Design Document

1. System Architecture

The intended user for Machine Reading (MR) system is the Question Answering (QA) Systems. The interaction between MR system and QA system is as following: QA system provides set of questions for a domain to MR system and MR system uses these questions to generate a corpus that contains related information about the domain and sends back the explored corpus back to QA system. QA system will rely on the explored corpus to complete QA tasks.

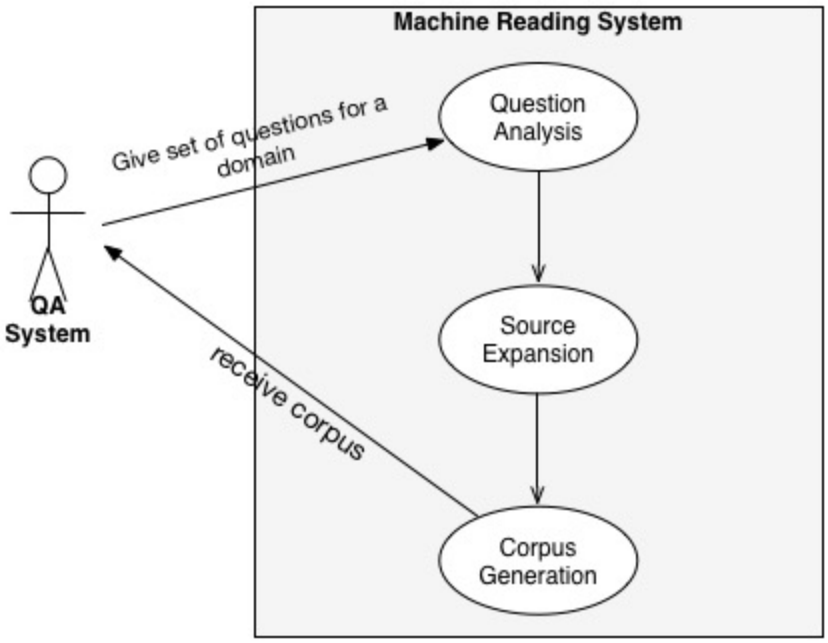

Fig 1. Use-case diagram

2. System Design

The system takes files of questions and answers as input and outputs an explored corpus. The parameters of the MR system, the paths to the files containing the training questions and answer are all given as input to the MR system through a configuration file.

The machine reading process is modeled as states and state transitions. Each state is defined by the set of seeds, a corpus and a snapshot of the knowledge graph. The seeds are the queries [phrases] to be fed into the search engine. The corpus is the set of documents that already has been explored by the machine reading system. The knowledge graph is graph that has entities as nodes and relationships as edges. The corpus, seeds and

knowledge graph are associated with their state. The transition will transit from one state to a new state by updating corpus and seeds in current state.

The system framework diagram is shown in figure 2. Each machine reading system will have a MachineReader responsible for the initialization and state transitions. The MachineReader initialize the rootState and continues doing transitions until it hits the evaluation threshold. For different strategies or QA system, the MachineReader algorithms may change, so as the MRState. So the design is making MachineReader and MRState as the interface and for each specific strategy or QA system implements the interfaces. For this project, the system implements the WatsonMachineReader and WastonMRState. For each state, it has three attributes: seeds, corpus and knowledge graph. The transition function is responsible for the logic operations, such as when to update corpus, when to update corpus based on the evaluation results. The transition function will call build-in APIs, which generate and process the corpus and seeds. The APIs and transition function are loosen-coupled so that different algorithms can be developed independently inside this framework.

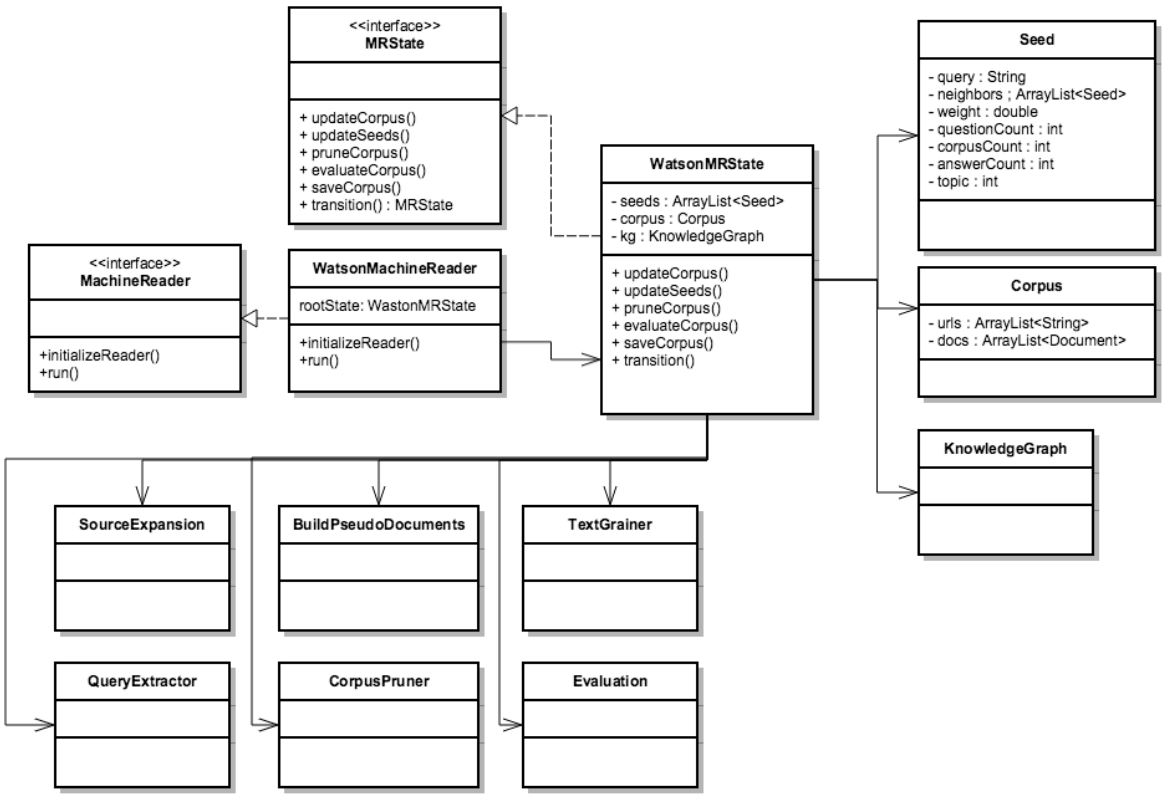

Fig 2. System framework diagram

3. System Workflow

Figure 3 shows the overall system workflow. The QA system gives a set of questions for a specific domain to Machine Reading system. MR system does question analysis first to find out the domain specific information. The next step is to initialize the MR system, such as creating root state. Once initialization is done, the MR system will start state transition procedure. The result of state transition is a new state. The system will evaluate the state status. If the corpus recall is beyond the threshold then the system will come to the final step, corpus generation. The result corpus will be the final output and return to QA system. However, if the corpus recall doesn't meet the requirement, then system will continue running state transition until the corpus reaches to the threshold.

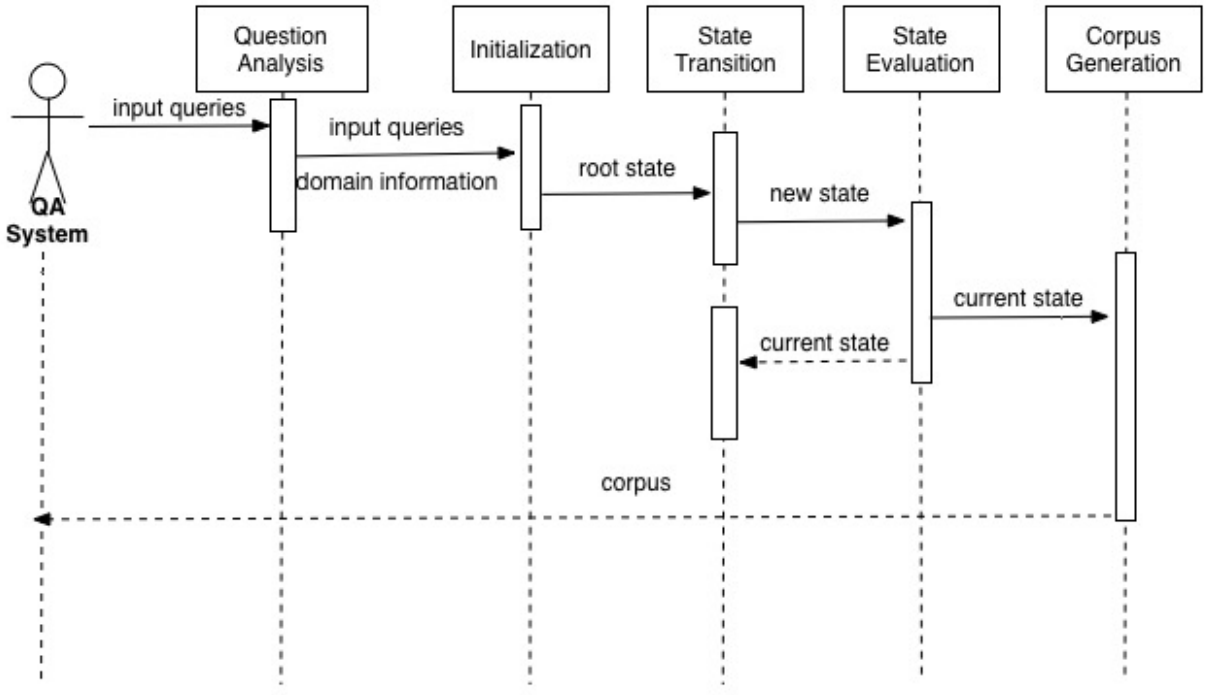

Fig 3. System Overall Workflow

The state transition workflow is shown in figure 4. Firstly, the system will expand the corpus based on the current seeds via search engines, which is called source expansion. After source expansion, the system matches the relative sentences of designated domain from the corpus explored in previous step. Upon finding the relative sentences, the system conduct sentence smoothing. Then the evaluation part will compute the recall of current corpus against the test set. If the recall is beyond the threshold, then the machine reading task completes. The corpus in current state will send back to the QA system as the final result. Also, the final corpus is pruned by system, which removes the irrelevant information. On the other hand, state transition needs continue. The query extraction will generate more relative seeds based on current explored corpus. The building knowledge graph process will be conducted in before evaluation. After these steps, the corpus is changed as well as the seeds

and knowledge graph, which results in a new state. All of the state transition actions will execute again on the new state.

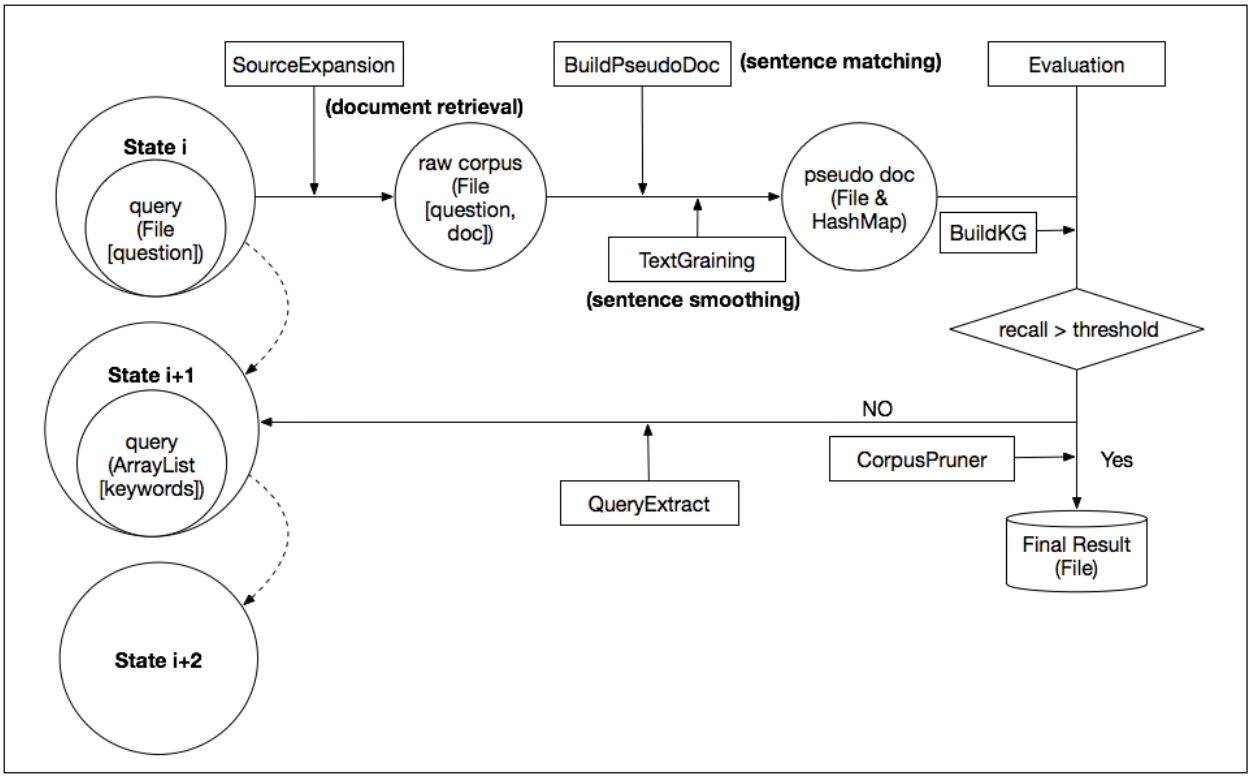

Fig 4. State Transition Workflow

4. Component Design

1. Machine Reader:

This component is responsible for main logic operation, including the system initialization and state transition. This component continues calling state transition function until the corpus reaches the requirement. Each state transition call will generate a new state and these states will be a state sequence.

2. Question Analysis

This component is used to get additional domain specific information from original questions.

3. Source Expansion

This component is responsible for retrieve documents from search engine of given seeds (queries). The number of retrieved document are dynamic parameter, which is rely on the weight of the seed. Also, the system supports multiple search engines, including Bing, Google, DBpedia. It can easily add more search engine with this framework.

4. Build Pseudo Document

After retrieving the documents from web, we need to find the relevant sentences from large volume of the documents. This work is done by BuildPseduoDocument component. The input of this component is the current corpus inside of the state and the output is a file that contains all of the relevant sentences. Different sentence matching algorithms can be easily add into this component independently while not influence other components.

5. **Evaluation**

In this component, we utilize the recall as evaluation metrics. If the recall increases to certain threshold, then we will output the final results to Watson. Otherwise, we continue to do the source expansion.

6. **Text Grainer**

This component is mainly responsible for sentences smoothing. For each relevant sentences, the context of that sentence may also be relevant to the domain. Text grainer can extract the context of found relevant sentences in a predefined or dynamic changing window size based on different algorithms. Different sentence smoothing can add into this component independently while not influence other components.

7. **Query Extractor**

The query extractor component can extract new seeds (queries) from explored corpus in the current state. As more relevant documents retrieved, the wider information system can get from these documents, which means more hints we can extract for this domain. These hints can be used as new queries to generate more relevant documents. This strategy will make the one-time source expansion forward further. Different query extractor algorithms can add into this component independently while not influence other components.

8. **Corpus pruner**

Corpus pruner is used to prune the corpus to a confined size based on different pruning algorithm. One of the requirement of MR system is the the explored corpus should be in a reasonable size. The corpus pruner will remove irrelevant, duplicate and some of the redundant information from corpus to decrease the corpus size. Different pruning algorithms can add into this component independently while not influence other components.

5. Dependencies:

Our system is dependent on some external services like Bing and DBPedia for the data, and the dependency is strong, as those are our data sources at any given point of time.

We need our system to have an active internet connection during the training time.

We also depend on the current versions of Stanford parser and NLTK and hope that they have a good parsing performance.

7. Assumptions:

We assume that the document quality from our data sources is actually really good, and we look at the top document for every retrieval, assuming that to be the best document qualitatively. We assume that the data sources are up and running at every point of time, as long as we have an active internet connection. For now, we assume that our system has no bound on time and memory and just focus on improving the BAR score.

8. External tools, resources and services:

1. Bing : We used the BingSearch API to fetch results from Bing for a given query
2. DBPedia : We used DBPedia to help us with source expansion
3. NLTK: To perform string operations like Tf-Idf scoring.
4. Stanford parser: We used Stanford parser to recognize Named Entities, Noun and Verb from the sentence.
5. Watson: We ran our implementation on the Watson instance and reported the results we got.