Building an Optimal Question Answering System Automatically using Configuration Space Exploration (CSE) for QA4MRE 2013 Tasks

Alkesh Patel, Zi Yang, Eric Nyberg, Teruko Mitamura

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

alkeshku@anderw.cmu.edu,{ziy, ehn, teruko}@cs.cmu.edu

Abstract

Different software systems for automatic question answering have been developed in recent years. Some systems perform well on specific domains, but may not be appropriate for other domains. As the complexity and scaling of such information systems become ever greater, it is much more challenging to effectively and efficiently determine which toolkits, algorithms, knowledge bases or other resources should be integrated into a system so that one can achieve a desired or optimal level of performance on a given task. In this working notepaper, we present a generic framework that can be used for any machine-reading task and automatically find the best configuration of algorithmic components as well as values of their corresponding parameters. Although we have designed the framework for all QA4MRE-2013 tasks (i.e. Main task, Biomedical about Alzheimer's and Entrance Exam), our analysis will mostly focus on Biomedical about Alzheimer's task. We introduce the Configuration Space Exploration (*CSE)* framework, an extension to the Unstructured Information Management Architecture (UIMA) which provides a general distributed solution for building and exploring possible configurations for any intelligent information system. For the Biomedial about Alzheimer's task, CSE was used to generate more than 1000 different configurations from existing components; we selected the 3 best runs for submission. We achieved an average c@1 of 0.27; our highest score was 0.60 for reading-test-1, and our lowest was 0.0 for reading-test-3. We further enhanced the system by introducing point-wise mutual information (PMI) scoring for answer ranking, which produced an average c@1 of 0.4025, with a highest score of 0.77 for reading test-1 and a lowest score of 0.2 for reading test-2.

Keywords: biomedical question answering, configuration space exploration, unstructured information management architecture

1. Introduction

When selecting algorithms or tools for text analysis, it can be difficult to determine which tools will be suitable for a particular task. Performance varies depending on the application and domain in which the tools are applied. Software frameworks, which support the integration of text analysis algorithms into end-to-end solutions, make it possible to build complex, high-performance systems for information extraction, information retrieval, and question answering; IBM's Watson is a prominent example of such a system. As the complexity of information systems become ever greater, it is much more challenging to effectively and efficiently determine which toolkits, algorithms, knowledge bases (KBs) or other resources should be integrated into the system in order to achieve an acceptable or optimal level of performance on a given task.

This paper presents a generic framework for QA systems, which is suitable for all tasks included in QA4MRE. It utilizes the Configuration Space Exploration (CSE) framework, an extension to the UIMA framework[1] , which provides a general distributed solution for building and exploring configuration spaces

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

for information systems. Complex QA systems are usually built as a solution to a specific task in a particular domain, and mostly lack a clear separation of framework, component configuration, and component logic. This in turn makes it difficult to adapt and tune existing component for new tasks without editing the original component source code. Such changes are expensive, and often preclude an exhaustive and/or efficient exploration of the large set of possible configurations. To fully leverage existing components, it must be possible to automatically explore the space of possible system configurations to determine the optimal combination of tools and parameter settings for a new task. We refer to this problem as *configuration space exploration* [16].

As a pilot task, we have implemented a UIMA (Unstructured Information Management Architecture) pipeline using various components required for the Alzheimer's QA task. The details of pipeline architecture are discussed in Section 3. Then, we talk about our experimental design for run submission in Section 4. In Section 5, we provide a detailed analysis and the results we obtained for each run. We conclude and describe directions for future work in Section 6.

2. Previous Work

The field of automatic question answering has seen many advances in the past decade. Previous research investigated solutions to the problem of comparative evaluation, and addressed two different aspects: *workflow management* and *benchmark evaluation*. Most of the systems available in the scientific community for evaluation focus on one or the other of these two aspects, and are not directly suitable for evaluating text information systems. Moreover, existing evaluation approaches don't provide a formal performance evaluation of each individual component in a system, which is crucial in designing the best-performing overall system. The success of IBM Watson in the Jeopardy! Challenge[2] provided great hope that complex question answering systems could achieve human-level performance when appropriate effort is invested in system tuning and optimization. The Apache UIMA framework used in Watson is highly suited to the creation of solutions which integrate multiple components for information retrieval, natural language processing, machine learning etc. However, UIMA by itself does not include a facility for optimization via automatic, comparative evaluation of multiple system configurations.

The QA research community has conducted many conferences and competitions to create standard benchmarks and metrics to evaluate participating QA systems, and organizers have tried to consolidate and analyze the most important features of high-performing systems. For example, for the TREC Genomics task, Hersh et al. employed multivariate regression analysis to discover factors associated with variation in question answering performance [11, 12]. Reproducible experiments may also be used to try different combinations of modules (e.g. across participating organizations) to find the best-performing combination. More recently, the NTCIR Advanced Cross-Lingual Information Access (ACLIA) task employed a standardized question answering data flow for two successive subtasks: IR4QA and CCLQA. The results of the IR4QA task were collected from each system and shared for processing by all participants in the CCLQA task. Mitamura et al. showed that this approach was able to find system configurations that combined different IR4QA and CCLQA systems to attain a better result than the originally reported systems [8, 10], which provides further motivation for a principled approach to configuration space exploration. However, the ACLIA evaluation used task-specific XML schemas for data exchange between subsystems, and did not provide a general framework for describing and evaluating configuration spaces that included variation of individual components and their parameter settings.

In the work described in this paper, we have UIMA for workflow management, coupled with a framework for benchmark evaluation through configuration space exploration (CSE) developed at Carnegie Mellon University [16].

[2] http://www.jeopardy.com/

3. System Architecture

We have built a UIMA-based annotation pipeline for the QA4MRE tasks. UIMA annotators are the analysis components that can be plugged into the UIMA framework to analyze unstructured information. The design of the architecture is shown in Figure 1.

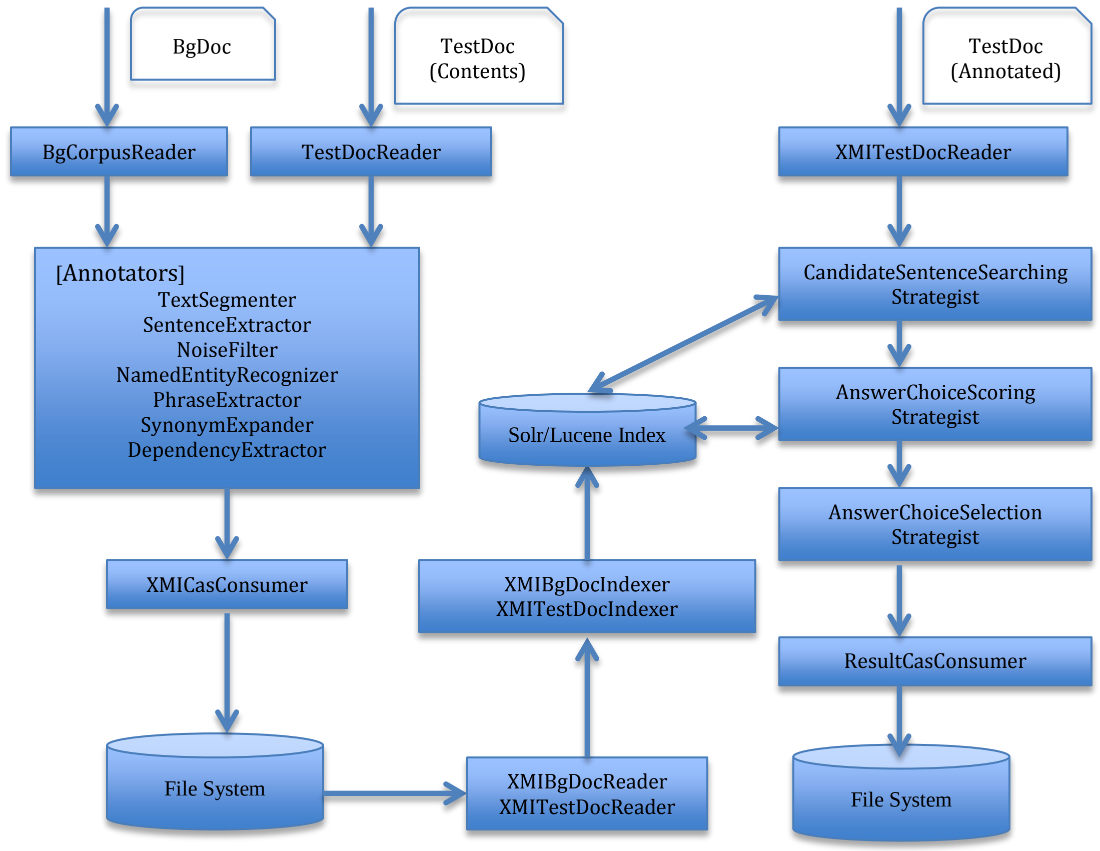

Figure 1. UIMA-based System Architecture for QA4MRE

The detail of each component is given in the following subsections.

3.1 Data Readers

Readers are basically the modules responsible for reading and parsing raw or annotated data stored in the file system. There are 4 types of data readers in our system. In QA4MRE 2013, we were given a huge background document collection for the Main task and the Alzheimer's task. *BgCorpusReader* reads the background corpus data in plain text format. In general, there can be multiple implementations for *BgCorpusReader* depending on the source and format. *TestDocReader* reads and parses the contents of the test document that contains document text and corresponding question-answer set. After passing through UIMA-based annotators, each document is serialized in the form of an XMI (XML Metadata Interchange) file. *XMIBgDocReader* and *XMITestDocReader* read XMI documents and de-serialize them into appropriate data structures for further processing (i.e. the UIMA CAS object; see below).

3.2 Annotators

In the UIMA framework, an *Analysis Engine* is a program that analyzes documents and infers information from them. Analysis Engines are constructed from building blocks called *Annotators*. An annotator is a component that contains analysis logic[3]. The annotated data is propagated from one annotator to next annotator in the form of a *CAS* (Common Analysis Structure). Depending on the domain and application there can be a long chain of annotators. Here, we have described the set of annotators we used in Alzheimer's task.

Text Segmenter: This component is important when the raw data is inherently noisy, e.g. because it was converted from PDF using OCR. In this case, it will be useful to segment the text and remove the unnecessary segments such as "References", "Figure" and "Table" headers that are unlikely to contain relevant answer text.

Sentence Extractor: This module extracts sentences from the text. We used the StanfordCoreNLP toolkit[4] for extracting sentences.

Noise Filter: There are many low-quality sentences in the data, which do not contribute to finding the correct answer. We use heuristic methods to calculate a score for each sentence and apply a threshold to discard noisy from further consideration. For example, important sentences cannot be less than '*C*' characters long, cannot contain less than 'W' words, cannot contain too many digits or too many author names (likely to be a reference) etc. Removing noisy sentences improves the candidate sentence generation process during subsequent analysis.

Named Entity Recognizer: This module extracts the named entities such as person, organization, location, protein, gene, drug etc. In many questions, specific named entities are the direct answer. We have used the StanfordCoreNLP toolkit and ABNER[5] (A Biomedical Named Entity Recognizer) for named-entity recognition in our pipeline.

Noun-Phrase Extractor: The named entity recognizer module often fails to capture important noun phrases which can be potential answer candidates. We also use a phrase extractor that utilizes Part-of-Speech tagging provided by StanfordCoreNLP toolkit to extract text matching appropriate patterns (e.g. NN*-NN* and JJ*-NN*) as key phrases.

Synonym Expander: This module uses external resources to further annotate the extracted named entities and noun phrases. We have used the DISCO[6] (**DIS**tributionally related words using **CO**-occurrences) semantic similarity tool to look up synonyms. This tool has been developed based on a statistical analysis of a very large text collection (e.g. including Wikipedia and PubMed).

Dependency Extractor: In factoid QA, the grammatical structure of the question text and the answer-bearing passage are more likely to match (e.g. by matching dependency parse relations). Hence dependency extraction can be a useful step in ranking candidate sentences. We used the StanfordCoreNLP toolkit for extracting dependency tuples (e.g., <Relation, Governor, Dependent>).

The annotation process described above is used to process both the background corpus and the test document containing questions and answer choices.

3.3 Document Indexer

After annotation of documents, it is useful to index the annotated documents to provide efficient document search during run-time question answering. Many open-source tools are available for indexing and retrieval (e.g. Solr/Lucene, Indri etc.). We used Solr[7], as it comes with a wide variety of

[3] http://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html
[4] http://nlp.stanford.edu/software/corenlp.shtml
[5] http://pages.cs.wisc.edu/~bsettles/abner/
[6] http://www.linguatools.de/disco/disco_en.html
[7] http://lucene.apache.org/solr/

configuration options for its components and provides effective indexing a search over annotated documents.

3.4 Candidate Sentence Search Strategy

As mentioned earlier, test documents are annotated, and the corresponding UIMA CAS is serialized in the form of an XMI file. The actual process of finding the best answer for the given question starts by reading the annotated XMI document. Candidate sentence extraction involves multiple steps. The system first considers the annotated question and forms an appropriate search query. Search is performed on the Solr index and the most relevant sentences for a given question are listed in descending order of their relevance score. From this list, an appropriate strategy can be used to pick the top K candidate sentences that are most likely to contain a correct answer. We have experimented with the following approaches:

- i) Search is performed on question keywords and at max K candidate sentences are selected dynamically by considering sentences having a relevance score within P% of the maximum relevance score.
- ii) Search is performed on synonyms of question keywords and at max K candidate sentences are selected dynamically by considering sentences having a relevance score within P% of the maximum relevance score.
- iii) Search is performed with question keywords and/or their synonyms as usual. Then the dependency matching score between the question and the candidate sentence is calculated to re-rank the K candidate sentences.

3.5 Answer Choice Scoring Strategy

Input to this module is the K candidate sentences and the output is a score for each answer choice with respect to each candidate sentence. Various strategies can be used to assign a score to an answer choice. We have experimented with the following approaches:

- i) Number of noun-phrases/named-entities matched in candidate sentence and answer choice;
- ii) Number of synonyms of noun-phrases/named-entities matched in candidate sentence and answer choice;
- iii) Point-wise Mutual Information (PMI) between noun-phrases/named-entities in candidate sentence and answer choice with respect to the background corpus.

3.6 Answer Choice Selection Strategy

Input to this module is the score of each answer choice with respect to each candidate sentence. Multiple strategies can also be used here; we have experimented with the following approaches:

- i) Aggregate the scores of each answer choice across the candidate sentences and pick the highest scoring answer choice;
- ii) Take voting of winner answer choice in each candidate sentence and use majority voting scheme to pick the answer choice.

The output of this module is the final answer for the question.

3.7 CAS Consumers

CAS consumers persist the output of an analysis process in different file formats. We are using two types of CAS consumer: *XMICasConsumer* and *ResultCasConsumer.* XMICasConsumer simply writes out the annotated CAS in XMI format, where the ResultCasConsumer writes final results for each reading-test document in format required for evaluation. In our case, the ResultCasConsumer generates its XML output according to the QA4MRE guidelines for a run submission.

4. Experimental Design

We submitted 3 official runs for Biomedical about Alzheimer's Task. In order to choose these three best runs, we conducted a large-scale experiment using our Configuration Space Exploration (CSE) framework. As explained in Section 3, for finding the correct answer choice, an annotated document passes through three main stages: Candidate Sentence Searching, Answer Choice Scoring and Answer Selection. Each of these stages has different possible strategies, and each strategy involves a set of configuration parameters values, e.g. Max K candidate sentences, %P threshold for dynamic K selection, how many synonyms to consider, etc. Given this possible variation, the total number of configurations turned out to be 1020 possible systems/runs. The CSE framework provides an easy way to configure and execute a comparative evaluation using appropriate YAML[8] descriptors and a gold-standard dataset. The YAML descriptor that we used is shown in Figure 2.

```
configuration:
  name: qa4mre-13-test-alzheimer
  author: oaqa-cmu
persistence-provider:
  inherit: persistence.local-persistence-provider
  collection-reader:
    inherit: collectionreaders.CollectionReaderDescriptor
    dataset: QA4MRE-13-test-alzheimer
    INPUT_DIR: data/13-test-alzheimer/
  pipeline:
    # question and document annotations: data -> XMIs
    - inherit: ecd.phase
      name: TextSegmenter
      options: |
        - inherit: annotators.TextSegmenter
    - inherit: ecd.phase
      name: StanfordSentenceAnnotator
      options: |
        - inherit: annotators.StanfordSentenceAnnotator
    - inherit: ecd.phase
      name: NoiseFilter
      options: |
        - inherit: annotators.NoiseFilter
    - inherit: ecd.phase
      name: StanfordNLPAnnotator
      options: |
        - pipeline:
            - inherit: annotators.StanfordNLPAnnotator
            - inherit: annotators.StanfordQuestionNLPAnnotator
    - inherit: ecd.phase
      name: PhraseAnnotator
      options: |
        - pipeline:
            - inherit: annotators.PhraseAnnotator
            - inherit: annotators.QuestionPhraseAnnotator
    - inherit: ecd.phase
      name: NEAnnotator
      options: |
        - pipeline:
            - inherit: annotators.NEAnnotator
            - inherit: annotators.QuestionNEAnnotator
    - inherit: ecd.phase
      name: SynonymAnnotator
      options: |
        - pipeline:
            - inherit: annotators.SynonymAnnotator
            - inherit: annotators.QASynonymAnnotator

    - inherit: ecd.phase
      name: WikiSynonymAnnotator
      options: |
        - pipeline:
            - inherit: annotators.WikiSynonymAnnotator
            - inherit: annotators.WikiQASynonymAnnotator

    - inherit: consumers.CasConsumerDescriptor
      OutputDirectory: results/13-test-main/final-xmis

    # indexing
    - inherit: ecd.phase
      name: SolrIndexer
      options: |
        - inherit: annotators.SolrIndexer

    # answer ranking and merging: XMIs -> results
```

[8] http://www.yaml.org

```
- inherit: ecd.phase
  name: QuestionCandSentSimilarityMatcher
  options: |
    - inherit: annotators.QuestionCandSentSimilarityMatcher
    - inherit: annotators.QuestionCandSentDependencyMatcher
    - pipeline:
        - inherit: annotators.QuestionCandSentSimilarityMatcher
        - inherit: annotators.QuestionCandSentDependencyMatcher
    - pipeline:
        - inherit: annotators.QuestionCandSentSimilarityMatcher
        - inherit: annotators.QuestionCandSentSynonymMatcher
    - pipeline:
        - inherit: annotators.QuestionCandSentSimilarityMatcher
        - inherit: annotators.QuestionCandSentSynonymMatcher
        - inherit: annotators.QuestionCandSentDependencyMatcher

- inherit: ecd.phase
  name: AnswerChoiceCandAnsSimilarityScorer
  options: |
    - inherit: annotators.AnswerChoiceCandAnsSimilarityScorer
    - inherit: annotators.AnswerChoiceCandAnsPMIScorer
    - pipeline:
        - inherit: annotators.AnswerChoiceCandAnsSimilarityScorer
        - inherit: annotators.AnswerChoiceCandAnsPMIScorer
    - pipeline:
        - inherit: annotators.AnswerChoiceCandAnsSimilarityScorer
        - inherit: annotators.AnswerChoiceCandAnsSynonymScorer
    - pipeline:
        - inherit: annotators.AnswerChoiceCandAnsSimilarityScorer
        - inherit: annotators.AnswerChoiceCandAnsSynonymScorer
        - inherit: annotators.AnswerChoiceCandAnsPMIScorer

- inherit: ecd.phase
  name: AnswerChoiceCandAnsSimilarityScorer
  options: |
    - inherit: annotators.AnswerSelectionByKCandVoting
    - inherit: annotators.AnswerSelectionByKCandAggregation

# output format
- inherit: consumers.ResultCasConsumer
  OUTPUT_DIR: results/13-test-alzheimer/final-outputs
```

Figure 2. YAML descriptor used in CSE framework for Biomedical Alzheimer task

We used the CLEF 2012 test set and CLEF 2013 sample set (whose Gold Standards were already given) as reference data to estimate the best configuration for the CLEF 2013 test data. We selected the top 3 runs for the submission. Details of these three runs are given in the following subsections.

Run1: In this run, we used a very simple strategy. We process the annotated question and form a Lucene query using underlying noun phrases and named entities. E.g. for the question "What protein activity can be inhibited by LiCl?", the following Lucene query is produced:

> text: (what protein activity can be inhibited by Licl) AND
> nounphrases: "protein activity"^2 AND
> ners: LiCl^2

There can be many different schemes to form a Lucene query with more sophisticated boosting of various search fields; a well-formulated query can boost the ranking of the most relevant sentences in the search result. We applied a dynamic threshold on the relevance score, and selected all relevant sentences that came within 60% of the maximum relevance score in a given set of search results as the K candidate sentences (potential answers).

Then, we extracted noun phrases and named entities from each candidate sentence and matched them with the answer choices. We calculated the matching score for each answer choice across all candidate sentences. For selection of the final answer, we aggregated the matching scores obtained for each candidate sentence and the highest scoring answer choice was picked.

Run2: In this run, we utilize synonyms of noun phrases and named entities in addition to their surface forms to construct the Lucene query. Again K candidate sentences are selected by applying the same dynamic threshold strategy used in Run1.

We used the same scheme for matching each answer choice with every candidate sentence's noun phrase and named entities. However, in Run 2 we also added a semantic similarity score between synonyms of the answer choice and candidate sentence terms. For selection of the final answer, we used a majority-voting scheme where the winning answer choice is determined for each candidate sentence, and the final answer is chosen by summing votes across the candidate sentences.

Run3: This run is almost exactly similar to run1. The only difference is in the final answer selection strategy. We used the majority-voting scheme across all candidate sentences.

As we mentioned earlier, our main objective was to automatically find the best combination from the available components, although these available components may not be intrinsically the best implementations of the desired functionality. After release of the Gold Standards for the QA4MRE-2013 Alzheimer's task, we conducted an error analysis and tried to incorporate more sophisticated strategies to improve system performance. We added a PMI (Point-wise Mutual Information) module for ranking answer choices, and noted a drastic improvement. We added 3 more runs (see below) that show how the addition of a better module in combination with existing components can result in significant performance gain using configuration space exploration.

Run4: In this run, we processed the annotated question and formed a Lucene query using underlying noun phrases and named entities. We then searched for K relevant candidate sentences with the same strategy described earlier. We extracted noun phrases and named entities from each candidate sentence. We calculate the Point-wise Mutual Information (PMI) of noun-phrase/named-entity and a given answer choice using following formula:

$$PMI(entity, answerchoice) = log\,\frac{hits(entity\ AND\ answerchoice)}{hits(entity)\cdot hits(answerchoice)}$$

The cumulative PMI score for each answer choice is calculated with respect to every candidate sentence. The idea behind using the PMI strategy is to capture the intuition that terms in the correct answer choice will be co-occurring significantly with the noun-phrases/named-entities of the candidate sentences. Scores for each answer choice are aggregated across all candidate sentences and the highest scoring answer choice is selected as the final answer.

Run5: This run uses a combination of the matching score between noun phrases/named entities of K candidate sentences and answer choices, as well as the PMI score calculated according to Run4. Answer selection was done using the aggregated score.

Run6: We wanted to check the influence of dependency matching between question and the relevant sentences retrieved; answer choice scoring was done using simple noun-phrase/name-entities matching, and the final answer choice was selected via the highest aggregate score.

5. Results & Analysis

The results we got for our submitted runs are shown in Figure 3 as pie charts. We did not try to guess an answer for any question, so the 'Unanswered, Right' and 'Unanswered, Wrong' categories are not applicable in our case. There were a total of 40 questions for 4 reading sets; each set consists of 10 questions.

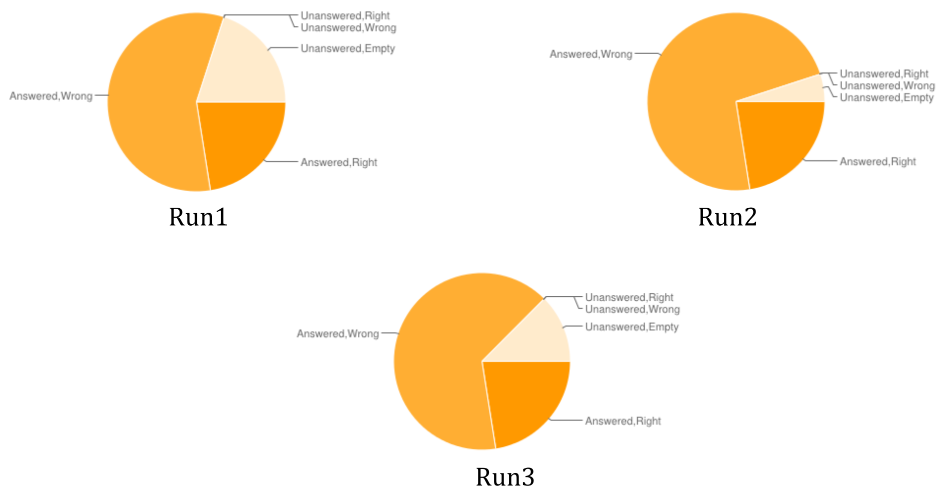

Figure 3: Pie charts of submitted runs showing Answered Right, Answered Wrong and Unanswered

The evaluation measure used in the QA4MRE tasks was 'c@1'. The formula used for calculating c@1 is given below:

$$c@1 = \frac{nr + nu * \left(\frac{nr}{n}\right)}{n}$$

where:
nr: is the number of correctly answered questions
nu: is the number of unanswered questions
n: is the total number of questions

For Run1, we answered 32 questions out of which 9 were correct. This gives c@1 value 0.27. Although the value is higher than the baseline value (i.e. 0.20 if one randomly guesses from 5 answer choices), it is not significantly higher. We also found that for reading set 3, none of the answers were correct. However, for reading set-1, Run1 could answer 5 out of 10 questions.

During our error analysis, we found that some of the answer choices in reading set-3 seemed to be designed to confuse an automatic system. E.g., for the question *"What group of receptors can regulate the expression of the Seladin-1 gene?"*, the following candidate sentences were generated:

1) On the other hand, as shown in Figure 3C, administration of TO significantly increased Seladin-1 gene **expression** in the forebrain of TRKI mice compared with that of Sham group about 1.2-fold.

2) Estrogen receptor (ER) is another nuclear receptor closely related to Seladin-1 gene **expression**.

3) However, the molecular mechanism by which **LXR**s regulate Seladin-1 gene **expression** has been yet to be elucidated.

4) Interestingly, an agonist of **liver X receptor (LXR)**, TO901317 (TO) administration in vivo increased Seladin-1 gene and protein **expression** in the mouse forebrain only in a hypothyroid state and in the presence of mutant TR-β, suggesting that **LXR**-α would compensate for TR-β function to maintain Seladin-1 gene expression in hypothyroidism and resistance to TH.

5) A) TO failed to induce the Seladin-1 gene **expression** in **LXR**-knockdown HTB185 cells.

6) The up-regulation of Seladin-1 gene **expression** by TO requires **LXR**s in HTB185 cells.

7) These data indicate that TO increases Seladin-1 gene **expression** in HTB185 cells through intrinsic **LXR**s.

8) Seladin-1 gene **expression** is down-regulated in the vulnerable region in the brain of AD patients [21].

9) In the current study, we demonstrate that TH increased Seladin-1 gene and protein **expression** in the mouse forebrain.

10) While Seladin-1 gene expression is induced in a thyrotoxic state, under hypothyroid treatment, the gene and protein **expression** levels were not significantly reduced compared to those in a euthyroid state (Figure 1).

As *"expression"* was one of the multiple choices and appeared more than *"liver X receptor"* (which is the correct choice), 'expression' got a higher aggregated matching score compared to "liver X receptor". In such a case, a better answer ranking module can play an important role. Adding synonyms into the search query for finding K candidates and using them for further matching did not actually improve the c@1 score. In Run3, the majority-voting scheme performed better than Run2 but could not surpass the c@1 in Run1 with the aggregate scoring scheme.

As discussed earlier, it was surprising that none of these three runs gave any right answers for reading set-3. To overcome the obvious underlying problem, we tried to use background corpus information. E.g. we searched in background corpus to verify whether 'liver X receptor' or 'expression' is more likely to appear in the context of the question being asked. We found the following PMI scores for each answer choice:

expression	3.58259073012903
AD	3.588834358208376
TH	5.057297430222549
Liver X receptors	**10.693749999999998**
β-amyloid	0.0

Correct Choice:	Liver X receptors
Best Choice:	Liver X receptors

PMI score proved to be significantly useful in ranking answer choices across all reading sets. We got the highest c@1 i.e. 0.77 for reading set-1. The overall c@1 scores for all submitted and un-submitted runs are given in Table 2. Results of individual reading sets is given in Table 3.

Table 2: Overall c@1 measure

Runs	c@1 Score
Run1	0.27
Run2	0.24
Run3	0.25
Run4	0.40
Run5	0.32
Run6	0.27

Table 3: c@1 score for individual reading tests

Runs	r_id 1	r_id 2	r_id 3	r_id 4	Median	Mean	S. D.
Run1	0.60	0.36	0.0	0.13	0.25	0.27	0.26
Run2	0.55	0.30	0.0	0.11	0.21	0.24	0.24
Run3	0.60	0.33	0.0	0.12	0.22	0.26	0.26
Run4	**0.77**	0.2	**0.4**	**0.22**	**0.31**	**0.40**	0.26
Run5	0.66	**0.30**	0.20	0.11	0.25	0.32	0.24
Run6	0.44	0.20	0.20	0.22	0.21	0.27	**0.12**

6. Conclusion & Future Work

In this paper, we investigated the usefulness of *configuration space exploration* (*CSE*) in discovering effective combinations of components for QA4MRE 2013 - Biomedical about Alzheimer's task. The CSE framework was used to conduct more than 1000 experiments involving different configurations of components and their corresponding parameters. The framework automatically and efficiently evaluated different system configurations, and identified a configuration of the given components that achieved competitive results with respect to any prior published result for similar data sets. We also observed that PMI scoring for ranking answer choices could provide a substantial enhancement in final answer selection.

As a next step, we are interested in combining various tools used in all tasks (Main, Alzheimer's and Entrance Exam) that turned out to be the best in QA4MRE 2013. It would be exciting to see which combinations outperform existing combinations that were reported in the original evaluation.

Acknowledgments

We thank all members of the OAQA project team, and especially Collin McCormack, for their valuable contributions to this work. We also thank the students in the Question Answering Lab course who experimented with and provided a comparative analysis of the different tools we used in this work.

References

[1] *Attardi,* G., Atzori, L. and Simi, M. Index Expansion for Machine Reading and Question Answering. *Working notes of CLEF 2012: Evaluation Labs and Workshop for QA4MRE'12 Tasks.*

[2] *Bhattacharya,* S. and Toldo, L. Question Answering for Alzheimer Disease Using Information Retrieval. *Working notes of CLEF 2012: Evaluation Labs and Workshop for QA4MRE'12 Tasks.*

[3] *Martinez,* D., *MacKinlay, A., Molla-Aliod,* D., *Cavedon,* L. and *Verspoor,* K. Simple Similarity-based Question Answering Strategies for Biomedical Text. *Working notes of CLEF 2012: Evaluation Labs and Workshop for QA4MRE'12 Tasks.*

[4] Tsai, B.H., Liu, Y.Z. and Hou, W.J. Biomedical Text Mining about Alzheimer's Diseases for Machine Reading Evaluation. *Working notes of CLEF 2012: Evaluation Labs and Workshop for QA4MRE'12 Tasks.*

[5] Verbeke, M. and Davis, J. A Text Mining Approach as Baseline for QA4MRE'12. *Working notes of CLEF 2012: Evaluation Labs and Workshop for QA4MRE'12 Tasks.*

[6] Dupont, G. M., de Chalendar, G., Khelif, K., Voitsekhovitch, D., Canet, G. and Brunessaux, S. Evaluation with the virtuoso platform: an open source platform for information extraction and retrieval evaluation. In *Proceedings of DESIRE'11*, pages 13–18, 2011.

[7] Eckart de Castilho, R. and Gurevych, I. A lightweight framework for reproducible parameter sweeping in information retrieval. In *Proceedings of DESIRE'11*, pages 7–10, 2011.

[8] Mitamura, T., Shima, H., Sakai, T., Kando, N., Mori, T., Takeda, K., Lin, C.Y., Song, R., Lin, C.J. and Lee, C.-W. Overview of the ntcir-8 aclia tasks: Advanced cross-lingual information access. In *Proceedings of NTCIR-8*, pages 15–24, 2010.

[9] Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W., Hill, D. P., Kania, R., Schaeffer, M., St Pierre, S., Twigger, S., White, O. and Rhee, S.Y.Y. *Big data: The future of biocuration. Nature, 455(7209):47–50, Sept. 2008.*

[10] Mitamura, T., Nyberg, E., Shima, H., Kato, T., Mori, T., Lin, C.-Y., Song, R., Lin, C.-J., Sakai, T., Ji, D. and Kando, N. Overview of the ntcir-7 aclia tasks: Advanced cross-lingual information access. In

Proceedings of NTCIR-7, pages 11–25, 2008.

[11] Rekapalli, H.K., Cohen, A. M. and Hersh, W.R. A comparative analysis of retrieval features used in the trec 2006 genomics track passage retrieval task. In *Proceedings of AMIA'07*, pages 620–624, 2007.

[12] Hersh, W. R., Cohen, A.M., Roberts, P. M. and Rekapalli, H.K. Trec 2006 genomics track overview. In *Proceedings of TREC'06*, 2006.

[13] Meij, E., Jansen, M. and de Rijke, M. Expandingqueriesusing multiple resources. *In Proceedings of TREC'06, 2006.*

[14] Si, L., Lu, J. and Callan, J. Combining multiple resources, evidences and criteria for genomic information retrieval. *In Proceedings of TREC'06, 2006.*

[15] Turney, P. D. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *In Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001), pp. 491-502.*

[16] Yang, Z., et al. (2013). Building Optimal Information Systems Automatically: Configuration Space Exploration for Biomedical Information Systems. In *Proceedings of the ACM Conference on Knowledge and Information Management (CIKM) 2013* (to appear).