**Project Name:**

CS6320.501 QA System

**Group Name:**

Who is What, When?

**Team Members:**

Jian Liang (jxl170003)

Linqian Zhu (lxz190009)

**Demo Time:**

1:30pm - 1:45pm

# Table of Content

[**Table of Content**](#_fgs4macmc9xo) **2**

[**Project Description**](#_rp8lhlw6esme) **3**

[**Proposed Solution**](#_u47w4pqmlq0y) **3**

[**Full Implementation Details**](#_2lu8a63kg9zr) **3**

[Tool Used](#_vi0wwbnnmg6r) 3

[Architecture Diagram](#_5j6kklknozmt) 4

[Results and Error Analysis](#_5ne78kk7ggr6) 5

[Issue Encountered](#_u07wzfz5enku) 5

[Pending Issues](#_dvw2hnmwce3a) 6

[Potential Improvements](#_itj3xu9ck4x) 6

# Project Description

The goal of the project is to build a Question and Answer System that can return a single sentence and its support document that provides answers to the following types of questions: questions start with WHAT, questions start with WHEN, and questions start with WHO.

However, it is not the goal of this project to collect support documents on its own. The system will only be able to provide answers to questions that the system has a document that contains the answer to the question.

# Proposed Solution

The system will contain two parts. We will be using Elastic Search Engine for the retrieval of the potential candidates. Then, out of those candidates, we will deploy different Natural Language Processing methods to narrow down our candidates and return only one sentence as our final result.

To set up our Elastic Search Engine, we created two indexes: one containing all the individual documents, and another one containing the sentence trigrams of all the documents.

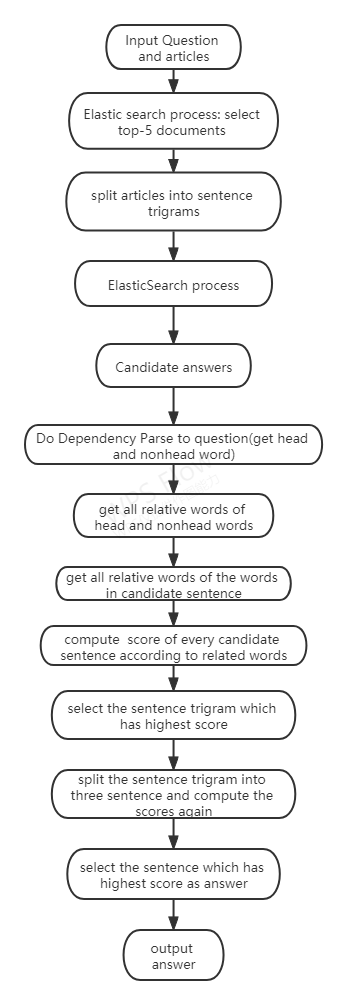
To begin the pipeline, we first filter the top 5 documents that have the highest similarity score to the question. Then we filter the top 10 sentence trigrams that have the highest similarity score within the top 5 documents. Then we do Dependency Parse to the question sentence to get the head and non-head words in it. Next, get all related words of these head and non-head words, including stem, lemma, synonyms, antonyms, hypernyms, hyponyms, meronyms, holonyms. Also, we get all related words of those words in candidate sentences. Then, we compute every candidate’s score by finding the similarity between the question sentence and the candidates. Next, we select the sentence trigram with the highest score. Split the sentence trigram into three sentences and compute the scores again. Finally, we select the sentence with the highest score as the answer sentence.

# Full Implementation Details

## Tool Used

The two mean tools that we used are elasticsearch and nltk library.

## Architecture Diagram



## Results and Error Analysis

The first thing we tried is breaking down documents into individual sentences. In the process, we found out that in a lot of our misses, the sentence that is most similar to the question often doesn’t contain the answer. But instead, the answer is sometimes in the next sentence or the sentence after that. Therefore, instead of creating the index using individual sentences, we decided to break down the documents into trigrams of sentences. And to speed up the search, we first performed a search on the documents to narrow our option down to only the top 5 documents. Then, we retrieved the top 10 trigram sentences. At this point, our accuracy is at around 80%.

After retrieving the top 10 trigrams, we tried many different methods to select the best trigram. The first thing that comes to our mind is to extract all the proper nouns in the question sentence and filter any trigrams that don’t have these proper nouns. After implementing the function, our evaluation shows that the method was only able to filter between one-half to one-quarter of the candidates but it also give us a 20% hit accuracy.

Another heuristic we decide to test out is that out of the 10 candidates, the document where most candidates come from is the one with the correct document id, and we will filter out any candidates that are not from the most prevalent document. That also resulted in too much of a hit to the accuracy.

After many trials, we finally decided to use the method which is described in Proposed Solution Part. After getting the top 10 candidate sentences. We do Dependency Parse to the question sentence to get the head and non-head words in it. Next, get all related words of these head and non-head words, including stem, lemma, synonyms, antonyms, hypernyms, hyponyms, meronyms, holonyms. Also, we get all related words of those words in candidate sentences. Then, we compute every candidate’s score by finding the similarity between the question sentence and the candidates. Next, we select the sentence trigram with the highest score. Split the sentence trigram into three sentences and compute the scores again. Finally, we select the sentence with the highest score as the answer sentence.

After testing, we find the final accuracy is round 70%.

## Issue Encountered

The biggest issue encountered is perhaps time wasted implementing functions that don’t make through into the final pipeline. Some honorable mentions: check pos, extract question type, filter proper noun, choose doc, etc. A lot of ideas that we had and implemented were unable to provide the filtering ability that justifies its hit to the accuracy and are therefore left out of the final product.

## Pending Issues

One-fourth of the misses also missed the document. In another one-fourth of the misses we were able to locate the correct trigram but misses the correct sentence.

## Potential Improvements

We believed that the ability to determine the type of questions that were asked could provide important value to the performance of the system.

Another potential improvement we can add is to manually provide some labels to each of the articles, which could have the potential of narrowing down the document for a problem in a very early stage.