CMU Multiple-choice Question Answering System at NTCIR-11 QA-Lab

Di Wang
diwang@cs.cmu.edu

Leonid Boytsov
srchvrs@cs.cmu.edu

Jun Araki
junaraki@cs.cmu.edu

Alkesh Patel
alkeshku@cmu.edu

Jeff Gee
jgee1@cs.cmu.edu

Zhengzhong Liu
liu@cs.cmu.edu

Eric Nyberg
ehn@cs.cmu.edu

Teruko Mitamura
teruko@cs.cmu.edu

Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA

ABSTRACT

We describe CMU's UIMA-based modular automatic question answering (QA) system. This system answers multiple-choice English questions for the world history entrance exam. Questions are preceded by short descriptions providing a historical context. Given the context and question-specific instructions, we generate verifiable assertions for each answer choice. These assertions are evaluated using several evidencing modules, which assign a plausibility score to each assertion. These scores are then aggregated to produce the most plausible answer choice. In the NTCIR-11 QALab evaluations, our system achieved 51.6% accuracy on the training set, 47.2% on Phase 1 testing set, and 34.1% on Phase 2 testing set.

Team Name

CMUQA

Subtasks

QA-Lab Center Exam (English)

Keywords

Question Answering, Fact Validation, Multiple-Choice Question

1. INTRODUCTION

In the Center Exam task of the NTCIR-11 QALab, participants need to develop a system that can *automatically* answer multiple-choice questions for the world history entrance exam. History questions are selected from archives of the the National Center Test for University Admissions in Japan. The original questions are in Japanese, but our team used an English translation provided by NTCIR-11 QA-Lab organizers (in an XML format).

Most commonly, students are asked to choose a correct statement with respect to given background information and an underlined portion of the text. The other types of questions are: "fill in the blanks", "choose the right combination of correct and incorrect statements," and "analyse an image provided with the question". Question types are explicitly indicated in the input XML file.

To select a correct answer from multiple answer choices, the system is supposed to leverage external knowledge sources (such as historical Wikipedia articles) as well as contextual information provided with the question. Participants can use any textual or knowledge sources, but they should not retrieve the actual answer keys by matching provided questions against questions from the National Center Test archives (the already provided answer keys can be used for training). In that, Japanese task participants are given Japanese high school textbooks on world history, which are annotated with the world history ontology.

The rest of the paper is organized as follows: Sections 2 through 4 describe the overall system architecture as well as individual components. Experimental results and error analysis are prevesnted in Section 5. In Section 6 we conclude the paper with the discussion of future work.

2. SYSTEM ARCHITECTURE

Our modular QA system is implemented in the form of an NLP pipeline on top of the UIMA framework[1]. There is a special component that reads an input XML file and generates an analysis object that "travels" from one processing module to another over a virtual conveyor belt (hence, the name *pipeline*).

A modular conveyor approach–readily supported by UIMA–is convenient for experimentation and rapid development, because it simplifies the addition of new processing modules and replacement of existing modules by compatible ones.

In particular, given a short historical description and question instructions, we generate verifiable assertions for each answer choice, which can be scored by multiple evidencing modules. To select the most plausible answer choice, we need to aggregate the scores. Again, multiple aggregation approaches can be used.

The major pipeline modules (in the order they process input data) are:

Collection Reader parses the input XML with questions and optionally an input XML file with gold standard data. The Collection Reader extracts short historical descriptions, generic and question-specific instructions, as well as answer choices. This information is then stored in the newly created

[1]`http://uima.apache.org/`

analysis object (represented by a UIMA CAS object).

Assertion Generator reads question-related information provided by the Collection Reader. It then generates an assertion, which can be validated by downstream processing modules. Each answer choice can be associated with one or more assertions. There are manually implemented assertion-generating procedures for most common question types.

An assertion typically contains an answer-choice text concatenated with substrings extracted from the question-specific context. It is possible, e.g., when students are asked to fill in the blanks, for a question to reference portions of the underlying historical description. In such a case, these references are resolved by inserting referenced portions of the description into the text of the assertion.

Evidencers assign a plausibility score to each assertion. The goal is to ensure that correct assertions generally have higher scores than incorrect ones. We implemented a diverse pool of evidencing components, which we describe in Section 3.

Answer Selector aggregates evidencing scores to produce the final score for each answer choice. If we are asked to select a correct answer, the answer choice with the highest final score is selected. Otherwise, we select an answer associated with the lowest final score. We employed several learning-to-rank and voting approaches to evidence aggregation, which we describe in Section 4.

Evaluator "consumes" pipeline's output, computes performance metrics, and (optionally) submits the output to an evaluation web service. The evaluation web service persists experimental data and metadata from our UIMA pipeline, making them easily accessible through the web user interface, and provides an interface to carry out an error and overlap analysis. Last but not least, this evaluation web service supports a leave-one-out cross-validation to evaluate performance of learning-to-rank based answer selectors.

To facilitate future collaborative efforts in the world history exam task, we open sourced our baseline system[2]. It includes a type system, a collection reader, a document retrieval based evidencer, and an evaluator which can also serve as a modular software platform for evaluating components' performance.

3. EVIDENCING MODULES

3.1 Semi-Phrasal Queries using Wikipedia, Gutenberg Collection, and Wikibooks

In this approach, we indexed mostly complete documents and ran semi-phrasal queries. Such queries do not enforce a strict phrasal match but rather boost scores of documents where query words appear close to each other. More specifically we used the following heuristics, which were implemented using the **ExtendedDisMax** parser of the SOLR search engine:

- Scores of the documents were boosted if query words appeared within a text window of a given size. The size of the window was proportional to the number of query words. We found that multiplying the number of query words by 1.5 or 2 results in good retrieval quality.
- Rather than relying on a score of a single top-ranked document, we computed a decaying sum (as done by Gondek et al. [4]). In the decaying sum, a weight of the document score is an exponentially decreasing function of a document rank (in our case, $2^{-\text{rank}}$).
- We run several queries with varying degrees of fuzziness. The strictest query required that all query words were present in a document, the next strictest required only 80% of them to be present, and so on. In the simple-sum approach, scores obtained for different degrees of fuzziness were summed up to obtain an ultimate assertion score. The scoring schema was BM25.

[2]`https://github.com/oaqa/ntcir-qalab-cmu-baseline`

Our approach relies on several parameters, which were manually tuned using two different test collection: the original test collection provided by NTCIR organizers and the collection created by us. To prevent overfitting, we tried to select parameters that worked well for both collections. The collection created by our team contained 200 questions and was based on the materials of the Regents Exam Preparation Center (see Section 5.1.2 for details).

Overall, we indexed three collections using SOLR:

- A subset of historical Wikipedia documents (about 400 thousand documents);
- A subset of historical books from the Wikibooks [3] collection (158 documents);
- A subset of historical books from the Gutenberg collection (about 23 thousand paragraphs from about 500 books), each of which was split into paragraphs.

A simple rule-based classifier was used to select historical articles. Namely, we checked if the list of categories contained one of the manually selected keywords, mostly "history" and "historical". Article text contains Wikipedia markup tags, most of which were deleted using simple regular expressions. After basic cleanup, we indexed complete documents.

In the case of the Gutenberg collection, the list of books was created manually. To find history-related books, we used site-restricted Google-search and browsing of the Gutenberg book catalog. For example, we used "history", "history textbook", "modern history", "history of China", and "history of Japan".

3.2 Passage Ranking using Wikipedia

In this evidencer, we have a document and a passage index. Scores are obtaind in three stages : document retrieval, passage retrieval, and passage reranking. Both the document and passage retrieval system employ the TF-IDF approach [9].

3.2.1 Document Retrieval

The document index contains all Wikipedia articles that are not redirects. These articles are indexed using Apache Solr[4]. To maximize recall, we create queries by OR-ing all assertion words and retrieve at most N_d (method parameter) documents.

[3]`https://en.wikibooks.org`

[4]`http://lucene.apache.org/solr/`

3.2.2 Passage Retrieval

Retrieved documents are divided into sentences using Stanford CoreNLP[5]. Then, a sliding window approach (N_s sentences make one passage) is used to combine sentences into passages [7, 2]. Next, passages are added to the passage index and passages with the 10 highest scores are retrieved.

On 2009 data set, the best separation of scores between true and false assertions was achieved for $N_s = 4$. Hence, we used $N_s = 4$ for all other data sets as well.

3.2.3 Passage Reranking

Following the idea of [1], we hypothesize that effectiveness of the passage retrieval, which relies on a bag-of-words model (sim_{BOW}), can be improved by combining the bag-of-words model with an annotation similarity model (sim_{ANN}). The latter model relies on computation of the overlap between passages and assertions with respect to n-grams and named entities.

We considered the following three options to compute the n-gram overlap (NO) between assertion a and passage p:

$$\begin{aligned} sim_{NO1}(a,p) &= \frac{|ngrams(a) \cap ngrams(p)|}{|ngrams(p)|} \\ sim_{NO2}(a,p) &= \max_{s \in p} \frac{|ngrams(a) \cap ngrams(s)|}{|ngrams(s)|} \\ sim_{NO3}(a,p) &= \max_{w_a \in p} \frac{|ngrams(a) \cap ngrams(w_a)|}{|ngrams(w_a)|} \end{aligned}$$

where $ngrams(t)$ denotes a set of n-grams in text t; s denotes a passage sentence; w_a denotes a text window containing the same number of tokens as the assertion a. In that, we use only unigrams, bigrams, and trigrams.

Similarly, we formulate the following three options to compute the named entity overlap (NEO) between assertion a and passage p:

$$\begin{aligned} sim_{NEO1}(a,p) &= \frac{|NEs(a) \cap NEs(p)|}{|NEs(p)|} \\ sim_{NEO2}(a,p) &= \max_{s \in p} \frac{|NEs(a) \cap NEs(s)|}{|NEs(s)|} \\ sim_{NEO3}(a,p) &= \max_{w_a \in p} \frac{|NEs(a) \cap NEs(w_a)|}{|NEs(w_a)|} \end{aligned}$$

where $NEs(t)$ denotes a set of named entities found in text t. Entities were extracted using Stanford CoreNLP.

For NO, the combined similarity score between a and p is equal to:

$$\begin{aligned} score(a,p) &= sim_{BOW}(s,p) \times sim_{ANN}(a,p) \\ &= \textit{TF-IDF}(a,p) \times (1 + \alpha\ sim_{NO}(a,p)) \end{aligned}$$

where α is a weight of the n-gram similarity model. The same approach was used to combine the bag-of-words and the NEO similarity scores.

For the training set, we found that an optimal performance was achieved when we used only sim_{NO3} with $\alpha = 0.1$. This option was also used for the final submission.

3.3 Source and Query Expansion

To mitigate the mismatch between terms in assertions and text collections, we use a combination of two strategies: a source/corpus and a query expansion. In particular, we treat assertions as queries and generate all possible query expansions as described below in Section 3.3.2. Then we obtain their scores as well as the score of the original assertion using the retrieval method described in Section 3.1. Additionally, we obtain the score of the original assertion in the expanded corpus.

If the difference between the score for the original assertion and the highest score of an expanded assertion is positive, this difference is used to boost assertion's original score. However, if this difference is negative and the assertion score for the expanded corpus is smaller than assertion's original score (for the original corpus), then this difference is used to decrease the original score.

3.3.1 Source Expansion

Source expansion [14] is a variant of multi-document summarization. It is a simple, yet effective, technique to automatically expand documents belonging to one corpus using text snippets (nuggets) from another corpus. Source expansion includes the following steps:

1. Given a document from the original collection (called a *seed* document), construct a query using keywords from the seed document (e.g., using title keywords);
2. Use this query to retrieve a list of documents from a large corpus;
3. Extract text passages (also called *nuggets*) from the retrieved documents;
4. Compute similarity scores between the seed documents and retrieved nuggets (using various features such as term overlap) and discard text nuggets with low scores;
5. Append nuggets to the seed documents (similar or identical nuggets may be eliminated at this stage).

The quality of the corpus can greatly affect performance of a retrieval-based QA system. For example, retrieval using a small domain-specific corpus may have low recall. Employing a larger corpus may improve recall at the expense of a substantial decrease in precision. Expanding this small domain-specific corpus using the larger one, may result in a more balanced text collection that provides a better recall-precision trade off than either of the corpora. In particular, in the IBM Watson QA system, the use of source expansion alone lead to a substantial improvement in answer accuracy (3.7% in the end-to-end pipeline).

In our system, Wikipedia articles were expanded by constructing queries from assertion text and submitting these queries to Bing. For example, given an incorrect assertion "Yan Zhenqing is a calligrapher representative of the *Song period*" we obtained the following nuggets:

- He is the best calligrapher in the *Tang Dynasty*, bar none;
- Yan Zhenqing was a leading Chinese calligrapher and a loyal governor of the *Tang Dynasty*;
- Yan Zhenqing was a prominent Chinese calligrapher of the *Tang Dynasty*, and remains one of the most famous and emulated calligraphers today;

[5]`http://nlp.stanford.edu/software/corenlp.shtml`

3.3.2 Query Expansion

In our system, query expansion consists in identifying assertion entities and replacing them with similar ones. For example, given the assertion "Yan Zhenqing is a calligrapher representative of the *Song* period", we generate expanded assertions such as "Yan Zhenqing is a calligrapher representative of the *Tang* period" and "Yan Zhenqing is a calligrapher representative of the *Qing* period".

To generate such neighbor assertions, we first identify entities using DBPedia Spotlight [10]. In that, entities with confidence scores smaller than 0.5 are discarded. For each entity, we generate a new assertion by replacing this entity with the most similar neighbor entity. For example, for the entity *Tang_Dynasty* the two nearest neighbors are *Song_Dynasty* and *Sui_Dynasty*.

To find similar entities, we employ already existing vector space mappings for 1.4 million Freebase entities [6]. The mappings were originally created using word2vec[12]. Closeness of vectors is measured using the cosine similarity.

DBPedia and Freebase cover somewhat different subsets of Wikipedia entities and use different identifiers to denote the same entity. Yet, both data sets have a good overlap and there is a publicly available mapping between DBPedia and Freebase identifiers[7].

3.4 Temporal Scorer

3.4.1 Types of Temporal Assertions

In many multiple-choice exams, candidate answers are constructed to be as deceptive as possible and seem to be all factually correct upon the first review. However, upon closer investigation, it is often possible to identify temporal clues that can be used to discard these incorrect answers.

For example, the statement "Columbus discovered America in the 16th century" is incorrect because this discovery happened a century earlier. However, the assertion "Columbus discovered America" alone (obtained by removing temporal information) is a valid statement. Therefore, comparing the actual discovery time against the time specified in the assertion is crucial to invalidating the original incorrect statement.

In this section, we describe a simple scoring pipeline to evaluate factual assertions based on keywords as well as on the temporal context. There are three main ways in which a statement can be constrained by time:

- A time period or date is explicitly specified;
- A time period is indicated via a reference to a historical event, such as "The Bolshevik Revolution".
- There is no temporal reference specified in a statement. However, temporal logic can be applied to the named entities in the statement.

Statements from the third category may indicate relations among multiple historical events. For example, a statement, "During the first World War, Abraham Lincoln instructed ..." can be determined to be incorrect because World War I corresponds to a period of time in which Abraham Lincoln was not able to "instruct" (he was deceased).

[6]https://code.google.com/p/word2vec/
[7]http://wiki.dbpedia.org/Downloads2014#links-to-freebase

Applying temporal logic to resolve interactions between historical entities requires a fundamental understanding of what type of relations infer what types of temporal constraints. For example, "Abraham Lincoln addressed Barrack Obama," contains an overlapping temporal constraint, where Lincoln and Obama must co-exist in the same time period. However, "Abraham Lincoln inspired Barrack Obama" is another case of overlapping historical entities that implies a different, more relaxed temporal constraint, in which Lincoln can additionally precede Obama.

In this work we do not attempt to account for all the complex interactions among temporal constraints, and instead, rely on heuristics.

3.4.2 Pipeline Architecture

In our experiments, we use a passage-based Wikipedia index. A passage length is two sentences. Passages are created using a standard sliding-window approach described in Section 3.2.2.

An assertion score is computed in three steps:

- Extraction of historical entities from the assertion text using DBPedia Spotlight [10].
- Scope-constrained passage retrieval.
- Filtering based on temporal constraints.

In the first step we identify historical entities contained in the assertion text. These entities define titles of potentially relevant Wikipedia pages and, therefore, the scope of the passage search.

After the search scope is defined, we retrieve K passages with the highest cosine similarity scores. These scores are computed between term frequency vectors for the assertion and the passage texts (terms are stemmed).

The scope-limiting heuristics of the passage retrieval is based on the assumption that the most important information about a particular historical figure/event is consolidated in its main Wikipedia page. For example, we assume that the main Wikipedia page for Abraham Lincoln includes all the (important) historical facts about Abraham Lincoln that other Wikipedia pages may provide.

Passages obtained in the previous steps undergo additional filtering based on how well they satisfy temporal constraints. To this end, we first identify all explicit temporal references (using the SUTime library [3]) and convert them into temporal constraints. For example, if an assertion begins with "From the 16th century until the 17th century...", we restrict dates to be in the range from 1500-01-01 to 1699-12-31. Eventually, we keep only passages that satisfy at least one temporal constraint and discard everything else.

3.5 Semantic Similarity based on Wikipedia graph

This scorer relies on a bag-of-entities model to represent assertions and context information. For example, the assertion "**Ouyang Xiu** and **Su Shi** are writers representative of the **Tang period**" is associted with the following bag-of-entities:

{**Ouyang Xiu**, **Su Shi**, **Tang period**.}

This assertion is incorrect because the entitiy "Ouyang Xiu" is not related to the Tang period and is, therefore, only weakly related to the other entities in this example. In contrast, if an assertion is correct, we expect entities to be semantically related to each other [5].

To compute similarity, we use a graph of Wikipedia entities, in which entities are nodes. Each node, therefore, is represented by a Wikipedia article. If there is a hyperlink from article A to article B or vice versa, the graph contains an *undirected* edge connecting A and B. A *weight* of this edge is equal to the overall number of hyperlinks from A to B and from B to A.

A common assumption is that "connectedness" of graph nodes is a good proxy for semantic similarity. More specifically, we expect that it takes fewer edges with larger weights to connect similar entities than dissimilar ones.

To quantify node similarity, we use the Personalized PageRank [6]. The classic PageRank [13] values are computed as a stationary distribution of the following random walk: A user either teleports to a randomly selected node with the probability α (in our experiments $\alpha = 0.85$), or follows a randomly chosen outgoing link with the probability $(1 - \alpha)/N$, where N is the number of outgoing links.

The personalized PageRank differs only in that the surfer always teleports to the same seed node rather than to a randomly chosen arbitrary one. Note that in our case, the graph is undirected, i.e., all edges are considered to be outgoing links. In that, a probability to randomly follow an outgoing link is proportional to the normalized weight of the corresponding edge.

Given a context and an assertion, we first build an entity subgraph containing only entities from the assertion and the context. For each entity present in the assertion, we then run the personalized PageRank algorithm using this entity as a seed node. Next, a score of the entity is computed as a weighted sum of the average, minimum, and maximum PageRank value over subgraph entities. Finally, we average these scores over all assertion entities to obtain the overall assertion score. For questions without the context, an assertion score is set to zero.

4. AGGREGATING EVIDENCING SCORES AND SELECTING ANSWER

Even though individual evidencing components are not perfect, it may be possible to build a better system by combining them. One straightforward approach would be to normalize evidencing scores and apply point-wise learning-to-rank methods. However, scores from retrieval-based evidencing component are query-specific. For example, assertions with many words tend to generate larger retrieval scores than short assertions containing only a few words. Thus, it is hard to normalize scores properly. This is exacerbated by the fact our training data is scarce. However, we discovered two general approaches that can effectively combine scores and improve system performance: voting and list-wise learning-to-rank.

4.1 Weighted Voting

A simple voting scheme selects an answer choice that is supported by the largest number of evidencing components. In a weighted voting scheme, each component has a voting weight. When an answer is selected by a component, we add the voting weight of this component to the score of the answer. In the end, we select the answer with the largest total score.

Voting can act as a boosting algorithm. It can be used if evidencers generate answers without providing confidence scores and, consequently, one cannot use more sophisticated boosting methods. Consider an example of three components, one of which has a superior performance. A combination approach (which we denote as **3Voters**) can rely on the following heuristic: use the answer selected by the strongest component, unless the two weaker components both agree on some other answer choice. A voting schema that implements this algorithm is the one where the strongest voter has weight 1.5, while the two weaker voters each gets a weight one.

Imagine that (1) n^+ is the number of questions when the strongest component is wrong, but the weaker components are jointly correct; (2) n^- is the number of questions when the strongest component is correct, but the weaker components are jointly wrong. If $n^+ > n^-$ we can hope that on the previously unseen set the described combination of three evidencers will perform better than any evidencer separately (especially if the query sample is large). Thus, we can use the results of the overlap analysis to create a weighted voting scheme that has a better average performance than any of the voters alone.

More generally, we can create a voting scheme that maximizes accuracy on the training data. To this end, we can use a grid search. When there are a lot of components, a grid search can be expensive and other weight-selection approaches should be used [8, 15].

4.2 Point-Wise Learning-to-Rank

To use a standard classifier, we reduce an answer-selection problem to the problem of binary classification by training a separate classifier for each possible answer choice k. The feature vector is binary: the i-element is equal to one if the evidencer i selects answer k (and is zero otherwise).

At test time, we run the classifier for each answer choice, and obtain classification confidence. If we are asked to select a correct answer choice, an answer choice corresponding to the highest confidence value is selected. Otherwise, we select the answer choice with the lowest confidence value. Such a classifier represents a point-wise learning method.

We evaluated performance of several standard Weka algorithms, including AdaBoost, Logistic Model Trees (LMT), Logistic Regression, LogitBoost, Random Forest, and linear SVM using leave-one-out cross validation. Logistic regression performed best and we used for our official submissions (we denote this method as **LogisticRegression**).

4.3 List-Wise Learning-to-Rank

We employ a list-wise learning-to-rank method, where performance is tuned using the classic coordinate ascent. It is implemented as a part of the RankLib library[8]. We use this algorithm to optimize parameters of a linear ranking model (the overall score is a weighted combination of confidence scores produced by individual evidencers, the answer choice with the largest overall score wins). This method is denoted as **CoordAscent**.

The optimization algorithm iteratively carries out a series of one dimensional searches. In each iteration, it varies one parameter (to optimize the target function) while keeping other parameters fixed. Although this method may con-

[8]`http://sourceforge.net/p/lemur/wiki/RankLib/`

Table 1: Summary of question types in the training set

Question Type	Count
True/false statement	107
Fill-in-the blanks	31
Combination of true/false statements	6
Image analysis	7
Chronological ordering	2
Total	153

verge slowly, it directly optimizes the objective function (such as the classification error), rather than a surrogate differentiable loss function that only approximates (or upper-bounds) the true objective function.

Metzler and Croft showed that in the context of information retrieval this simple method outperforms other learning methods [11]. We hypothesize that coordinate ascent is especially useful when training data is scarce.

5. EXPERIMENTS

5.1 Datasets

Our main data set was provided by NTCIR-11 organizers. Additionally, we created another set of questions using history quizzes freely available online. Below, we describe both data sets in detail.

5.1.1 Center Exam Training Dataset

This data set–provided by NTCIR-11 QA-Lab–contains exam questions and respective answer keys for the Central Entrance Exam that took place in 1997, 2001, 2005, and 2009. Overall, there are 153 questions of five types (see Table 1). Two questions were ignored, because organizers did not provide answer keys. Most commonly, students are asked to identify either one correct or one incorrect statement. The next most frequent question is fill-in-the-blanks. We can generate assertions for both types of questions. However, this is not possible for image analysis questions. Additionally, assertions were not generated for chronological ordering questions as well as for questions without answer keys. Thus, our system only generated valid assertions for 143 questions.

5.1.2 PREP: Regents Exam Preparation Center Data

The PREP collection–compiled by our team–has 200 questions. It is based on the materials of the Regents Exam Preparation Center (see Section 5.1 for details) [9] By design, all these questions are simple assertions, i.e., there is no additional context. We had to edit and cull out some of the questions to ensure that they represent simple assertions.

Consider, for example, a question that asks about the similarities between the Imperial Russia and the Japan during the Meji restoration period. Unless there is a text with a discussion on this topic (explicit comparison between Russian and Japan during given periods), it is not possible to answer such a question using a simple retrieval-based method.

[9] `http://www.regentsprep.org/`

Table 2: Methods' performance on the training set

Method	Accuracy
`AlwaysFirst`	0.258
`SimpleWikipediaSubset`	0.371
`SemiPhrasal (Wikipedia)`	0.430
`SemiPhrasal (Wikibooks)`	0.225
`SemiPhrasal (Gutenberg)`	0.318
`PassageRanking`	0.444
`SourceExpansion`	0.444
`Temporal`	0.397
`SemanticGraph`	0.385
`3Voters`	0.464
`LogisticRegression`	0.503
`CoordAscent`	0.516

Note: accuracy is computed for 151 questions with answer keys.

Apart from removing or editing questions that are not simple assertions, no further effort was made to select questions in a way that would deliberately decrease or increase performance of our system.

5.2 Baselines

Two simple evidencing scorers are used as baselines:

- `AlwaysFirst` is a method that always selects the first answer. Because most questions have four answer choices where a correct choice is essentially random, this baseline should have the accuracy close to 25%.
- `SimpleWikipediaSubset` relies on a small subset of historical articles filtered from a Wikipedia XML dump (11 thousand articles). The retrieval algorithm uses assertions as fully disjunctive queries (using the default scoring method of SOLR).

5.3 Results

In Section 3, we presented several evidencing approaches to guess the right answer choice. We evaluated performance of each individual component as well as performance of several aggregating approaches described in Section 4. Results are summarized in Table 2.

On the Center Exam training data set, all methods outperform the baselines. There are individual evidencing components that can correctly answer up to 44.4% of all questions. An even higher accuracy (more than 50%) can be achieved by aggregating evidencing components.

We also evaluated performance on the PREP data set. All components performed slightly worse compared to the Center Exam data set. This is a somewhat surprising result, because we expected our methods to perform better on simple assertions. However, they may have also benefited from context information, which is present only in the Center Exam data set, but not in PREP.

Table 3: Methods' performance in the official evaluation

CMUQA_EN Run ID (Method)	Accuracy	
	Phase 1	Phase 2
`FA_01 (CoordAscent)`	0.472	0.317
`FA_02 (LogisticWeighted)`	0.444	0.341
`FA_03 (3Voters)`	0.417	0.292

Table 3 shows the accuracy of official runs submitted by our team. The official evaluation had two phases, and, for both of them, we achieved the highest score among English subtask participants. When comparing English and Japanese submissions, we ranked third in phase one. In phase two, we ranked sixth (out of ten participants). However, it is important to note that English and Japanese subtask cannot be directly compared for two reasons:

1. Japanese participants had access to high school textbooks on world history, which were annotated with the world history ontology, while English subtask participants had to rely on Wikipedia or other resources that they had to collect on their own;

2. There could have been noise introduced during translation from Japanese to English.

5.4 Error Analysis

5.5 Assertion Generation

Performance of the assertion generator greatly affects performance of downstream evidencing components. Generating a meaningful verifiable assertion is a complex natural language generation task. Hence, we resort to simple heuristics and mostly concatenate an assertion text with question-specific instructions. Oftentimes this procedure generates a botched text that is not only nonsensical from a human perspective, but also not amenable to NLP tools. Because few of our assertions contained proper English sentences, we used only simple retrieval methods for assertion verification. One good example is Question 3 in 1997 data set (see Figure 1).

Quite often, an assertion text contains referential personal pronouns. We did not try to replace these pronouns. As a result, many of generated assertions are not sufficiently specific. For example, in Question 19 from 2001 data set, we generate the following non-discriminating assertions:

1. It implemented the marshall plan. economic and financial aid;

2. It implemented the dawes plan. economic and financial aid;

3. It issued the hoover moratorium. economic and financial aid;

4. It implemented the young plan. economic and financial aid.

We found 13 questions where better assertions could have been generated if references were resolved properly.

Context: *"many countries not only in Europe, but also in Asia, Africa, and America became involved in the war for various reasons, so it truly was a world war, as the name suggests."*

Instruction: *"In regard to the underlined portion (3), from (1)-(4) below, choose the one correct option that is the name of a country that participated in the war on the allied side."*

Choices:
(1) *Switzerland* (2) *Italy* (3) *Belgium* (4) *Bulgaria*

Generated Assertion for choice (1): *"of country participated war on allied side many countries not only in Europe, but also in Asia, Africa, and America became involved in the war for various reasons, so it truly was a world war, as the name suggests. Switzerland"*

Better Assertion: *"Switzerland participated in a world war on the allied side."*

Figure 1: Question 3 in 1997 data set

5.6 Retrieval-Based Validation

Our evidencing algorithms heavily rely on retrieval scores, which depend on the frequency and proximity of assertion terms in a document. Even though these scores are decent predictors of assertion correctness–the accuracy can be as high as 40%–they ignore complex syntactic and semantic phenomena. Retrieval-based evidencer failures are often an artifact of a specific weighting algorithm. In particular, longer assertions or assertions that have many terms with larger IDF values tend to generate larger scores regardless of their correctness.

Figure 2 shows Question 11 from 1997 data set. The correct answer choice is four, but both the Wikipedia-based semi-phrasal retrieval component (Section 3.1) and the passage retrieval component (Section 3.2) generate a substantially higher score for the answer choice three. The answer choice three is, however, incorrect, because the first Oil Shock was due to 1973 oil embargo and the second one was due to the decreased oil output in 1979. In that, the assertion for the third answer choice has three extra words, which is one of the reasons why it has a higher score. It may be possible to improve the accuracy by taking assertion lengths into account, especially we have more training data, but we have not been able to do this yet.

5.7 Entity Detection

The temporal scorer 3.4 carries out a scope-constraint search, where scope is defined as a subset of Wikipedia articles whose names can be found in an assertion text. Entities are extracted using DBPedia Spotlight [10]. In that, we could find cases where limiting the scope of the retrieval affects our results. Consider for example Question 37 in 1997 data set:

Correct Candidate Answer: *"Ignatius of Loyola formed the Society of Jesus in 1534, and sought to restore Catholic influence"*

Context: ...Moreover, in order to create peace, not only military factors, but also (9) complex problems such as economic factor came to be considered.

From 1-4 below, choose the one sentence that is correct in regard to the underlined portion (9):
1) The European Economic Community (EEC) promoted economic integration focused on the UK.
2) The North-South divide is a term that indicates the differences between the United States of America and the Soviet Union in terms of their political and economic systems.
3) The oil crises (Oil Shock) occurred because of a reduction in the price of crude oil and oversupply by the Arab oil-producing countries.
4) Asia saw the emergence of the newly-industrializing economies (NIES), which achieved rapid economic growth.

Figure 2: Question 11 in 1997 data set

Ideal Evidence Pages: *"Ignatius of Loyola"* and *"Society of Jesus"*.

However, for a chosen confidence threshold, the only entity detected by DBPedia Spotlight was *"Jesus"*.

6. CONCLUSIONS AND FUTURE WORK

We have developed a UIMA based question answering system to automatically answer multiple-choice questions for the entrance exam in world history. Our system is a modular software platform that can be used for future evaluations. The system includes assertion-validation components, whose combination was able to correctly answer 44% of questions in Phase 1 and 34% of questions in Phase 2. Our best systems were substantially better than baselines.

We believe that further improvements can be achieved by employing the following:

- Standardization - We need to be able to detect references to historical events in text and represent the correspdong events in a standard format. This format should allow us not only to establish equivalence of differently represented events, but also to verify if an event happened within a specified time frame. Such a standardization may require some external knowledge-bases such as DBPedia.

- Coreference Resolution and Disambiguation - Standardization often requires coreference resolution, because historical events are seldom referenced using their complete names. For example, in the Wikipedia article "Abraham Lincoln", "the war" often refers the Civil War. Nor are these short names unique. This is the common issue with revolutions ("Bolshevik Revolution" vs. "October Revolution") and wars ("World War II", "Second World War").

- Implicit Temporal Reference Resolution - Standardization requires handling of implicit temporal references, which can be considered a special case of coreference resolution.

7. REFERENCES

[1] J. Araki and J. Callan. An annotation similarity model in passage ranking for historical fact validation. In *Proceedings of SIGIR 2014*, pages 1111–1114, 2014.

[2] J. P. Callan. Passage-Level Evidence in Document Retrieval. In *Proceedings of SIGIR 1994*, pages 302–310, 1994.

[3] A. X. Chang and C. D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740, 2012.

[4] D. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. A. Duboue, L. Zhang, Y. Pan, Z. Qiu, and C. Welty. A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research and Development*, 56(3.4):14–1, 2012.

[5] X. Han. Collective Entity Linking in Web Text : A Graph-Based Method. In *Proceedings of SIGIR 2011*, pages 765–774, 2011.

[6] T. H. Haveliwala. Topic-sensitive PageRank. *Proceedings of WWW 2002*, pages 517–526, 2002.

[7] M. Kaszkiel and J. Zobel. Passage Retrieval Revisited. In *Proceedings of SIGIR 1997*, pages 178–185, 1997.

[8] L. I. Kuncheva and J. J. Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2):259–275, 2014.

[9] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[10] P. Mendes, M. Jakob, A. Garcia-silva, and C. Bizer. DBpedia spotlight: shedding light on the web of documents. *Proceedings of the 7th Conference on Semantic Systems*, pages 1–8, 2011.

[11] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*, pages 3111–3119, 2013.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[14] N. Schlaefer, J. Chu-Carroll, E. Nyberg, J. Fan, W. Zadrozny, and D. Ferrucci. Statistical source expansion for question answering. In *Proceedings of CIKM 2011*, pages 345–354, 2011.

[15] K. Venkataramani. *Optimal classifier ensembles for improved Biometric Verification.* PhD thesis, Citeseer, 2007.