SEMONTOQA - A Semantic Understanding-Based Ontological Framework for Factoid Question Answering

Md Moinul Hoque
Department of Informatics
University of Evora
Evora, Portugal
moincse@yahoo.com

Paulo Quaresma
Department of Informatics
University of Evora
Evora, Portugal
pq@di.uevora.pt

ABSTRACT

This paper presents an outline of an Ontological and Semantic understanding-based model (SEMONTOQA) for an open-domain factoid Question Answering (QA) system. The outlined model analyses unstructured English natural language texts to a vast extent and represents the inherent contents in an ontological manner. The model locates and extracts useful information from the text for various question types and builds a semantically rich knowledge-base that is capable of answering different categories of factoid questions. The system model converts the unstructured texts into a minimalistic, labelled, directed graph that we call a Syntactic Sentence Graph (SSG). An Automatic Text Interpreter using a set of pre-learnt Text Interpretation Subgraphs and patterns tries to understand the contents of the SSG in a semantic way. The system proposes a new feature and action based Cognitive Entity-Relationship Network designed to extend the text understanding process to an in-depth level. Application of supervised learning allows the system to gradually grow its capability to understand the text in a more fruitful manner. The system incorporates an effective Text Inference Engine which takes the responsibility of inferring the text contents and isolating entities, their features, actions, objects, associated contexts and other properties, required for answering questions. A similar understanding-based question processing module interprets the user's need in a semantic way. An Ontological Mapping Module, with the help of a set of pre-defined strategies designed for different classes of questions, is able to perform a mapping between a question's ontology with the set of ontologies stored in the background knowledge-base. Empirical verification is performed to show the usability of the proposed model. The results achieved show that, this model can be used effectively as a semantic understanding based alternative QA system.

FIRE '14, December 05-07, 2014, Bangalore, India

DOI: http://dx.doi.org/10.1145/2824864.2824886

CCS Concepts

•**Computing methodologies** → **Knowledge representation and reasoning;**

Keywords

Deep question answering system, Text ontology, Syntactic Sentence Graph, Text inference, Semantic mapping, Natural language processing, Text understanding.

1. INTRODUCTION

In the area of Natural Language Processing, Question Answering (QA) system is a task, which is designed to automatically answer a user's question placed in a natural language instead of a template base selection of user's requirements. The answer to a question may be produced from a background data collection. A QA system requires going through a different steps of scientific techniques in order to assemble a probable answer before returning the same to the user. An open-domain QA system generally requires returning an answer to a question in a precise or exact form or in a short form of natural language text snippet as expected by the users of the system.

The massive increase in the amount of on-line text these days and the drive for accessing various types of data have arisen the interest in a broad spectrum in the area of QA systems that go well beyond simple document retrieval [1]. It has now become a necessity to combine search technologies and knowledge about a user's query and context into a single framework for providing the most suitable answer to a user's requirements. In the case of QA systems, the burden of finding related material is placed on the system that is responsible for scanning and navigating the retrieved material to find an answer to a question. This requires a well representation of the background data and also interpreting a query in detail for locating the search context, as well as many other useful information from the question and the background data collection.

Most of the QA systems use the architecture of Information Retrieval (IR) [2] techniques at their core. These techniques typically depend heavily on utilizing the question words and syntactic structure of natural language texts and do not try to understand the semantics articulated by the text [3, 4]. Retrieval-based Question Answering systems [5] actually consider the question as a query and uses a standard word-based ranking model to retrieve the most relevant fragments from the test document residing at the back end. The current architecture of Question Answering systems re-

volves around more or less a standard [6] set of components that include a processing module for the input questions, a data retrieval module which may extract information likely to contain an answer and finally a module that extracts and ranks [7] the tentative answer part of the information isolated and presents the same back to the user. A smooth interaction between these modules is a prerequisite for arriving at a precise answer to a given question. In this type of architecture, the background data collection is stored in an index which hardly understand any coherent structure and meaning of the texts. The answer is extracted using a set of templates [8] or a set of question-answer patterns [9].

A few systems such as QANDA [10] combines both IR approach and natural language processing in a single framework to address the QA systems. Systems like QANUS [11] uses a pipelined architecture [12] with document retrieval and analysis at the core. Web-based QA systems like ARANEA [13] utilizes the vast and redundant resources available on the web to achieve a convincing performance over Text REtrieval Conference's evaluation data. Some recent works are based on pre-defined domain specific ontologies [14]. These systems are based on semi-structured knowledge-bases and do not use free text documents on the web directly. They require the web documents to be in a structured or semi-structured format to allow their systems to access them conveniently. Shallow QA systems locate the answers directly from documents, whereas the deep QA systems may require to have the capability to do inference on the facts.

Knowledge-based QA systems [15, 16, 17, 18, 19] use the general strategy of analysing the question and mapping the question words to the words appearing in the knowledge-base (KB) and formulate search queries to extract answers.

Most of the state of the art system architectures heavily depend on the similarity measure of the question sentence and the background sentences. For some types of questions, these systems may show acceptable performance to some extent, though they all miss out the points of understanding the question and background collections, in general. Extracting useful information by analysing unstructured data from any open-domain QA system is presently the most difficult challenge. Our Proposed system contributes in this area and presents a model which is designed upon open-domain unstructured data. The key idea is to create a useful structured knowledge-base (KB) from unstructured data which will extract and infer many useful information and store them ontologically in some spread over multiple data indices or KB. These KBs will play significant roles in answering factoid questions of various types ontologically and semantically. Our system can currently handle factoid questions which look for an entity, such as a person, organization, date/time, different types of numerical quantity, etc. The system also handles complex person description or definition types of questions that demand a short paragraph of few relevant sentences.

The paper is organized in the following sections. Section 2 discusses the proposed system architecture. Section 3 explains the experimental data set, results and comparison with other systems. Section 4 highlights a discussion and concludes the paper.

2. ARCHITECTURE OF SEMONTOQA

The proposed architecture presents a framework for a semantic understanding oriented knowledge-based QA system which we call 'SEMONTOQA'. The framework deals with open-domain free texts and creates a semantically rich and machine understandable ontological database from there in an effective manner. The working procedure of this module consists of three major modules, namely the Text Analyser for background data processing, Question Analyser and an Ontology Mapping module as shown in figure 1.

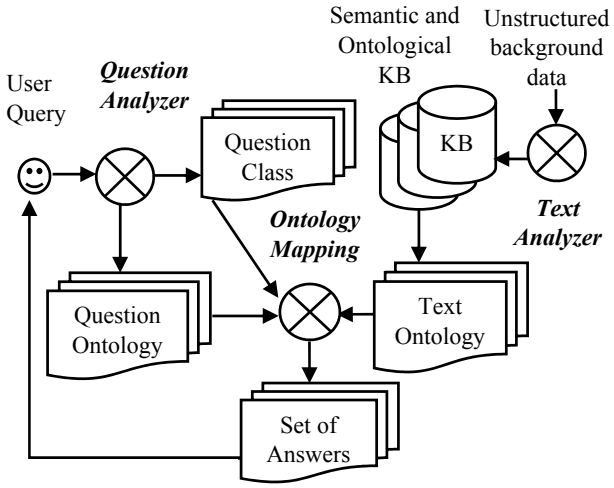

Figure 1: Architectural diagram of SEMONTOQA

2.1 Text Analyser

This module is responsible to build the core knowledge-base that will be used to answer the user queries. SEMONTOQA framework can answer a question from the available background data only. If there is no answer in the background-text, SEMONTOQA will not answer the question. This module receives unstructured free text as its input. The free text does not necessarily means noisy data. The text should be in English readable form. The Text Analyser module consists of two major sub-modules, namely an Automatic Text Interpreter (ATI) and a Text Inference Engine (TIE).

In the beginning, the Text Analyser divides the background-text data into a set of paragraphs [20, 21] or into text segments. A paragraph here is considered to be the candidate text having sufficient information retaining the possible context of the paragraph within the text itself. It is important to define the paragraph boundary in case no such boundary can be found in the background collection. For oversized paragraphs, we allow a maximum of 10 sentences in a paragraph.

In the next step, pronouns in the text paragraphs are resolved. We use state of the ART tools [22, 23, 24] for detecting pronominal references in the paragraph and resolve them with their corresponding referenced nouns. In the resolution step, personal and possessive pronouns are resolved and reflexive pronouns are removed. Complex and longer English language sentence that contains individual sentences joined with connectives are divided into simpler sentences using the Stanford TREGEX [25] tree matching patterns found in the Stanford parse tree [26] of the complex sentence. The parse

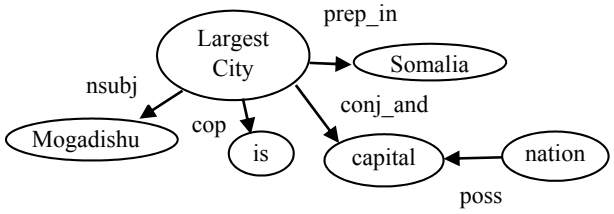

Figure 2: Syntactic Sentence Graph (SSG) model for the S2

tree is traversed and the complex sentence is segmented into individual sentences bearing independent meanings. This step is performed after the anaphora and co-references are resolved so that, bringing a complex sentence down to individual simpler sentences do not cause the sentences to lose any significant information from the complex form. Even the top of the line state of the art tools suffer to recognize the proper syntactic relations between word tokens in a long and complex sentence. For example, the following sentence S1 can be divided into two sentences starting from the underlined position as shown below.

S1: John Cena has started John Cena's professional wrestling career in 1999 with Ultimate Pro Wrestling, where John Cena held the UPW Heavyweight Championship.

The Text Analyser then represents each of the sentences of a paragraph using a minimalistic labelled graph that we call a Syntactic Sentence Graph (SSG). The graph depicts the relations among the expressions used in the sentence. We have utilized the typed dependency output given by the Stanford NLP parser [27] and MaltParser [28] for constructing the SSG. Natural language tools are prone to errors. Stanford parser was used at first and the *n-number* of highest-scoring parses for a question sentence were initially retrieved. In case, the parsing probability (normalized parsing score into percentage) of the highest parsing score was below a predefined threshold, MaltParser was used for getting the typed dependencies.

To improve the parsing speed, LingPipe's part of speech tagger [29] was used to label the part of speech (POS) tags of the each of the word tokens of the question sentence during the preprocessing step. A sample SSG for the sentence *S2: Mogadishu known locally as Xamar, is the largest city in Somalia and the nation's capital* is shown in figure 2. The SSG is constructed after dropping the articles, determiners and combining Named Entities, phrases, modifiers together, etc., from the typed dependency output.

Named Entities such as name, organization, location, date, time, duration, monetary items, etc., are detected using standard tools for NE detection[20]. We propose to use DBpedia [30] to assist the system recognizing the 'Person' type named entities. For example, in the sentence, 'Jay-Z co-owns the 40/40 Club', none of the NE detection tools managed to identify *Jay-Z* as a named entity. A simple SPARQL [31] query passed to the DBpedia and getting a non-nil set of documents as a return can confirm the detection of person names.

2.1.1 Automatic Text Interpreter (ATI)

The Syntactic Sentence Graph (SSG) is not semantic. Automatic Text Interpreter (ATI) uses a pre-learnt set of Text

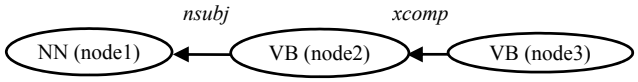

Figure 3: An example TIS learnt from the annotated texts

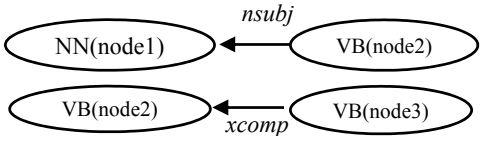

Figure 4: Simplified (Broken down) form of the TIS as shown in the figure 3.

Interpretation Subgraphs (TIS) to give a semantic meaning to the SSG. For getting the TIS set, we used a semi-supervised learning approach to train the system using a set of sentences each of which was annotated with possible entities, their corresponding types, their features mentioned anywhere in the sentence and all possible actions taken by them appearing anywhere in the sentence in the following way:

> [Sentence$_1$]:[Subject$_1$ <NE type/Non NE>][Feature$_1$, Feature$_2$, .. Feature$_n$][Action$_1$, Action$_2$ .. Action$_n$].

During the training period, the ATI module traverses the SSG using the annotated sentence instances and learns a set of subgraphs that tells the system on how to reach the features and actions from the annotated subject entity. For example, a simple Text Interpretation Subgraph (TIS) as shown in the figure 3 represents that the subject entity mentioned in *node 1* (part of speech tag: noun) is the action taker or the subject of both nodes two and three.

When the ATI uses the learned subgraphs to locate features or actions taken by an entity, it has to perform a subgraph matching between each of the TIS$_i$ in the TIS set and the SSG. The longer subgraph comparison is computationally expensive. So, we actually break down the longer subgraphs into a shorter form of subgraphs with only two nodes and a relation.

For example, the TIS$_i$ shown in figure 3 can be broken down to two smaller TISs (figure 4) which keeps the meaning in the same way, but expresses them in a different way. In this case, the TIS in figure 4 articulates that, *node 2* is the action of *node 1* and *node 2* and *node 3* connected by an open clausal verb complement are equivalent actions taken by a common subject. Breaking down the longer TISs into shorter ones actually boils down to only a pairwise subgraph matching between the TIS entries and the SSG, which is computation-wise significantly less expensive.

Based on the subgraphs found in the TIS set, we manually design a total of nine matrices, six of which are designed for inferring features, actions, contexts of subjects. These six matrices are called *inference matrices*. The rest of the matrices contain various properties of features and actions. Each row in a matrix contains a pair of data. The matrix Subject-Feature (SF) contains the subjects and their directly connected features. Feature-Context (FC) contains the context of the features connected with prepositions and other connectives. Equivalent-Feature (EF) contains pair-

wise features that are supposed to have a common subject. Subject-Action (SA) contains the direct actions taken by subjects. Equivalent-Action (EA) contains pairwise action verbs that are tentatively equal from the point of view of a common subject. Action-Context (AC) contains the contexts of the actions connected with prepositions and other connectives. The non-inference matrix Action-Object (AO) contains a pair of an action and the direct object upon which the action takes place. Action-Temporal (AT) and Feature-Temporal (FT) matrices contain the temporal state of the features and actions respectively. The temporal state refers to *present*, *past* or *future* event or features. The connected auxiliary verb with a feature determines the temporal state of the feature. The state of an action verb, as well as the auxiliaries used with it, determine the temporal state of the action.

These matrices are populated by the ATI according to the TISs in the TIS set for each of the entities in the background data sentences inside each of the paragraphs. *Algorithm 1* shows the working procedure for the Automatic Text Interpretation module.

Algorithm 1 Automatic Text Interpretation

```
1: procedure ATI
2:     for each TIS_i instructions in the TIS Set do
3: Perform a pairwise subgraph matching between TIS_i and the SSG
4:         if there is a match with the nodes and relation then
5: Extract the nodes from the SSG at the matching point
6: Put the entity/feature/action/context/temporal value nodes in the inference and other matrices according to the TIS_i instruction.
7:         else
8:             Continue to match with the next TIS_i
9:         end if
10:    end for
11: end procedure
```

2.1.2 Text Inference Engine (TIE)

The Text Inference Engine (TIE) module takes the *inference matrices* as input and infers features and actions of the entities that may not be directly connected with them. The TIE performs cross matching *Cartesian product* between the inference matrices to pull out indirect features and actions by inference. The set of Text Interpretation Subgraphs (TIS) puts a semantic interpretation among a subject, their features, actions, objects of the actions and contexts in the SSG. The TIE lets the system to infer many features and relations that could not be detected otherwise, even with any current state-of-the-art discourse representation tools [32].

If we take a look at the sentence *S3: John Cena started his pro wrestling career in 1999 with Ultimate Pro Wrestling, where he held the UPW Heavyweight Championship and then signed a developmental contract with the World Wrestling Federation in 2001.* None of the existing state-of-the-art tools can infer that *John Cena signed a developmental contract ...*

For the sentence S3, the Automatic Text Interpreter (ATI) loads the following information in the Subject-Action matrix SA: *(John Cena, started; John Cena, held)* and the Equivalent-Actions matrix EA is loaded with the equivalent actions :*(started, held ; held, signed)*. A cross matching Cartesian product here considers each of the rows in the participating matrices as single entries that actually converts the participating matrices into two sets of elements. So, an *SA x EA* will produce the following information by taking $(x, y)|x \in SA \wedge y \in EA$:

(John Cena, started started, held), (John Cena, started held, signed), (John Cena, held started, held) , (John Cena, held held, signed)

And this will allow the system to infer a new fact, such as *'John Cena, signed'* and update the SA matrix with this new entry.

If we take a look at the sentence *'S4: Mogadishu known locally as Xamar, is the largest city in Somalia and the nation's capital'*, a Cartesian product between *SF (Mogadishu, city)* and *EF (city, capital)* allows the system to infer and add the following fact into *SF: (Mogadishu, capital)*. Moreover, an *EF (city, capital) x FC (city, in Somalia; capital, of nation)* allows the system to infer the additional facts, such as:

[*Entity:* Mogadishu <*type: city*>][*feature*: capital][*context*: in Somalia <*type: NE, Country*>]

[*Entity*: Mogadishu <*type: city*>][*feature*: city][*context*: of nation]

A question like: *'What is the capital of Somalia?'* can be answered from the above facts by an ontology mapping which we discuss in the upcoming sections. The TIE finally reports unique actions and features of each of the subjects from the SA and SF respectively. The contexts of the features and actions are reported from the FC and AC matrices respectively. Temporal status of the features and actions are stated from the Temporal matrices. The objects of the actions if there is any will be taken from the Action-Object matrix AO. All these information will be stored ontologically in a knowledge-base for the future use.

Algorithm 2 shows the working procedure for the action and feature inference part of the Text Inference Engine.

Algorithm 2 Text Inference Engine for actions and features inference

```
1: procedure TIE FOR ACTIONS AND FEATURES
2:     repeat
3: Perform Cross-matching Cartesian product
4:   Between SA x SA and update EF with new equivalent features
5:   Between SA x EA and update SA with new actions of a subject
6:   Between SF x EF and update SF with new features of a subject
7:   Between SF x SA and update SA with new actions of a subject
8:   Between EF x FC and update FC
9:   Between EA x AC and update AC
10:    until no new features or actions can be inferred for each of the subjects
11:  Report unique features of each of the subjects from SF
12:  Report unique actions of each of the subjects from SA
13: end procedure
```

Cognitive Entity-Relationship Network (CEN)

To understand natural language text in a more in-depth manner, we build a Cognitive Entity-Relationship Network

(CEN). To give the system a way to understand the text deeply, the CEN built manually can come very handy. It actually adds a significant extension to the text understanding process. Each entry in the CEN set is divided into two parts and these parts are connected with a logical operator such as implication, equivalence, etc. The first part of a CEN entry connects two entities with a relation between them. The relation can be a feature or an action. The second part of the entry tells the extended meaning of the first part in the same way as the first part. The feature/action appearing in the relation part is further extended with their syntactically relevant synonyms.

The TIE takes the knowledge-base (KB) entries and compare them for a match with a part of the CEN_i in the CEN set and based on the logical operator applied between the parts of the CEN_i entry, the system adds the other part of the CEN_i into the KB for extended inference capability of the system to answer questions in the future.

A few example CEN entries look like:

> [*Entity* : * <*type: NE, Person*>][*feature*: Professor] [*context*: University] => [*Entity* : * <*type: NE, Person*>][*action*: teach, instruct, profess][*context*: University]
>
> [*Entity* : * <*type: NE, Person*>][*action*: work][*context*: * <*type: organization*>] <=> [*Entity* : * <*type: NE, Person*>][*feature*: employee, worker] [*context*: * <*type: organization*>]

The first example CEN entry tells that *if a person is a professor at a University*, the system can add that the person takes actions such as *teach, instruct, profess* at a University. Similarly, the second entry tells that, *if a person works in some organization*, the system can add corresponding features of that person such as *worker, employee*. The equivalence logical operator mentioned here allows a similar interpretation for the other side of this entry too.

The CEN is not represented in a graph-based design, as it will require an extensive graph matching with the Syntactic Sentence Graph of the text snippet. Representing the CEN in terms of flat knowledge-base like entries, as shown above, allows the system to search for a similarity using a simple attribute wise matching with the entries of CEN and the text ontology.

Moreover, the CEN helps the system to understand complex facts. For example, A CEN entry like the following:

> ([*Entity* : X <*type: male, NE, Person*>] [*feature*: married] [*context*: Y <*type: female, NE, Person]*)
> =>
> ([*Entity* : X <*type: NE, Person*>] [*feature*: husband] [*context*: Y <*type: NE, Person*>] ∧ [*Entity* : Y <*type: NE, Person*>] [*feature*: wife] [*context*: X <*type: NE, Person*>] ∧ [*Entity* : X <*type: NE, Person*>] [*feature*: spouse] [*context*: Y <*type: NE, Person*>] ∧ [*Entity* : Y <*type: NE, Person*>] [*feature*: spouse] [*context*: X <*type: NE, Person*>])

helps the system to infer facts that *if X is married to Y* , then X and Y has features like *husband* and *wife* and they are *spouse* of each other. None of the existing QA systems interpret these type of facts to answer questions.

CEN also adds entries in the CEN set for some features which have a small set of values and the background-text mentions the values of those features without explicitly mentioning the features. For example, in the text snippet: *'Mahatma Gandhi was a Sanatani Hindu'*, our system extracts from the text that, the entity *'Mahatma Gandhi'* has a feature: *Sanatani Hindu*, but cannot understand that, it is talking about the entity's religion. CEN extends this type of deep understanding capability to the system. Our system can capture this significant detail by inferring the text with the help of a simple CEN entry such as:

[*Entity:* * <*type :NE, Person*>][*feature* : Hindu, Christian, Muslim , Jew...][*context* : *] => [*Entity*: * <*type :NE, Person*>][*feature* : religion] [*context* value: Hindu, Christian, Muslim , Jew ..]

Pattern-based Information Extraction from the Text to the Knowledge-base

Apart from detecting entities, features and actions, our system classifies the features of the entities to a few different finer types [33]. These finer types include various numerical types for handling questions that ask for a distance between entities, length, weight, speed, monetary value, size of entities, etc. To understand these types from the text and storing them in the knowledge-base (KB), we use a set of pattern-based rules to associate a numeric number or text to a numeric type and to an entity.

For example, from the text snippet: *'...the Great Wall of China is 5,500 miles long..'*, our system associates a feature *'5,500 miles long'* of the entity *'Great Wall of China'*. Using a predefined pattern in a patter set, such as: [*Length: (0-9)* mile/km/meter long/tall..*], the Text Inference Engine sets the *'length'* property flag in the KB to true with this detected feature. So, the KB contains an entry like the following for the above-mentioned text snippet:

[*Entity*: Great Wall of China <*type: unknown*>][*feature*: length] [*context*: 5500 miles][*location*: China <*type:NE, Country*>] .. [*lengthflag* : true>] ..[..]

This allows our system to understand and answer *numeric type* of questions such as *'How long is the Great Wall of China?'*.

Because, the system looks for type-specific keywords and patterns in the text, it can handle different representations of the same text containing similar information. So, another representation of the above text snippet such as *'The length of the Great Wall of China is 5,500 miles ..'* can still be associated the same way based on the word *length* and a matched pattern with the *5,500 miles*. Similarly, for the text: *'The best price of Apple iPhone 6 in India is Rs. 42606.'*, Our system sets the *moneyflag* to *true* in the following way: [*Entity*: Apple Iphone 6 <*type: unknown*>][*feature*: price, cost][*context*: RS. 42606] [*location*: India <*type:Country*>] .. [*moneyflag*: true].

2.1.3 The Knowledge-base

Information extracted from the text is stored in multiple spread over data storages that we call a knowledge-base (KB). The KB contains, a paragraph-level, a sentence-level, an entity-feature and an entity-action level data storage having useful information extracted from the text. The entity-feature storage in the KB contains entities, types, their direct and inferred features, contexts, and other associated information of the features that we call properties such as location, date/time, duration, monetary value, length, height etc. The entity-action storage contains similar information,

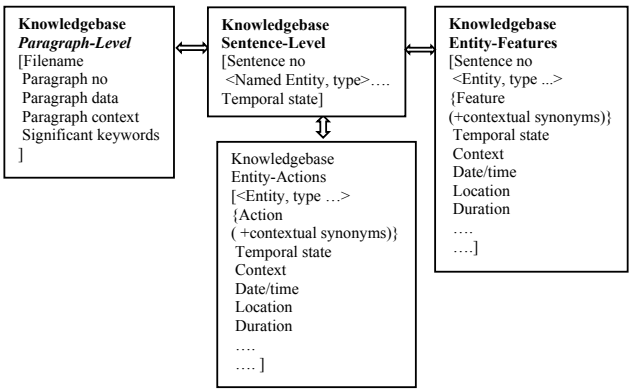

Figure 5: The Knowledge-base (KB)

such as entities, their direct an inferred actions, objects of the actions (if any), the contexts of the actions, and associated properties the same way as that of the entity-feature storage (figure 5).

Contextual Word Sense Disambiguation is performed on the text using WordNet [34, 35] and the semantically related synonyms of the features/actions are also preserved to extend the ability to understand their meanings in a more detailed way.

So, the entity-feature storage in the KB built from the following text snippet: *'George Walker Bush is an American politician and businessman who was the 43rd President of the United States from 2001 to 2009, and the 46th Governor of Texas from 1995 to 2000.'* will uncover one of the many other facts in the text ontology in the following way:

[*Entity*: George Walker Bush <*type: NE, Person*>][*feature*: president, head of state, chief of state, chief executive..[*context*: of United States <*type: NE, Country*>][*temporal*: past] [*date/time* : 2001 to 2009][*location*: nill] ..[..]

Similarly, the text snippet: *'On June 25, 2009, Jackson died of acute propofol and benzodiazepine intoxication..'* reveals the following fact in the entity-action storage of the KB such as:

[*Entity*: Michael Jackson <*type: male, NE, Person*>] [*action*: died][*context*: of acute propofol, of benzodiazepine intoxication] [*date/time* : June 25, 2009]..[..]

Extracting Significant Keywords and Storing the Paragraph Contexts in the Knowledge-base

To create the paragraph-level KB, the Text Analyser module extracts useful phrases and keywords from the text and calculates the relevance score of each of the keywords/phrases. Top two keywords/phrases are assigned as the theme of the paragraph that the paragraph is talking about.

A question like: *Q1:'Who was the sole survivor of Sago Mine disaster?'* could not be directly answered from the text snippet: *'The Sago Mine Disaster was a coal mine explosion on January 2, 2006, in the International Coal Group's Sago Mine in Sago, West Virginia. The blast and ensuing aftermath ... Only one miner,* ***Randal McCloy****, was the survivor.'*, though the snippet contains the answer for sure. Our TIE can detect from the last line of the paragraph that, *'survivor'* is the feature of the entity *Randal L. McCloy*. But this line does not tell that, the entity is the survivor of the event *Sago Mine disaster*. Here the con-

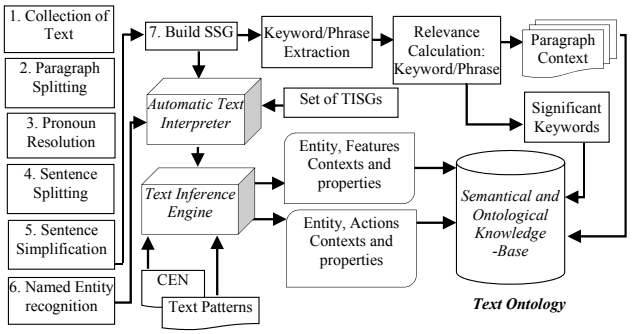

Figure 6: Working Procedure of the Text Analyser module

text/theme of the paragraph is *Sago Mine Disaster*. If the system would have the ability to read in context of the paragraph, it could understand that *Randal McCloy* was the sole survivor of the *Sago Mine Disaster*.

To detect the top ranking keyword/phrases from a paragraph, each of the extracted keywords/phrases are ranked based on their relevance scores in the paragraph. For calculating the relevance scores of the keywords/phrases, the system isolates the smallest noun phrases first. The relevance score is normalized to a [0-1] range. All kinds of date/time/numbers are avoided from the keyword/phrase list as they bear very less significance in the context/theme calculation of a paragraph. Points are calculated based on the criteria shown below and a positive(+1) or negative(-1) points are added to a keyword/phrase based on some heuristics.

Criteria for point assignment of the keyword k_i or phrase p_i:

C1: Term frequency (including individual tokens of the phrase p_i / keyword k_i), C2 : ((Total number of candidate keywords or phrases in the text)/(position of appearance of the keyword k_i or phrase p_i in the paragraph))

Heuristics for point addition /subtraction of the keyword k_i or phrase p_i: H1: Part of a Named Entity [organization, location, person etc.](+1); H2: Participates in any action(s) (+1) / Has feature(s)(+1); H3: Is the entity a subject(+1), H4: First letter capitalization for the word tokens of keyword k_i or phrase p_i (+1 for each capitalization). H5: Object of a subject(-1), H6: Object of a preposition(-1).

Each keyword k_i or phrase p_i , thus gets a relevance score and the top scoring two candidate keywords/phrases get selected as the contexts/theme of the paragraph. In the above text snippet, the phrase *Sago Mine Disaster* achieves the highest relevance score and is preserved as the contextual theme of the paragraph. The system, in the context of the paragraph can then answer the above-mentioned question Q1 easily. The overall architecture for the Text Analyser module is shown in figure 6.

2.2 Question Analyser

The SEMONTOQA module deals with factoid types of questions along with person description, definition types of questions. It does not handle questions that ask for a procedure or reason.

2.2.1 Question Preprocessing and Spell Checking

The Question Analyser module performs question prepro-

cessing, which removes unwanted noises or symbols from the question and performs a non-interactive and automatic spell checking that replaces a misspelled word to the suggested words without the user's intervention.

2.2.2 Question Classification

We used the question classification taxonomy used in [33] and questions were classified in 50 finer sub-classes of 5 major classes such as Entity, Location, Numeric, Description, Human. An extended set of Text Interpretation Subgraph that covers the question sentence structure detects a question's focus. The proper meaning and root the of question focus word was detected in the context of the question words using WordNet. The meaning of the question focus was then compared with the detailed definition of the finer sub-classes for a similarity score. The finer class definition getting the highest similarity score was assigned to the question. Each finer class definition is connected to a major class and thus, the major class could be detected from the connection with the detected finer sub-class.

We also combine the approach of a finite state machine based classification to learn a set of patterns for various question classes [36]. Question class gives the system an idea on the type of answer to look for in the knowledge-base. For example, the question, *'Who is the CEO of 3M?'* is asking for an entity as answer whose type is Named Entity:Person. The Question: *'How long is the Brooklyn bridge?'* is asking for the *length* of the entity '*Brooklyn bridge*.

2.2.3 Building the Question Ontology

SEMONTOQA does not depend on a simple Information Retrieval based approach that uses keywords from the question to find a set of documents first and finds an answer using a set of question-answer patterns only. No types of understanding is involved in such systems. Because the answering process of SEMONTOQA is understanding based, the Question analyser has to understand the question both syntactically and semantically and generate a question ontology with entities, attributes, features, actions and other properties so that the system can look into the knowledge-base for a valid response.

The Question Analyser converts a question into a Syntactic Sentence Graph (SSG) in the same way the Text analyser module does and uses the set of Text Interpretation Subgraphs (TIS) to extract the text ontology from it. The question sentences differ from the background data collection in terms of the sentence structure. Questions are supposed to start with a question word such as *what, who, how, when, where, whom*, etc. A Question starting with a question word actually asks for some information that are mentioned in the question in a complex way. If we take a look at a question *Q2: 'When was John F. Kennedy killed?'* , our system using the TIS set, can detect from the question's SSG that the system is looking for the *date/time* of an action *'kill'* which is a *past event* and the object of the action is *John F. Kennedy*. No subject of the action is mentioned in the text. So, a question ontology can be formulated as follows for the above question:

[*Entity*: *][*action*: kill, assassination, murder][*object*: John F. Kennedy <*type: NE, Person*>][*location*: *]... [*date/time*: ?][..] ... [..]

Here * means, the value for the attribute can be anything, and a question mark in the *date/time* field indicates a value

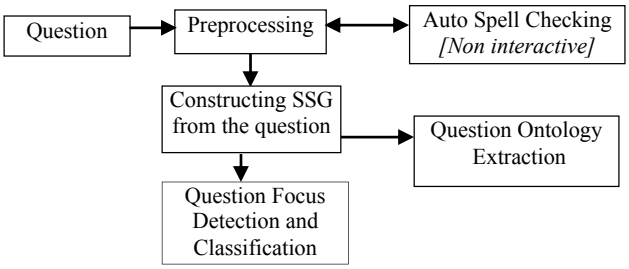

Figure 7: Working procedure for the Question Analyser

that the question is looking for.

Depending on the question class, the ontology builder uses different strategies to extract multiple question ontologies. For example, the question *'How tall is the Eiffel Tower?'* is asking for the height of the entity *'Eiffel Tower'* based on the question focus word *'tall'*. Depending on the *Numeric:height* class type, the system generates following alternative question ontologies for the answer mapping module:

[*Entity*: Eiffel Tower][*feature* : ?][*context*: *] [*temporal*: present][*location*: *] .. [*date/time*:*] .. [..] [*lengthflag*: true]

[*Entity*: tall, height][*context*: Eiffel Tower][*feature* : ?][*temporal*: present][*location*: *] .. [*date/time*: *] [..] ...[..] [*lengthflag*: true]

When generating a question ontology, a question is contextually analysed for an all-word Word Sense Disambiguation and the syntactically related synonyms of the action/features or subjects, etc. are also added into the question ontology to expand the query coverage.

The overall working procedure for the Question Analyser is shown in figure 7.

2.3 Ontology Mapping Module

Once we have the question ontology in hand, the Answer Mapping Module (OMM) searches the background knowledge-base (KB) to extract the desired answer based on it. The Ontology Mapping module uses some heuristic guidelines based on the question classes for picking up one of the few different strategies to look for an answer.

The background knowledge-base is represented in a MySQL database as well as in a database like format using Lucene's data structure [37]. The OMM converts Question Ontology to corresponding SQL and Lucene queries. For partial matching, we have used Lucene's proximity search technique that matches the availability of tokens within a specific distance rather than matching the whole string which is likely to cause the system to fail while finding a match with the names of Named Entities. In the case of partial matching search, the search is made in the Lucene-based KB first to reduce the search space. OMM, then builds a *'SELECT'* database SQL query [38] and passes to the KB stored in the MySQL environment to get the results (figure 8).

2.3.1 Direct Mapping of the Text Ontology with the Question Ontology

If we take a look at the following question: *Who is the author of 'Harry Potter and the Goblet of Fire'?* and the question ontology in the following form of attributes and values.

[*Entity*: ? <*type: NE, Person*>][*feature*: author, writer

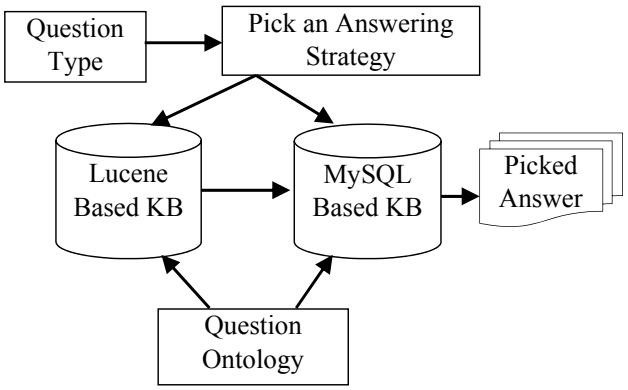

Figure 8: Working procedure of OMM

..] [*context*: Harry Potter and the Goblet of Fire][*date/time*: nill][*location*: nill][..]..[..]

In case, the knowledge-base has the following text ontology, the OMM using the above question ontology can give a straightforward answer by a mapping the question ontology to the text ontology.

[*Entity*: J. K. Rowling <*type: female, NE, Person*>][*feature* : author, writer] [*context*: Harry Potter and the Goblet of Fire][*date/time*: nill][*location*: nill][..]..[..]

2.3.2 Question Ontology Conversion

If the system is asked a question like *'Who is the killer of John F. Kennedy ?'*. It is very likely that, it won't have a direct mapping of the question ontology to the text ontology. If the system has the following text ontology: [*Entity*: Lee Harvey Oswald <*type: male, NE, Person*>] [*action*: kill,assassinate, murder..] [*temporal*: past event] [*context*: John F Kennedy <*type: male, NE, Person*>] [*date/time*: nill] [*location* : nill] [..].. [..], the OMM requires to perform a feature (noun) to action (verb) transformation using WordNet and get the verb representation of the feature. Here, the noun feature *killer* in the question ontology can be transformed to its equivalent verb *kill* and the question ontology can be transformed into the following form:

[*Entity: ?* <*type: NE, Person*>] [*action*: kill, murder..] [*temporal*: *] [*object*: John F Kennedy <*type: male, NE, Person*>][*date/time*: nill] [*location* : nill][..]..[..]

The previous *temporal state* of the feature *'killer'* is changed from *present* to *any* as the temporal state of the feature cannot be confidently applied to the transformed action. The transformed question ontology can now have a match with the text ontology and an answer can be returned to the user of the system. The Cognitive Entity Relationship Network is also consulted to check if it can help the feature to action conversion or vice versa with the pre-stored facts stored in it.

For answering *Human:description* questions like*'Who is Jay-Z?'*, the OMM matches the Named Entity *'Jay-Z'* in the subject field of the KB and returns the paragraph that starts with a feature description of the entity or talks more about the entity's features. In this case, the system uses partial matching for mapping the name. In case, the NE matches partially with several different names, the system selects the name with the highest term frequency in the KB. Since our system is not a dialogue based interactive system, it uses the above heuristics to show the description of the name that is most popular in the KB.

Moreover, a set of question class based Answer Interpretation Rules was developed that tells the OMM that the entity-action ontology: *'Jay-Z was born in New York.'*, is also a valid selection for responding to the above type (Human: description) of questions, in case the system fails to locate any paragraph that begins with or contains features of the subject. Similar type of strategy is applied for answering *definition* types of questions.

3. EMPIRICAL VERIFICATION

We set the experimental environment for testing the effectiveness of the Automatic Text Interpreter and Text Inference Engine first and then test the effectiveness of the whole system SEMONTOQA in terms of questions from the TREC 2007 [39].

3.1 Performance Measure of the Text Analyzer Module

For testing the performance of the Text Analyser module in terms of ATI and TIE, we take a total of 100 factoid questions from the TREC 2007 data set based on AQUAINT-2 corpus. For the selected questions, from the answer containing paragraphs of them, we took 100 diverse sentences containing features and actions related to the questions. Another 100 sentences were picked randomly that contained a significant number of features and actions connected with various subjects directly and indirectly. This was done to ensure that, the system can get used to the various text representation format. We also use a Jsoup [40] based custom implementation for searching on the website of Google [1] to get top 5 documents for the diverse types of selected questions and a total of 100 Sentences were manually picked from there that had plenty of different types of features, actions related to various entities.

A set of 100 questions was created manually to test the performance of the ATI and TIE. Performance was measured in terms of *Precision*, *Recall* and *F-measure*. Precision, Recall and F-measure were calculated based on the following formula for testing the feature extraction process. A Similar formula was applied for testing the action extraction.

$$\text{Precision} = \frac{\text{relevant features} \bigcap \text{retrieved features}}{|\text{retrieved features}|}$$

$$\text{Recall} = \frac{\text{relevant features} \bigcap \text{retrieved features}}{|\text{relevant features}|}$$

$$\text{F-Measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A total of 300 training set sentences were divided into a sets of 100, 200 and 300 sentences respectively. *Table 1* shows the results for *feature* extraction and *Table 2* depicts the similar results for the *action* extraction.

Natural language processing tools are not perfect. From the results in *Table 1* and *2*, we can see that *precision* and *recall* values improved with more training set data used to train the system. Larger training set allowed the system to learn varieties of Text Interpretation Subgraphs that the ATI and TIE used later on to understand texts. Because of wrong dependency parsing, the TIS set was populated with some confusing subgraphs. A manual intervention was

[1]http://www.google.com

Table 1: Performance measure of TIE for feature extraction using TISs

Training Set Size	Precision	Recall	F-Measure
100	0.84	0.83	0.83
200	0.91	0.87	0.89
300	0.96	0.91	0.93

Table 2: Performance measure of TIE for action extraction using TISs

Training Set Size	Precision	Recall	F-Measure
100	0.80	0.77	0.78
200	0.84	0.80	0.82
300	0.89	0.85	0.87

required for getting the TIS entries in the TIS set right. Some fine tuning is still required for getting a more accurate TIS set. The action extraction process using the TISs labelled some actions erroneously with wrong entities and the precision value went down for that reason. Because of the complex nature of free English language text, our system failed to locate some features and actions properly. The performance achieved is still quite satisfactory considering the complexity found in the sentence structures in the test set.

3.2 Performance of the Question Answering System

In order to evaluate the performance of the overall SEMONTOQA system, we use the questions and answers from the TREC 2007 [39]. We use AQUAINT-2 corpus which contains around 2.5 GB of background text in XML format. We use a custom built XML processing unit to extract the raw text data from the XML files. The corpus was built from news documents which contain plenty of noise and less significant texts. To avoid handling the huge, noisy data set, our system skipped portions of the data which did not qualify as useful texts. Because, our approach is not Information Retrieval (IR) based, it was not important for us to worry about the recall values of the document retrieval. Rather, we were concerned about the text and question understanding and the construction of an effective knowledge-base from the free text which was the most important step for the system. Currently, SEMONTOQA supports only factoid type of questions and for that reason, we use the general evaluation metric used for getting the accuracy of the factoid questions, which is defined using the following formula:

$$\text{Accuracy} = \frac{\text{Total no. of correctly answered questions}}{\text{Total no. of factoid questions in the test set}}$$

TREC 2007 data set had a total of 360 questions. We have used a total of 200 questions for evaluating the performance of the system. The questions were about some target of interests. The target of interests was about any person, organization, something, or an event. TREC 2007 also used the Blog06 corpus and we did not have that corpus with us. For that reason, we fetched Google's top 10 documents for each of the questions and added them into the system for analysing. The Text Analyser created the knowledge-base using the data from the AQUAINT-2 corpus and from the files fetched from Google's top 10 results.

The top system in the TREC 2007 QA track was **LYMBAPA07** with an accuracy of **0.706** [39]. The accuracy of SEMONTOQA calculated from the above formula was **0.740** which we find as quite satisfactory. Our system did not find the answer to a total of 15 questions in the KB and answered a total of 148 questions correctly. Out of a total of 185 questions answered, 37 questions did not have the right answers or the system gave wrong types as answers. Misclassification and lack of a proper interpretation of the questions were the contributors to this error. Moreover, the error in the TIE module propagated to the answer mapping module and was another contributor to the overall error rate of the system. Fine tuning the Text Interpretation subgraphs should be able to reduce the overall error rate of the SEMONTOQA by a good margin.

SEMONTOQA was heavily benefited from the use of Cognitive Entity-Relationship Network, which allowed it to answer questions like: *'Which university does Paul Krugman teach?'* from the text snippet: *'Paul Krugman, the New York Times columnist and professor of Princeton University...'* as it could infer the fact from the CEN that the entity *Paul Krugman* being a *professor* at Princeton University *teaches* at Princeton University.

It is very difficult to classify every feature/action of entities to proper semantic classes. For example, our system could answer the following question confidently, *What organism causes Lyme disease?* from the KB entry that was formed from the text snippet: *"...Borrelia burgdorferi, the organism that causes Lyme disease..."*. though, it could not detect *'Borrelia burgdorferi'* as an *organism* because, an extensive semantic type annotator is required for this purpose. Currently, there is no such tools that can perform semantic type annotation for everything in open-domain free texts. In a domain-specific system, it may be possible to develop such an annotator to cover all possible semantic types which will allow our system to perform even better in terms of understanding.

We could not get into a head to head comparison with the systems competed in TREC 2007 as the data set that we used and the total number of questions were not exactly the same that other systems used during the competition. Our aim was to see how our semantic understanding knowledge-based system performs with the questions and corpus used by the competing systems. A better accuracy achieved at the current version of the SEMONTOQA is very much encouraging and we will be glad to test the system with other corpus and questions in the future. Results from the current prototype clearly suggest that SEMONTOQA can possibly become an alternative way for the QA systems in future.

4. DISCUSSION AND CONCLUSION

The current work is presented as an outcome of an ongoing research in the field of Natural language based Question Answering system. For the factoid QA systems, the performance of the existing systems for English language lies on or below the 70% range and thus allows newer techniques to be incorporated for better results. Though, knowledge-based QA systems do exist, the understanding based QA systems rarely exist and was the motivation behind the current work. Our system achieved an accuracy rate of around 74% which is very encouraging. Most importantly, the sys-

tem tried an understanding based approach to answer the fact-based questions. In the future, we shall present the report of the performance of SEMONTOQA on other corpus and questions.

Addition of *Cognitive Entity-Relationship Network* to assist the Text Inference module performed well beyond expectation in the case of factual inference. For example, an entry in the CEN that tells that *'If a person X is the son of a person Y(male) => Y is father of person X'* extends in reality the inference capability to a great extent. If the system is domain specific, then the addition of CEN in the way it was incorporated in our system based on domain-specific knowledge will allow the system to understand facts in a proper way. The simpler attribute based implementation will also perform very fast in computationally powerful computing systems. Designing a CEN for an open domain Question Answering system was a difficult task. We think that the CEN can be enhanced more by continuously adding newer facts into the system over time.

Our TIE module suffered because of wrong dependency parsing by the dependency parsers used. It will be interesting to see the performance of the TIE module by incorporating Combinatory Categorial Grammar (CCG) [41] based parser. We could not test the system using a CCG parser in this version. In the future, we plan to change the parser and train the system to learn different subgraphs and observe the effect on the accuracy.

At the moment, the system is fully dependent on the offline knowledge-base (KB) and cannot answer any question if the KB does not have relevant facts pre-stored. Though, the current model can be modified easily to crawl web-based texts during the non user-interaction mode and update its KB frequently to accommodate newer questions over the time. We have a plan to implement this step in the near future to transform the system into a more usable QA system that understands to some extent both the background text and the users' questions before giving an answer.

5. ACKNOWLEDGEMENTS

The current work is funded by EMMA in the framework of the EU Erasmus Mundus Action 2.

6. REFERENCES

[1] E.M. Voorhees and D. Harman., Overview of the eighth text retrieval conference (trec-8). pages 1–24, 2000.

[2] G. Salton., Automatic Information Organization and Retrieval. *McGraw-Hill*, NewYork, 1968.

[3] A. Hickl, K. Roberts, B. Rink, J. Bensley, T. Jungen, Y. Shi, and J. Williams., Question Answering with LCCs CHAUCER-2 at TREC 2007. In *Proceedings of Text Retrieval Conference.*, 2007.

[4] S.R. Joty and Y. Chali., University of Lethbridge's Participation in TREC 2007 QA Track. In *Proceedings of Text Retrieval Conference.*, 2007.

[5] S. Verberne., Retrieval-based Question Answering for Machine Reading Evalua-tion, CLEF. In *CLEF 2011 Labs and Workshop, Notebook Papers.*, Amsterdam, September 2011.

[6] D. Jurafsky and J.H. Martin., Speech and Language Processing. *2nd Edition, Prentice Hall Series in Artificial Intelligence.*,2008.

[7] J. Ko, L. Si and E. Nyberg., Combining evidence with a probabilistic framework for answer ranking and answer merging in question answering. *Elsevier Journal: Information Processing and Management 46.*, 541–554, 2010.

[8] A. Andrenucci and E. Sneiders., Automated question answering: review of the main approaches. In *Proceedings of ICITA '05*, pp:541–554, 2010.

[9] A.C. Mendes, L. Coheur, J. Silva and H. Rodrigues., Just.Ask - A multi-pronged approach to question answering. In *International Journal on Artificial Intelligence Tools* , vol.22, n.1, 2013.

[10] J. D. Burger, L. Ferro, W. Greiff, J. Henderson, M. Light, and S. Mardis., MITRE's Qanda at TREC-11. In *Proceedings of the Eleventh Text Retrieval Conference.*, 2003.

[11] J. Ng and M. Kan., QANUS- An Open-source Question-Answering Platform. 2010.

[12] L. Hirschman and R. Gaizauskas., Natural Language Question Answering: The View From Here. *Natural Language Engineering* , Vol:7, Issue 4, pp:275–300, December 2001.

[13] J. Lin., An Exploration of the Principles Underlying Redundancy-Based Factoid Question Answering. *ACM Transactions on Information Systems.*, 27(2): 1–55, 2007.

[14] D.S. Wang., A Domain-Specific Question Answering System Based on Ontology and Question Templates. In *Proceedings of 11th ACIS International Conference on Software Engineering.*, Artificial Intelligences, Networking and Parallel/Distributed Computing, SNPD, London, 2010.

[15] V. Lopez, V. Uren, E. Motta and M. Pasin., AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web.*, 5(2), 72–105, 2007.

[16] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum., Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.*, pp: 379–390, July 2012.

[17] C. Unger, L. Buhmann, J. Lehmann, A.C. Ngonga Ngomo, D. Gerber, and P. Cimiano., Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, pp: 639–648, April 2012.

[18] D. Damljanovic, M. Agatonovic and H. Cunningham., FREyA: An interactive way of querying Linked Data using natural language. In *The Semantic Web: ESWC 2011 Workshops.*, Springer Berlin Heidelberg, pp:125–138, January 2011.

[19] C. Unger, P. Cimiano., Pythia: Compositional meaning construction for ontologybased question answering on the Semantic Web. In *Natural Language Processing and Information Systems.*, Springer Berlin Heidelberg, pp:153–160, 2011.

[20] C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard and D. McClosky., The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for*

Computational Linguistics: System Demonstrations., pp: 55–60, 2014.

[21] OpenNLP., *OpenNLP Tools, https://opennlp.apache.org/*, Accessed on October 1 2014.

[22] R. Mitkov., Anaphora Resolution: The State of the Art. *Paper based on the COLING'98/ACL'98 tutorial on anaphora resolution.*, University of Wolverhampton, 1999.

[23] M. Denber., Automatic Resolution of Anaphora in English. *Technical report, Eastman Kodak Co.*, 1998.

[24] E. Bengtson and D. Roth., Understanding the Value of Features for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp:294–303, 2008.

[25] R. Levy and G. Andrew., Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC 2006).*, 2006.

[26] D. Chen and C.D. Manning., A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of EMNLP 2014.*, pp:740–750, 2014.

[27] Marie-Catherine, B. MacCartney, and C.D. Manning., Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC* , 2006.

[28] J. Nivre, J. Hall and J. Nilsson., MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of LREC2006*, Genoa, Italy, pp:2216–2219, 2006.

[29] LingPipe., LingPipe tool kit for processing text using computational linguistics. *http://alias-i.com/lingpipe/*, Accessed on 01 March 2015.

[30] DBpedia., DBpedia Knowledge Base. *http://dbpedia.org/*, Accessed on 01 March 2015.

[31] SPARQL, DBpedia., SPARQL RDF query language. *http://dbpedia.org/sparql*, Accessed on 01 March 2015.

[32] Boxer., Boxer CandC tools. *http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo.*, Accessed on 11 March 2015.

[33] X. Li and D. Roth., Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics.*, pp: 1–7, Taipei, 2002.

[34] C. Fellbaum., WordNet: An Electronic Lexical Database. *Cambridge, MA: MIT Press.*, 1998.

[35] G. Miller., WordNet: A Lexical Database for English. *Communications of the ACM.*, 38(11):39–41, 1995.

[36] M. Hoque, T. Goncalves and P. Quaresma., Classifying Questions in Question Answering Systems using Finite State Machines with a simple learning approach. In *Proceedings of PACLIC '27*, pp:409–414, Taiwan, 2013.

[37] LUCENE., LUCENE : Apache Lucene Core. *https://lucene.apache.org/core/*, Accessed on 01 March 2015.

[38] A. Silberschatz, H.F. Korth and S. Sudarshan., Database System Concepts. *McGraw-Hill , Chapter: 3: Introduction to SQL.*, 6th edition.

[39] H.T. Dang, D. Kelly, and J. Lin. Overview of the TREC 2007 Question Answering Track. *TREC 2007.*

[40] JSOUP., *http://jsoup.org/download*, Accessed on 01 March 2015.

[41] S. Clark and J.R. Curran. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4), 2007.