Proposal | Applied NLP Project, Team: [Akshay Jagtiani, Pallavi Singh, Rahul Agarwal]

1. Task Addressed in the Project.

Our aim is to create a Question-Answer agent that can answer question (up to a certain nesting level) about not all, but constrained domains like people, places, books, films, etc. To achieve this task, our project would require two modules.

1.1 A knowledge base/ontology that has answers to all the questions. (E.g. Linked Data: DBPedia)

1.2 A Natural Language-to-KB Query conversion utility.

2 Points of Interest and Challenges.

2.1 Wolfram Alpha, Google, Evi, etc. have been trying to answer natural queries rather than giving links to relevant pages. So when a user asks: "**What is the capital of France?"** instead of links to the files, he gets the answer – "Paris". QA agent is a core NLP open-ended problem to be solved; it is a part of many devices in the form of Siri, Cortana and certainly an interesting domain to work on.

2.2 Natural language queries can be vague and complex, atomic/factoid sentences can be broken into RDF format and converted into a DBPedia query, but the challenge is to break more nested and complex natural texts into atomic ones and understand the dependencies between them.

3 Who would benefit, how?

An Internet user usually needs answers to various kinds of factoid questions like, 'Who is the CEO of Microsoft?' or 'When was Pink Floyd founded?' Modern search is all about - user being provided with the exact answers rather than the links. Today, it is all about how quickly people get their answers and how satisfied are they with the answers. With linked data, not only can we provide the pinpointed answers to their question, but also other related facts, which might be useful.

4 Data and Knowledge sources we will be using.

4.1 We have decided to use linked data ontologies (like DBPedia, LinkedOpenData), which are provided in the RDF format. This will be used as the Knowledge Base from which all the inferences will be drawn.

4.2 Data for classification and processing of Natural Language questions -

4.2.1 To classify questions based on their answer types. -- **TREC** data is openly available and is considered gold standard by many NLP applications. TREC has a question answer track, which contains classified question based on their answer types.

4.2.2 We would need to POS tag, NER tag and form chunks of data to identify the resource, subject and object in our question, which will then be converted into SPARQL query which DBPedia understands. For this we will need annotated English text. NLTK or CoreNLP can be used to achieve this task, using annotated brown corpus.

5 Collection of Data

Since TREC data is readily available for training a model in our application, there should not be a need to extract more data from any other sources. We will however need to collect data for the evaluation of our approach using various metrices. As our QA agent is domain constrained, we have thought of manually curating a set of 200-500 questions, which are uniformly distributed over all the domains our application finds answers to.

6 Specific Techniques

We will be using following approach to try and answer a question, let's assume we have a atomic factoid question…

6.1 Given DBPedia as Knowledge base.

6.2 Input Question

*(say: "**What is the capital of the France?**")*

6.3 Classify the answer type of the question, using the TREC data as training.

*(CLASS: **CITY/LOCATION**)*

6.4 After classifying the question, next task is tokenizing the question and removing stop words; removal of stop words in necessary as later we will be forming chunks of continuous words in our question to form the query keyword.

*(TOKENS: **[What, capital, France]**)*

6.5 After tokenization, next step is to identify the **Resource** in our question, which could be mapped to a DBPedia resource. DBPedia has a finite number of resources; we could check our tokens and identify the DBPedia resource in them.

*(Resource = **France**)*

6.6 Next, we need to find the keyword in the question, which is also the focus of the question. This can be achieved using POS tagging and looking for continuous non-stop words to start with.

*(Keyword: "**capital France**")*

6.7 So now, we know that our resource is **France**; we can pull the entire DBPedia ontology of **France** through a Web Service and look for a ontological classes in resource France that is most similar to the keyword Capital.

*(Resource - **http://www.dbpedia.org/resource/France**)*

6.8 So find the similarity, find the synonyms to capital - **([capital, metropolis, first city])** and the most similar ontological class in **France**.

*(most-similar-ontological-class = **dbpedia-owl:capital**)*

6.9 As a last step, having mapped all the parts of natural query to a SPARQL query, convert the natural text into SPARQL query to get the answer.

(Query = select distinct * where { <http://dbpedia.org/resource/France> <http://dbpedia.org/ontology/capital> >?v })

7 Evaluation Metrices

7.1 As mentioned earlier since this is a domain specific QA agent. We have decided to manually curate a set of 200-500 questions that are uniformly distributed over our domain. Using these questions and their correct answers, we calculate the accuracy and F_Score.

7.2 As a secondary metric, we would be comparing other QA agents like Evi, Wolfram Alpha with our agent, using the same set of questions, to check how close our agent performs w.r.t commercial applications.

8 Work Breakdown Structure

Extraction of TREC data for training question types	Rahul Agarwal
Tokenize, clean, POS tag input question	Pallavi Singh
Identify the resources in the question and convert question into RDF format	Akshay Jagtiani
Retrieve the identified resource from DBPedia	Akshay Jagtiani
Identify the keyword in the question	Pallavi Singh
From the ontology find the most matching/synonymous ontological class to the keyword	Rahul Agarwal
Convert RDFized question into SPARQL query	Pallavi Singh
API call to DBPedia	Akshay Jagtiani
Post Processing of the output	Rahul Agarwal
Web App / UI / etc	Akshay Jagtiani

9 References

Following are the papers/articles we have consulted while building our approach to the QA agent

9.1 https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/nlpreduce/
9.2 http://airccse.org/journal/ijwest/papers/4313ijwest03.pdf
9.3 http://ebiquity.umbc.edu/_file_directory_/papers/591.pdf