Building 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

Software tools that facilitate question answering are increasing in past few years. Some tools perform better on specific domain 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) framework that provides a general distributed solution for building and exploring configuration spaces for any intelligent information system. CSE generated more than 1000 different configurations from existing components. We selected 3 best runs for submission in Biomedical about Alzheimer's task. We achieved average c@1 as 0.27 with highest 0.60 in reading-test-1 and lowest 0.0 in reading-test-3. We observed that, with more sophisticated tools and algorithms, one could obtain optimal performance of the QA system. We demonstrated enhancement by introducing point-wise mutual information (PMI) scoring for answer ranking which gave average c@1 as 0.4025 with highest 0.77 for reading test-1 and lowest 0.2 for reading test-2.

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

1. Introduction

There are a big number of tools coming up every year for text analysis. It is sometime hard to determine which tool will be suitable for particular task. Their performance varies depending on the application and domain in which they are applied. Software frameworks, which help integration of text analysis algorithms, make it possible to build complex, high performance information systems for information extraction, information retrieval, and question answering; IBM's Watson is one of the most prominent examples such framework. As the complexity of information systems become ever larger, it is much more challenging to effectively and efficiently determine which toolkits, algorithms, knowledge bases (KBs) or other resources should be integrated into an information system to achieve an optimal level of performance on a given task.

This paper presents a generic framework for QA system, which is suitable for all tasks included in QA4MRE. It utilizes 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 change is expensive and often precludes efficient exploration of a 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*.

As a pilot task, we have implemented a UIMA (Unstructured Information Management Architecture) based pipeline of various components required in question answering system for Alzheimer's 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 detailed analysis and results we obtained for each submitted as well as un-submitted run. We conclude and provide future work direction in section 6.

2. Previous Work

The area of automatic question answering has gone through lot of advancements in past decade. Previous research investigates solutions to the problem of comparative evaluation, and provides information from two different aspects: *workflow management* and *benchmark evaluation*. Most of the systems available in the scientific community for achieving these two lack either one or the other. Even if they provide both workflow management and benchmark evaluation, their workflow is not suitable for text information systems. Moreover, they don't provide performance evaluation of each individual component, which is crucial in designing best performing overall system. The success of IBM Watson in 'Jeopardy'[2] provided great hope that complex question answering system can beat the best human participant. In 'Watson', apache's UIMA framework was used. UIMA is very convenient platform to combine multiple components for information retrieval, natural language processing, machine learning etc. However, it doesn't include ability to find the best blend of components automatically.

Question and answering system 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. employ multivariate regression analysis and discover factors associated with variation in question answering performance [11, 12]. Organizers usually want to reproduce experiments to try different combinations of modules across participants 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. presented 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 further motivates us for configuration space exploration. However, in ACLIA evaluation they used task-specific XML schemas, and did not provide a general framework for describing and evaluating configuration spaces that included variant on components and their parameter settings.

In this paper, we have used the best state of the art available as workflow management tool i.e. UIMA and a framework on top of UIMA for facilitating benchmark evaluation through configuration space exploration i.e. CSE (Configuration Space Exploration) developed at Carnegie Mellon University.

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

3. System Architecture

We have built UIMA-based annotation pipeline for 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.

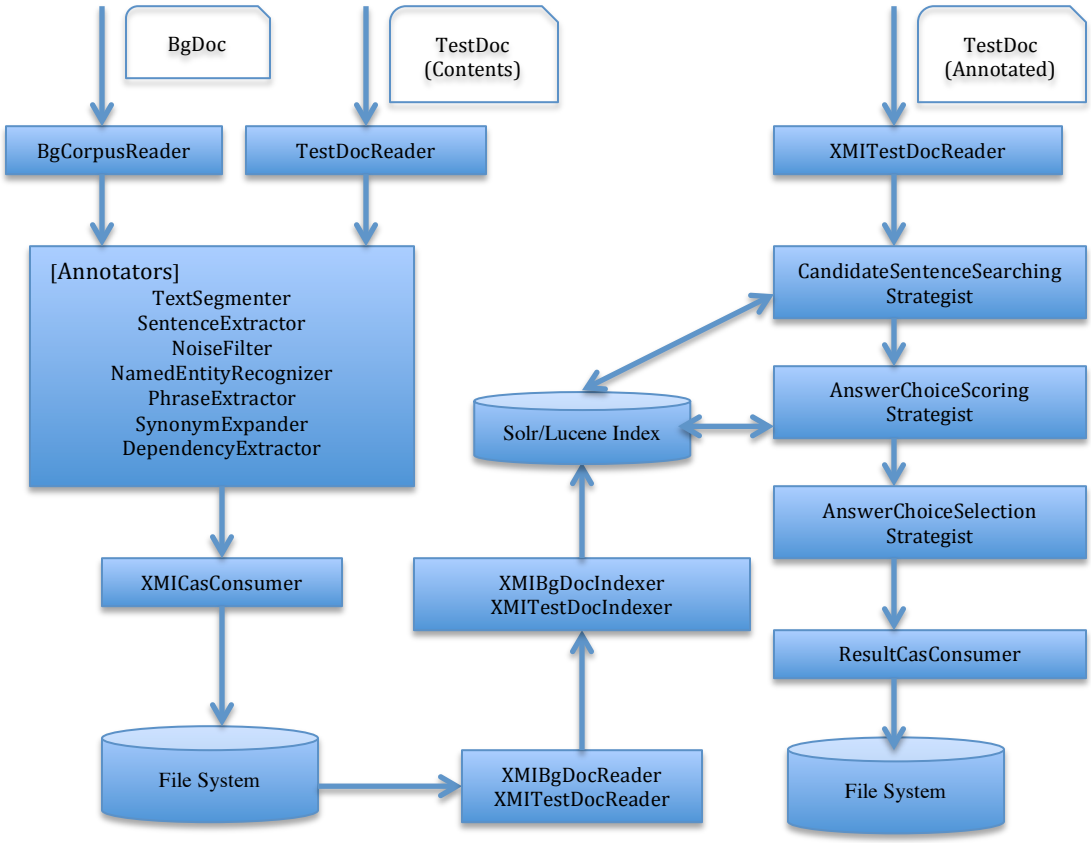

Figure 1. UIMA-based System Architecture for QA4MRE

The detail of each component is given in following subsections.

3.1 Data Readers

Readers are basically the modules responsible for reading and parsing raw or annotated data stored on file system. There are 4 types of data readers in our system. In QA4MRE 2013, we were given huge background document collection for Main task and Alzheimer's task. *BgCorpusReader* reads the background corpus data given 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 annotations each document is serialized in the form of XMI file. *XMIBgDocReader* and *XMITestDocReader* read XMI document and de-serialize the annotated document in appropriate data structures for further processing.

3.2 Annotators

In 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 *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 raw data is inherently noisy because it may be OCR converted from some PDF document. 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 not going to help in finding the answer.

Sentence Extractor: It extracts the sentence from the text. Detecting sentence boundaries is not easy task and one can have customized module for his/her need. We have used StanfordCoreNLP toolkit[4] for extracting sentences.

Noise Filter: There are many low quality sentences, which do not contribute in finding the correct answer. We calculate the scores of each sentence and apply threshold to discard it from further consideration. We have used some heuristics for calculating sentence score. For example, important sentence cannot be less than '*C*' characters, cannot be less then 'W' words, cannot contain too many digits or too many author names (likely to be a reference) etc. Removing noisy sentences will improve the candidate sentence generation process in subsequence analysis.

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

Noun-Phrase Extractor: Named entity recognizer module many times fail to capture some important noun phrases, which can be potential answer candidates. So, we also use phrase extractor that utilizes Part-of-Speech tagging provided by StanfordCoreNLP toolkit. We, then extract appropriate patterns (e.g. NN*-NN* and JJ*-NN*) to extract the key phrases.

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

Dependency Extractor: In many situations, especially in factoid question, grammatical structure of question text and most-likely candidate sentence's text matches in their corresponding dependency structure. So, dependency extraction can be a useful step in ranking candidate sentence. We have used StanfordCoreNLP toolkit for dependency extraction. It extracts dependency in the form of (Relation, Governor, Dependent) tuple.

Above annotation process is common for both background corpus document and test document containing questions and answer choices.

3.3 Document Indexer

After annotation of document, it is good idea to index the annotated document using some efficient technique because we may need them while searching for the answer in future analysis. Many tools

[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

are available (e.g. Solr/Lucene, Indri etc.) as open source. We used Solr[7] as it comes with wide variety of configuration options for its components and perform very well on for indexing and searching task.

3.4 Candidate Sentence Searching Strategist

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

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

3.5 Answer Choice Scoring Strategist

Input to this module is K candidate sentences and output is score for each answer choice with respect to each candidate sentence. Various strategies can be used to assign score to answer choice. We have experimented 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 background corpus

3.6 Answer Choice Selection Strategist

Input to this module is score of each answer choice with respect to each candidate sentence. Now, here also, multiple strategies can be used. We have experimented 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

Output of this module is final answer for the question.

3.7 Cas Consumers

Cas consumers do a job of generating the output of analysis process in required format as file. We are using two types of Cas consumers: *XMICasConsumer* and *ResultCasConsumer.* XMICasConsumer simply writes out the annotated CAS in XMI format. On the other hand, ResultCasConsumer writes final results for each reading-test document in required format. In our case, ResultCasConsumer generates the XML output according to QA4MRE guidelines for run submission.

[7] http://lucene.apache.org/solr/

4. Experimental Design

We submitted 3 official runs for Biomedical about Alzheimer's Task. However, before choosing these three best runs from our system, we conducted huge experiment using our Configuration Space Exploration (CSE) framework. As explained in section 3, for finding correct answer choice, annotated document passes through three main stages: Candidate Sentence Searching, Answer Choice Scoring and Answer Selection. Each of these stages has different strategies and each strategy involves set of configuration parameters e.g. Max K candidate sentence, %P threshold for dynamic K selection, how many synonyms to consider etc. Therefore, when we took all possible combinations and configurations, the total number of experiments turned out to be 1020. CSE framework provides easy way to configure using appropriate YAML[8] (YAML Ain't Markup Language) descriptors that does the desired task and finds the best configuration that gives optimal performance on reference data. A typical YAML descriptor 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
```

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

```
    options: |
      - inherit: annotators.SolrIndexer

  # answer ranking and merging: XMIs -> results
  - 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 considered CLEF 2012 test set and CLEF 2013 sample set (whose Gold Standards were already given) as reference data to estimate best configuration for CLEF 2013 test data. We selected top 3 runs for the submission. Details of these three runs are given in following subsections.

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

> 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 to various search fields. A good formation of search query can bring most relevant sentences up in the ranking. We then applied dynamic threshold on relevance score and selected all relevant sentences that came within 60% of maximum relevance score in search results. We call these relevant sentences as K candidate sentences for potential answer.

Then, we extracted noun phrases and named entities from each candidate sentence and matched with the answer choices. We calculate the matching score for each answer choice across all candidate sentences. For selection of final answer, we aggregate the matching score obtained in each candidate sentence and highest scoring answer choice is picked.

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

We use same scheme for matching each answer choice with every candidate sentence's noun phrase and named entities. However, this time we also add semantic similarity score between synonyms of answer choice and candidate sentence. For selection of final answer, we used majority-voting scheme where winner answer choice is determined for each candidate sentence and final answer is chosen by taking votes across candidate sentences.

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

As we mentioned earlier, our main objective was to find the best combination from available components automatically. These available components do not necessarily be best in themeselves. After release of Gold Standards for QA4MRE-2013 Alzheimer's task, we did error analysis and tried to incorporate more sophisticated strategy to improve question answering system performance. We added PMI (Point-wise Mutual Information) module while ranking answer choice and found drastic improvement. We added following 3 more runs that show how addition of better module in combination with existing components can result in significant performance gain using configuration space exploration.

Run4: In this run, we process annotated question and form a Lucene query using underlying noun phrases and named entities. We then search for relevant K candidate sentences with same strategy described earlier.
Now, we utilize already indexed background corpus. We extracted noun phrases and named entities from each candidate sentence. Each such noun-phrase/named-entity and given answer choice is searched together in background corpus to find the Point-wise Mutual Information (PMI) using following formula:

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

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

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

Run6: This also came up in list of best performing runs using CSE. We wanted to check the influence of dependency matching between question and relevant sentences found. Answer choice scoring was done using simple noun-phrases/name-entities matching and final answer choice was selected using highest aggregated score.

5. Results & Analysis

The results we got for our submitted run are shown in Figure 3 as pie charts. We did not try to guess the answer for the question rather, we did not answer it, and hence 'Unanswered, Right' and

'Unanswered, Wrong' is not applicable in our case. There were total 40 questions for 4 reading sets each set consists of 10 questions.

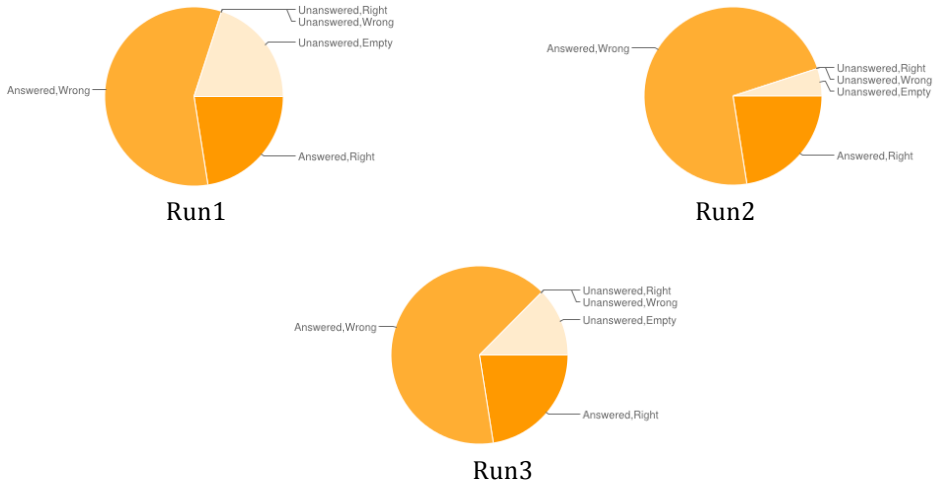

Figure 3: Pie charts of submitted runs showing Answered Right, Answered Wrong and Unanswered

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 baseline value i.e. 0.20 (if one randomly guesses from 5 answer choices), it is not significantly higher. We looked into deeper and found that for reading set 3, none of the answers was 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 were designed to confuse the automatic system to pick wrong answer. E.g. For question *"What group of receptors can regulate the expression of the Seladin-1 gene?"*, 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 correct choice), 'expression' got more aggregated matching score compared to "liver X receptor". In such case, better answer ranking module can play important role. Adding synonyms into search query for finding K candidates and using it for further matching did not actually improve the c@1 score. In Run3, majority-voting scheme performed better than Run2 but could not surpass the c@1 in Run1 with aggregate scoring scheme.

As discussed earlier, it was surprising that none of these three runs gave any right answer for reading set-3. To overcome the obvious underlying problem, we tried background corpus information. E.g. we search in background corpus to verify whether 'liver X receptor' or 'expression' is more likely in context of question being asked. We found 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 highest c@1 i.e. 0.77 in 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 *configuration space exploration* (*CSE*) in discovering right combination of components for QA4MRE 2013 - Biomedical about Alzheimer's task. The CSE framework

was used to conduct more than 1000 experiments involving different configuration of components and 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 the similar data set. We also observed that PMI scoring for ranking answer choice could provide a substantial enhancement in deciding final answer.

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 best in QA4MRE 2013. It would be exciting to see which combinations have great potential to succeed in real life situation.

Acknowledgments

We thank all fellow group members, especially Collin McCormack, for their valuable inputs. We also thank our classmates in Question-Answering Lab course for experimenting with different tools and providing their analysis about performance of related tools.

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.*