Project Initial Report

Weidong Yuan	Shunyuan Zhang	Yazhi Gao	Zijun Shi
weidongy	shunyuaz	yazhig	zijuns

1 Overview

Our question answering system overall is a two-component system that consists of question generation and answering module.The two components collaboratively build a pipeline which first extract question from text and answer them based on the text. Most of the system will be based on information retrieval and classic NLP algorithm approaches. In the following sections, we will discuss the design of the modules, the technical stack of the project and the management of our system.

2 System Design

2.1 Preprocessing

Before preprocessing, we will remove possible URLs in each document.

First, we will segment the continuous articles into separate sentences by identifying the boundaries of the sentences. The sentence splitter from NLTK can be used to accomplish this task.

The second step is to tokenize each sentence into words using the tokenizer from NLTK. Then we may do stemming using the widely-used Porter Stemmer, which is usually done when setting up Information Retrieval systems. However this could be optional because it increases the risk of losing useful information.

Next, we will generate part-of-speech tags using pos tagger from NLTK.

The next step of preprocessing is named entity recognition, which is the process of identifying and classifying elements in text into pre-defined categories such as the names of persons, organizations, locations and so on. To implement, We plan to use the name entity tagger from NLTK.

As a step further, we will treat each sentence in the article as a document. Then we construct the corpus, and import the gensim.models module to adjust the weight from counts to tf-idf.

As an alternative step, we will create a stop-word list. We will not implement stop words elimination at this point since we plan to do stemming and tf-idf. However a list of stop words will be saved for further use.

2.2 Question Generation

We plan to generate two types of questions: Yes/No questions and w-h question (i.e., those starting with where, who, what, when, which, etc.).

There are multiple ways to generate a Yes/No question. One way is to move the modal verb to the front of the sentence and add a question mark at the end. This can be achieved via transducer template solutions.

To generate a w-h question, we can replace the named entity with the corresponding question word, according to the tags of each word. This is similar RDF triples used in Knowledge-based QA system. For a sentence containing name entities we read the tags to build the logical form, from which a question can be generated correspondingly.

For questions to have proper complexity, we will optionally replace certain words in the documents with their synonyms.

To evaluate the quality of the generated questions, we will examine their fluency and reasonableness. In terms of fluency, we will look at grammatical errors and ambiguity of the question. In terms of reasonableness, we can reward questions with good answers, by leveraging our answering system. For a given answering system, the goodness of the question can also be evaluated with candidate answers. For example, a good question must be one that a definite answer can be found from the document. If the entropy over top 5 ranked candidate answers is large, it may be that the question can be interpreted in many different ways.

2.3 Question Answering

2.3.1 Module Design

The answering module in our system is a pipeline of three language processing units. We plan to mix machine learning and information retrieval approaches to comprehensively perform anwsering functionality. The three-stage pipeline consists of candidate retrieval unit , question classification unit and answer generation unit. The candidate retrieval unit retrieves relevant candidate sentences in the Wiki article and then select the most relevant sentence by certain ranking metric for the generation unit to process. The question classification unit classifies the sentence into binary and w-h question categories. The answer generation unit processes the input and then extract the key component of the answer and ultimately generate a grammatically fluent and semantically relevant answer sentence.

2.3.2 Candidate Retrieval

In this stage, we propose two algorithms: Feature space similarity retrieval and searching-based retrieval. For feature space similarity retrieval, we assign the question an sentence embedding vector with weights like tf-idf and apply cosine similarity to the question and article sentences to retrieve the relevant sentence candidates. For searching-based retrieval, we use a miniature search engine indexed on the article to retrieve candidates based on term occurrences. We will evaluate the accuracy of the two algorithms to decide which one to apply the deployable system or we can apply an averaging ensemble of both. From the Leaderboard of candidate sentences, we use an alignment model to select an answer sentence most aligned with question in predicate sentence form. The answer sentence will be passed onto the corresponding answering generation module after classification.

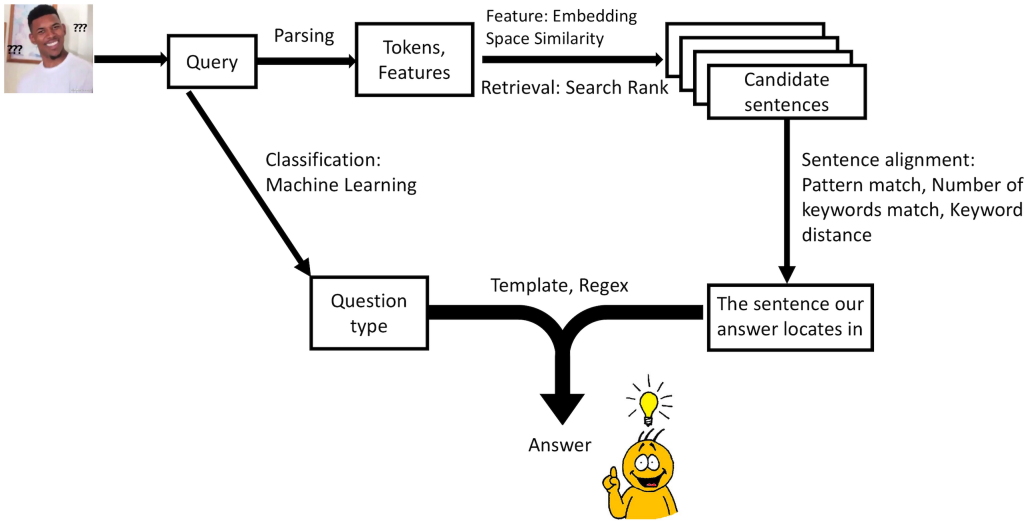
2.3.3 Question Classification

The question classification unit accepts the candidate sentence as input and outputs the sentence's question category for further generation. The questions are categorized into binary and w-hl. Binary questions focus the binary assertion on provided fact like "Is he...","Does he...".W-h questions focus on the entity or activity query on provided fact like "Who is","Where is..".

2.3.4 Answer Generation

The answer generation units are categorized and template based. For binary question, we use alignment model to check the whether the question transformed in predicate form align with the answer candidate sentence. If yes, we output yes for it ,No otherwise. For w-h questions, we use grammatical parser and ENR resolution to analyze to answer candidate sentence. We examine all the specific type of entities the question queries and use alignment to select the right entity as the key answer component. Afterwards we apply a transducer to produce a reasonable sentence.

2.4 Module Illustration



3 NLP resources and libraries

NLTK3: For stemming, sentence segmentation,tokenization,chunking, tagging etc.
WordNet:For classification, finding synonymy, measuring word similarity.
Experimental Data for Question Classification(Xin Li and Dan Roth): Question classification.
Natural Language Toolkit, Contrib Area (NLTK-Contrib): For tranducer and regex checking.

4 Coordination & Project Management

The Project will be managed through private git repository.
We conduct bi-weekly meetings to check system development progress and weekly meetings to remotely communicate with each other on technical issues.

References

[1] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.

[2] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[3] A. van den Bosch and G. Bouma, *Interactive multi-modal question-answering*. Springer Science & Business Media, 2011.

[4] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, Nov. 1995.

[1] [2] [3] [4]