of the art Natural Language Processing, the party (who it refers to), nature (a type of element), and category (specific class) are extracted from elements of a document.

Query and Search

Once our documents are uploaded and enriched by the Watson APIs, we are ready to start asking the question to the system. Using the API, we can ask the question in the natural language form; the system will then try to understand and process and classify the question.
We can question the system using the API mentioned below:
curl -u "{username}":"{password}" "https://gateway.watsonplatform.net/discovery/api/v1/environments/{environment_id}/collections/{collection_id}/query?version=2018-03-05&query=relations.action.lemmatized:acquire&count=15&filter=entities.text:Watson service &return=text"

Answer Passages

When we enter the question, the system searches the entire content to produce an answer. When creating a question, we can be as vague or as specific as we want. The more specific the question, the more targeted the results. Watson presents the answer in the form of passages. By default, five passages of around 400 characters each will be returned for a query.

Passages

The system returns a set of the most relevant passages from the documents generated against the question. The passages are generated by sophisticated Watson algorithms to determine the best passages of text from all of the documents returned by the query. Discovery attempts to return passages that start at the beginning of a sentence and stop at the end using sentence boundary detection. To do so, it first searches for passages approximately the length specified in the 'passages.character' parameter (default 400). It then expands each passage to the limit of twice the specified length to return full sentences.

Passages Count

This parameter defines the maximum number of passages the system has to return. The system will search and present the answer in the form of passages. We can define the number of passages we want the system to return by using the 'passages.count' parameter.

Passages Characters Count

This parameter defines an approximate number of characters that any one passage should contain. The default is 400. The minimum is 50. The maximum is 2000. Passages returned may be up to twice the requested length to get them to begin and end at sentence boundaries.

Training of QA system

We can train the system to improve the response of the system. To train the system, we have to provide the training data. When we provide the training data, the system uses machine-learning techniques to find signals in our content and questions. The service then reorders question results to display the most relevant results at the top. As we add more training data, the service instance becomes more accurate and sophisticated in the ordering of results it returns.
Training must meet the following minimum requirements for Watson to begin applying our ratings:
We must train a minimum of 49 queries, and possibly more.
Watson will give us feedback if it needs more queries to train.
We should apply both available ratings to our results: Relevant and Not relevant. Only rating the Relevant documents will not provide the data needed.
The training queries must include some term overlap between the question and the desired answer so it can be retrieved by the Watson service's initial search, which is broad in scope. We can train the system either using the API or using the system tooling. Training the system using system tooling is easier. In system tooling, we have to enter the query and rate the answers generated by the system as relevant or non-relevant. Watson assigns a weight of 0 to the answers marked as non-relevant and a weight of 10 to the answers marked as relevant.

Confidence Scores

Trained collections will return a confidence score as the result of a natural language query. This confidence number is calculated based on how relevant the result is estimated to be, compared to the trained model.

Final Integrated QA system

The QA system using IBM Watson APIs presents five passages as a response to the input question as illustrated in the figure3.3. Although these passages are ranked by relevance, the top-ranked passage may not always be the answer to the question. The answer may be contained in the lesser-ranked passages also. We extracted the top ten ranked passages from the Watson QA system. After that, we implemented the ranking algorithm to re-rank these passages to get the answer passage. Once we get the answer passage, we extracted the answer from this passage. To extract the answer passage accurately, we implemented a ranking function on top of ten Watson passages.
Okapi BM 25

We first used the BM25 ranking function to extract the answer passage. The Watson passages were re-ranked according to their relevance w.r.t to question using the BM25 algorithm. This method delivered positive results but failed on some occasions as BM25 does not preserve the semantic relation. To analyse the performance of the BM25 algorithm, We formed a set of 20 questions whose answers were not present in the first passage. We then tried to re-rank these passages with BM25. From the set of 20 question, we got the answer in first passage for 16 question.

Word Mover Distance (WMD)

WMD calculates the distance between two vectors. We used WMD to calculate the distance between question and five Watson passages. These passages were re- ranked according to their distance with the question in increasing order. From the set of 20 question, we got the answer in first passage for 17 question.

Combining WMD with BM25

Both WMD and BM25 are efficient ranking algorithms but have limitations also. BM25 fails when trying to identify synonyms and discards semantic relation. On the other hand, WMD preserves semantic relation but fails when the passage contains repetitive question terms. To overcome their limitations and to incorporate best features of both, we combined both the ranking algorithm in a single algorithm. We assigned appropriate weight-age to each of them in our final ranking algorithm. This hybrid ranking algorithm delivered excellent results when analysed with a set of various question and Watson passages. The architecture of our ranking algorithm is illustrated in figure.

Integration of User Feedback with Database

The user can give his feedback by rating the answer generated by the system as relevant or non-relevant. We have integrated our system server with the MySQL database, to store the user feedback. When a user rates an answer, the information like, user question, system answer and user feedback are stored in the database. The feedback from the user provides the system administrator a more profound insight about the activities happening on the client side as well as behaviors of the system. This information (user feedback) stored in the database can be used to train the system or update the corpus. This, in turn, enhances the performance of the system.

Results:

We have developed three QA models using word embedding, TF-IDF, WMD and NLP techniques. To asses the performance of these model we created a document on the topic of India's Nuclear program using the Wikipedia content. We used a set of 50 question that we entered into these QA models and evaluated their performance. the table illustrates the comparison between these models. The first column of the table represents the type of the technologies used in the QA models. Column second to sixth represents the number of questions in which the correct answer was obtained at rank 1, rank 2, rank 3, rank 4 and rank five respectively. The seventh column represents the number of questions for which the system was not able to generate the correct answer in top 5 candidate answers.

Analysis of QA Models						
QA Models	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Incorrect Response
Bag-of-words, TF-IDF, LSI and co-sine similarity	11	7	5	2	1	24
Word2vec,TF-IDF,Centroidand cosine similarity	13	8	6	2	2	19
Word2vec,WMD and Euclidean dis-tance	14	9	6	3	1	17

Table-5

Rank (MRR) for each system. The MRR is a statistical measure for evaluating the performance of the systems that produce a list of possible responses to a sample of queries, ordered by probability of correctness. Column one of the table represents the type of technology used in the models. The second column contains the value of Mean Reciprocal Rank (MRR) for each model.

Analysis of QA Models	
QA Models	Mean Reciprocal Rank
Bag-of-words, TF-IDF, LSI	0.34
Word2vec,TF-IDF,Centroid	0.39
Word2vec, WMD	0.43

Table-6

Analysis of the First Model

The Question Answering model developed using Bag-of-words, TF-IDF, LSI and cosine similarity responded with a list of 5 possible answers, for a given question. We asked 50 questions to the model; the questions were based on the content of our document.

Remarks
The system was able to generate the correct answer in the top 5 responses for 52% of the total questions asked.
The system produced the correct answer at 'Rank1' for 22% of total questions.
The Mean Reciprocal Rank of the system is 0.34.
The system was not able to provide the correct answer for 24 questions out 50 questions.

Analysis of Second Models

The Second model developed using word2vec, TF-IDF, centroid and cosine similarity responded with a list of 5 possible answers, for a given question. We asked 50 questions to the model; the questions were based on the content of our document.

Remarks
The system was able to generate the correct answer in the top 5 responses for 61% of the total questions asked.
The system produced the correct answer at 'Rank1' for 26% of total questions.
The Mean Reciprocal Rank of the system is 0.39.
The system was not able to provide the correct answer for 19 questions out 50 questions.

Analysis of the Third Model

The Question Answering model developed using word2vec, WMD and Euclidean distance responded with a list of 5 possible answers, for a given question. We asked 50 question to the model; the questions were based on the content of our document.

Remarks
The system was able to generate the correct answer in the top 5 responses for 66% of the total questions asked.
The system produced the correct answer at 'Rank1' for 28% of total questions.
The Mean Reciprocal Rank of the system is 0.43.
The system was not able to provide the correct answer for 17 questions out 50 questions.

Analysis of QA system developed using IR and NLP techniques

We tested the performance of the QA system on the topic of 'Software Architecture'. The system responded with a list of 5 possible answers, for a given question. To asses the performance of the system, we asked 100 questions to the system. We evaluated the system's performance based on its response to these questions.
Table illustrates the response of the QA system on the topic of 'Software Architecture'. Column second to sixth represents the number of questions in which the correct answer was obtained at rank 1, rank 2, rank 3, rank 4 and rank 5

respectively. The seventh column represents the number of questions for which the system was not able to generate the correct answer in top 5 candidate answers.

QA system Analysis - Software Architecture						
Model Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Incorrect Response
QAsystem usingIR andNLP techniques	53	9	7	3	1	27

Table-7

The results of the QA systems was evaluated by calculating the Mean Reciprocal Rank (MRR). The analysis of the system on MRR is illustrated in table

Analysis of QA system - Software Architecture	
QA Models	Mean Reciprocal Rank
IR and NLP based system	0.60

Table-8

Remarks
The system was able to generate the correct answer in top 5 responses for 73% of the total questions asked.
The system produced the correct answer at 'Rank1' for 53% of total questions.
The Mean Reciprocal Rank of the system is 0.60.
The system was not able to provide the correct answer for 27 questions out of 100 questions.

Analysis of QA system developed using IBM Watson APIs

We tested the performance of the QA system built using IBM Watosn on the topic of 'Software Architecture'. The system responded with a list of 5 possible answers, for a given question. We asked 100 questions to the system on each topic. The performance of the system was evaluated based on the system's response to these questions.

Analysis on the topic of Software Architecture

Table illustrates the response of the QA system on the topic of 'Software Architecture'. Column second to sixth represents the number of questions in which the correct answer was obtained at rank 1, rank 2, rank 3, rank 4 and rank 5 respectively. The seventh column represents the number of questions for which the system was not able to generate the correct answer in the top 5 candidate answers.

QA system Analysis - Software Architecture						
Model Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Incorrect Response
QAsystem using Watson APIs	60	13	8	3	2	14

Table-9

The results of the QA systems was evaluated by calculating the Mean Reciprocal Rank (MRR). The analysis of the system on MRR is illustrated in table

Analysis of QA system - Software Architecture	
QA Models	Mean Reciprocal Rank
IBM Watson	0.71

Table-10

Remarks
The system was able to generate the correct answer in top 5 responses for 86% of the total questions asked.
The system produced the correct answer at 'Rank1' for 60% of total questions.
The Mean Reciprocal Rank of the system is 0.71.
The system was not able to provide the correct answer for 14 questions out of 100 questions.

Analysis on the topic of Naval Weapons

The performance of our final QA system was evaluated on the topic of 'Naval Weapons'. Table5.9illustrates the response of the QA system. Column second to sixth represents the number of questions in which the correct answer is obtained at rank 1, rank 2, rank 3, rank 4 and rank 5 respectively. The seventh column represents the number of questions for which the system was not able to generate the correct answer in top 5 candidate answers.

Trained QA system Analysis - Naval Weapons						
Model Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Incorrect Response
QAsystem using Watson APIs	72	13	7	0	1	7

Table-11

The results of the QA systems was evaluated by calculating the Mean Reciprocal Rank (MRR). The analysis of the system on MRR has illustrated in table

Trained QA system Analysis - Naval Weapons	
QA Models	Mean Reciprocal Rank
IBM Watson	0.81

Table-12

Remarks
The system was able to generate a correct answer in top 5 responses for 93 % of the total questions asked.
The system produced the correct answer at 'Rank1' for 72% of total questions.
The Mean Reciprocal Rank for our system is 0.81
The system was not able to provide the correct answer for 07 questions out of 100 question

Conclusion

QA systems are one of the most researched topics due to its vast application in the various field. We have built two different QA system using two different technologies. One using IBM Watson APIs and the other using IR and NLP techniques. Our final modal is built by integrating the advantages of these technologies.The final QA system using IBM Watson in its core is a unique and advance system as it incorporates the best elements of conventional IR,NLP technology and the advanced Machine Learning alogorithms.

We evaluated the performance of QA system built using IR and NLP technologies and achieved an MRR score of 0.60. The performance evaluation of QA system using IBM Watson produced an MRR score of 0.71. Our final model built using integrating IR and NLP techniques on top of Watson delivered an MRR score of 0.80. We analyzed its performance on four different domains (Software Architecture, Naval Weapons, Bhagwadgita, and Radar) and achieved an MRR score of around 0.80.

We have integrated speech interface in our QA system for hand's free operation. We explored the services of IBM, Google, and Microsoft for speech recognition (speech- to-text) and speech (text-to-speech) synthesis. The performance of Google API's has been observed to be better than IBM and Microsoft.

References

[1] Petr Baudiš. "YodaQA: a modular question answering system pipeline". In: *POSTER 2015-19th International Student Conference on Electrical Engineer- ing*. 2015, pp. 1156–1165.

[2] Jason Brownlee.*Word Embedding*. URL:https : / / machinelearningmastery. com/what-are-word-embeddings/.

[3] Google.*Google Speech to Text API*. URL:https://cloud.google.com/speech- to-text/.

[4] IBM.*Passage Retrieval using IBM Watson*. URL:https://console.bluemix. net/docs/services/discovery/query-parameters.html#nlq.

[5] IBM.*Training of QA System*. URL:https://console.bluemix.net/docs/ services/discovery/train- tooling .html#improving - result - relevance - with - the-tooling.

[6] IBM.*Watson APIs*. URL:https://www.ibm.com/watson/services/speech-to- text/.

[7] IBM.*Watson Enrichment*. URL:https://console.bluemix.net/docs/services/ discovery/building.html#configuring-your-service.

[8] Armand Joulin et al. "Bag of tricks for efficient text classification". In:*arXiv preprint arXiv:1607.01759* (2016).

[9]Dan Jurafsky and James H. Martin.*Question Answering*. 2017. URL:https: //web.stanford.edu/~jurafsky/slp3/28.pdf.

[10] Matt Kusner et al. "From word embeddings to document distances". In:*In- ternational Conference on Machine Learning*. 2015, pp. 957–966.

[11] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International Conference on Machine Learning*. 2014, pp. 1188– 119

[12] Chatbot Magazine.*Natural Language Processing*. 2017. URL:https://chatbotsmagazine. com/what-is-the-working-of-a-chatbot-e99e6996f51c.

[13] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[14] Barbara Rosario. "Latent semantic indexing: An overview". In:*Techn. rep. INFOSYS* 240 (2000), pp. 1–16.

[15] TREC.*Question Answering Collections*. URL:https://trec.nist.gov/data/qa. html.

[16] Analytics Vidhya.*Word Embedding using One-hot Encoding*. 2017. URL:https: //www.analyticsvidhya.com/blog/2017/06/word-embeddings-count- word2veec/.

[17] IBM Watson.*IBM Watson Text to Speech.* URL:https:// www. ibm. com/ watson/services/text-to-speech/.

[18] Wikipedia.*QA System History*. URL:https://en.wikipedia.org/wiki/Question answering#History.

[19] wikipedia. *Cosine similarity*. URL:https://en.wikipedia.org/wiki/Cosine similarity. wikipedia.*Cosine similarity*.URL:https://en.wikipedia.org/wiki/Euclidean distance