Robust Answer Ranking for Biomedical Question Answering

Zhengzhong (Hector) Liu[1], Da Teng[2], Guoqing Zheng[1], Xiawen Chu[2], Ryan Carlson[1] (Team 3)
Language Technologies Institute[1], Carnegie Mellon University, Pittsburgh, PA 15213
Very Large Information System[2], Carnegie Mellon University, Pittsburgh, PA 15213
{liu, dateng, gzheng, xchu, rcarlson}@cs.cmu.edu

Abstract

This paper details efforts to improve a baseline question-answering system. In particular, we are working off an existing system for the QA4MRE-2013 tasks. We focused our efforts in two broad categories: (1) using robust information retrieval techniques to inform our results; (2) integrating and employing state-of-the-art natural language processing tools into the system; and (3) narrowing the space of answers by identifying question types. We developed and evaluated our system using the c@1 metric on the QA4MRE-2012 Alzheimer task test set, achieving a score of 0.375 once all the pieces came together, which obtain an improvement of more than 15% over the original baseline[1].

Introduction

This paper builds on a previous effort by Patel et al. (2013) using the Configuration Space Exploration (CSE) too. While our system does not use the CSE, we do share the underlying baseline system. It relies heavily on the Unstructured Information Management Architecture (UIMA) project, allowing us to create components, integrate those components into a pipeline, and efficiently run experiments in pipelines with different component configurations. In the rest of these working notes, we describe our annotators, the IR techniques and NLP tools we use, the question type annotation scheme we developed, and our results.

Basic system design and workflow

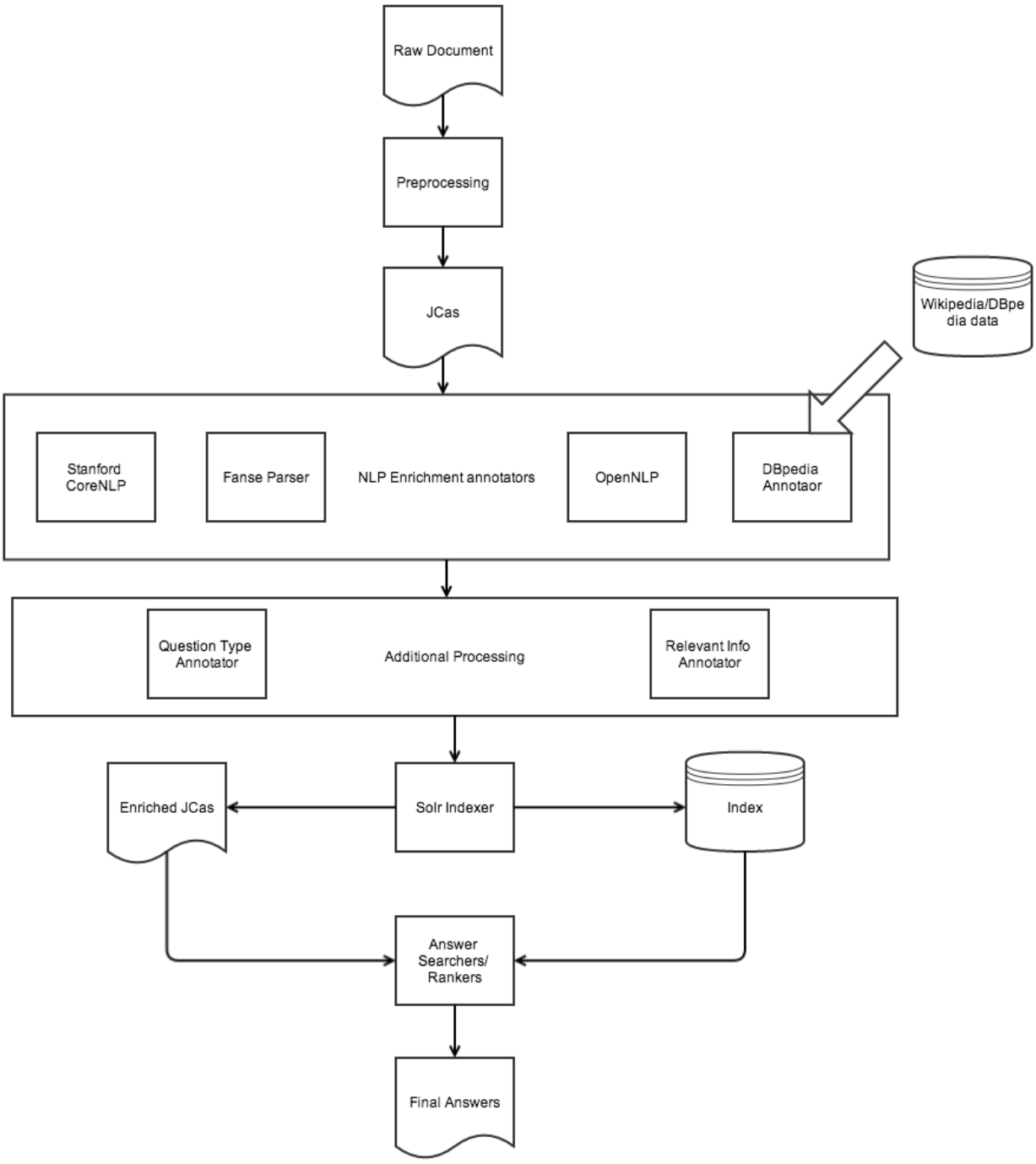
Here we introduce the basic system design roughly, detail implementations of each sub components are listed in each section respectively.

Our system is a relatively straightforward pipeline as shown in the following diagram. The system is designed to be flexible so that any NLP enrichment annotator can be plugged in easily and all these annotations can be run in parallel as the diagram suggested.

One design that worth mention is the additional processing layer, which is done after NLP enrichment. These steps are used to further clean up the noise and provide more guidances for answer ranking. These annotation require to get more information from the previous annotations, thus an additional layer is plugged.

Notably, the system bring in external information such as DBpedia using some of the annotators.

[1] Source code can be found on : https://github.com/hunterhector/hw5-team03



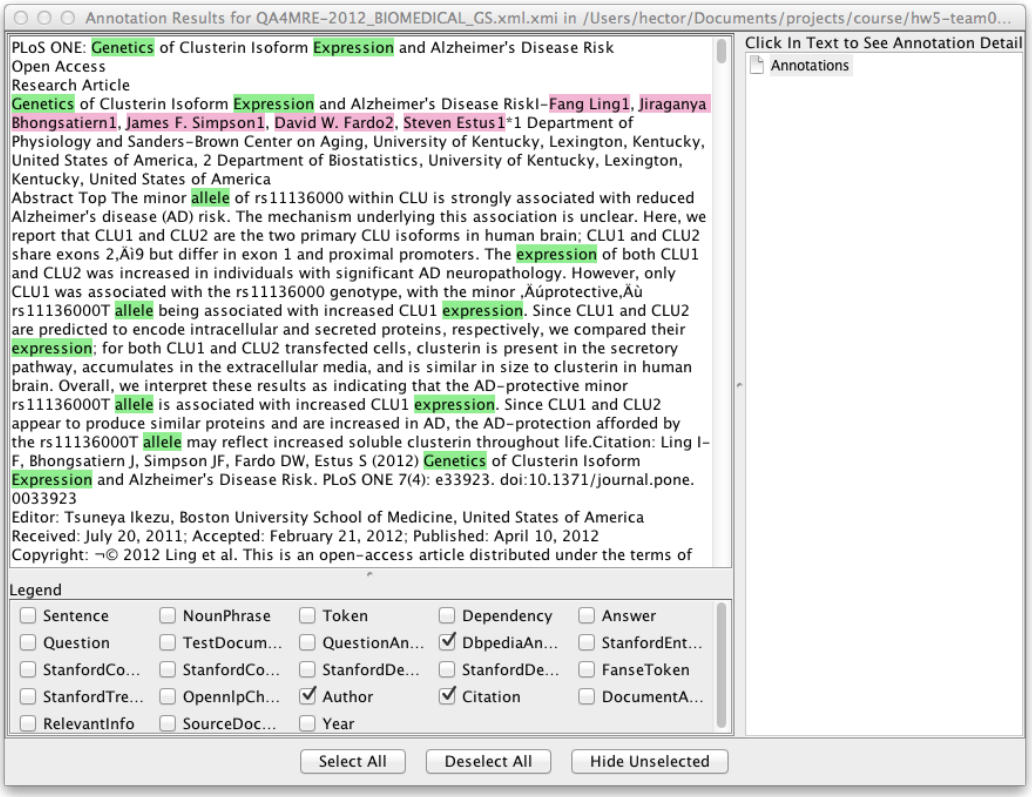
Preprocessing and data clean up (Hector)

Although in the baseline system a lot of data clean up methods are provided. The pipeline does not provide standard UIMA-style annotations, where “begin” and “end” for annotations are rarely used. This does not affect the system performance, but will make the debug process harder. A

few additional data clean up is done on top of the original system. Besides the small tweaks, two main changes are presented here:

1. **Correct annotation offsets**

With correct offsets, we better utilize the power of UIMA. As can be seen in the following exhibit, one can easily identify the annotations and have a empirical understanding about them.



2. **Append question, answer text at the end of the system**

We also append the Question(red), Answer(cyan) annotations nicely at the end of the text, as shown in the following exhibit. This give us ability to fast identify what kind of annotations on questions and answers can be used to improve the system.

Q: Which technique was used to determine the cellular locations of the CLU1 and CLU2 gene products?

A: intracellular and secreted
A: ER
A: intracellular localization
A: Golgi apparatus
A: immunofluorescence experiments

Q: What compartments inside the cell contain clusterin proteins?

A: ER and the Golgi apparatus
A: epitope tag
A: anibody
A: secretory pathway
A: secreted

NLP Annotation Enrichments (Hector, Da Teng, Xiawen)

The baseline system is built on the observation certain language units can convey more information and are less noisy. Examples includes biomedical entity mentions, noun phrases. This strategy tend to favor a high precision and will produce a conservative answerer, which will be less vulnerable to noisy tokens. This strategy has proven to be quite successful in the QA4MRE task 2012. However, the term mismatching problem becomes more severe. For example, in the Question there is the phrase “many people”. This expression does not occurs explicitly in the original text, a replacement “a lot of people” is used instead. One possible solution is to match these two phrases using their semantic meaning, which is itself another unsolved puzzle in natural language processing.In this system we choose a simple approach: enriching the indexing.

In the example above, a clear way to solve the problem is to include “token” in addition to phrases. While tokens could be noisy, they at least make sure we will get some candidates. In addition to “token”, we adopt modern natural language processing techniques to generate a lot of other information.

1. **DBpedia annotation**[2]: This annotation annotate text span in the document that can be identified as DBpedia resources. For example, the term Alzheimer will be link to “Alzheimer's Disease”. This annotation will provide additional knowledge about the terms, thus bringing in new knowledge
    a. Uri: The DBpedia URI
    b. similarityScore: The confidence that this text is really linked to the resource
    c. resourceType: The type of the resource, for example “Person”.
    d. abstract: A summary string for this resource, it is actually the first paragraph from Wikipedia.
2. **OpenNlp Chunking**: The original system find noun phrases by looking at the Part of Speech tag. We instead use the pre-trained model by OpenNLP to do phrase chunking. In addition, we can include other types of phrase if needed in the future.
3. **Stanford Parsing**: Stanford CoreNLP provides a set of natural language analysis tools which can take raw English language text input and give the base forms of words, their parts of speech, normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, and indicate which noun phrases refer to the same entities. In our project, we are using Stanford CoreNLP to annotate tokens, sentences, time, name entities, part of speech and entity coreference in the corpus.
4. **Fanse Semantic Parsing**: Fanse parser is based on the tree conversion of the Penn Treebank(Marcus, et al., 1993) which could support non-projective trees and can provide

[2] An unresolved issue is still on github, which is an offset mismatching problem by DBpedia. We suspect that is due to a problem in http transmission but we can’t find a solution to it. As does not affect our usage to the annotation, this issue is not resolved.

informative dependency labels and shallow semantic annotation for prepositions sense disambiguation, possessives interpretation, propbank SRL, and noun compound relations. In the initial pipeline, we will use the fanse parser to identify the semantic roles for the query term.

5. **Morphological annotators**: Although both Stanford and Fanse annotators provide lemma annotations, which greatly reduce the term mismatching problem, it is not solved so easily. For example, the following two terms are already lemmatized : neurodegenerative; neurodegeneration; However, our system is still not able to process them (especially the search engine). In order to resolve this, we add a morphologic neutralizer that give each token a more base form (called "morpha"). For some terms that the morphologic annotator cannot handle (out of vocabulary), we add a "shorten" field that only capture a prefix of certain length (12 in our final system).

As time does not permit, not all the information in the above mentioned annotations are used. For example, the semantic role of Fanse parser is not included for ranking the answers.

Robust Information Retrieval Techniques (Guoqing)

A particular focus of our design has been devoted to building a robust system that can generalize over domains. While tuning Biomedical-domain specific tricks is possible, we favor methods that works well in general. To achieve this, we design and improve an Information Retrieval based backbone that give the system the ability to rank answer using robust statistics. On the same time, this IR backbone provide a convenient infrastructure that facilitate many downstream works.

In this particular task where we aim to answer questions based on the information provided by a biomedical article, we propose to model the task as a sentence retrieval problem with robust statistics from the index built from the provided biomedical article to ensure the correct answer is as likely to be among the retrieved sentences as possible. That is, we try to include the correct answer in the retrieved sentences so that downstreams answer ranking annotations can recover it by reranking all the candidate sentences.

General framework

All sentence in the biomedical article are indexed and retrieved with standard vector space model from Solr 3.6.1;

Indexing Strategy

For indexing, we index the following fields:

- **dbp_annotation**:
  This field stores the annotations from DBPedia, such as *Alzheimer's* and *disease*
- **dependencies**:
  This field stores the output from the conventional stanford dependency parser

- **docid**:
  This field stores the id for each article the sentence belongs to, such as *QA4MRE11791_withQtype.xml_1*
- **id**:
  This field is the unique identifier for the sentences. One example is *QA4MRE11791_withQtype.xml_1_2*
- **nounphrases**:
  This field stores all the noun phrases found in the sentence, such as *due, fact, longer, Alzheimer, disease, elderly, Alzheimer, disease, need,* and *attention*
- **root_node**:
  This field stores the root node of the sentence in the parse result in order to identify the type of the sentence. One example is *increasing*
- **shortened**:
  This field stores all the lemmas in the sentence where the lemmas are enhanced by a list of manually annotated lemmatization results which the stemmer fails to do. It is designed to handle the case where the stemmer of Solr cannot handle the specific biomedical term correctly which hence leads to no retrieval result from the index.
- **text**:
  This field stores the original text of the sentence in the index.

Ranking Strategy

For retrieval, the strategies we adopt are as follows:

- We try to use as much available information, that is, we extend the way the baseline system uses to construct query to search more fields listed above, such as **dbp_annotation**, **root_node** and **shortened** to increase the likelihood to get more relevant sentences from the index;
- Another important and novel strategy we use in searching is that, when we get no results from the index with the queries constructed from above, we then resort to a backoff model to soften the queries and hope to at least retrieve something from the index instead of getting nothing and then not answering the question. For example, we have encountered the problem that Solr is not able to match “neurodegeneraion” in one query to the “neurodegenerative” in the documents. Because there are no exact occurences of “neurodegeneration” happened in the original article, and hence it is impossible to get the correct sentence the answer would be referring to if we cannot match “neurodegenrative” with “neurodegeneration”. In order to alleviate this problem, we devise a scheme that is we manually created a list of lemmas for this type of difficult terms and store the field as **shortened** in the index. We then do the same processing for the query terms and thus we are now able to match those different and difficult terms.
- A problem occur when we simply add all tokens and other annotations into search, the noise level is increased and the performance generally get worse. In order to sovle this problem, a second backoff strategy is adopted : if we do not get any results from the index and the problem is not the one described above, then we try to loosen the parts in the query where proximity is required. That is, if we cannot get results for a query with

phrases in it, then we resort to search the index with just terms from the index and try to get results to facilitate downstream answer rankers.

Question Types (Ryan)

Our goal is to classify question categories so that we might narrow the search space of plausible answers. Each question belongs to exactly one category. The first task is to generate an annotation manual. We then annotate a set of examples by hand according to the manual. The final task is to automatically tag the annotations.

Annotation Manual

We're given some clues from the guidelines, which suggest that a question can be one of the following:

- **Factoid**: Where or When or By-Whom
- **Causal**: What was the cause/result of Event X?
- **Method**: How did X do Y? Or: In what way did X come about?
- **Purpose**: Why was X brought about? Or: What was the reason for doing X?
- **Which Is True**: Here one must select the correct alternative from a number of statements, e.g. What can a 14-year-old girl do?

After looking at the data, we found that these categories aren't maximally informative for our purposes. We're looking for a coding scheme that helps us *find the answer* to a question. Thus, when we think about "question type" in some sense we're really thinking about "answer type". Is this a quantity question? Is this a pick-from-a-list question? Is this location question? A person question?

Knowing the answers to these questions helps us greatly eliminate noisy answers. For example, suppose the question was "How many people are on Earth" and possible answers are:

1. Yes
2. 7 Billion
3. 9,000,000,000
4. Barack Obama
5. All of the Above

We can immediately reject 1, 4, and 5 because they are not quantities. So, with that motivation, we present our coding scheme below. Note that some of the categories from the guidelines are included because they are useful, but elsewhere the ideas are expanded on.

- **Quantity**: Questions that are looking for a numeric answer. Often begins "How Many"
- **Cause**: Question that asks for some reason, asks the answerer to explain something, or asks for the cause or the effect. These tend to be "Why" questions.

- **Factoid**: Question that asks for a fact about the world. These are often "What" or "Which" questions.
- **Relation**: Question that is a FACTOID that's also looking for a relation between two items. For example, "What color is most similar to blue?" is a fact about the world, but it relates two items together. This may not be a particularly important feature, but it does seem at least qualitatively different.
- **Time**: Question that asks for a time period. These are "When" questions. They need not necessarily have numeric answers, but they often are.
- **Binary**: Question that can (in theory) be answered with a "yes" or a "no" question. In practice, there are always 5 different answer choices, so it's not really binary. These are "Is X true?" questions.

There are two additional codes that can be applied to each question:

- **Multiple**: If the question is looking for a singular or multiple answers. That is, the question "What are the two best colors in the rainbow" would be labeled as MULTIPLE.
- **Negate**: If the question is looking for NOT something, label this as NEGATE. For example, "Which of the following is not a true statement" would get this label.

Annotation Distribution

We hand-annotated the 334 questions in the data set. Given time constraints, only one person annotated. In a research setting, we would need at least 2 people annotating and coming to agreement with an acceptable kappa statistic.

The results of the annotation are as follows. We found: 219 factoids, 63 causes, 17 quantities, 17 binary question, 13 times, 5 relations. In addition, we found that 7 questions were negated, and 10 questions were requesting multiple answers. Note that because of the relative rarity of the auxiliary features, we did not attempt to further automatically code them. Instead we focused on the six primary features.

Automated Coding

To examine what a fully automated solution might look like, we extracted text features from the questions and answered and tested several machine learning algorithms to evaluate how well we can predict the 6 categories described above.

We used the LightSide workbench, which allows for easy text analysis using Weka as a backend. We extracted unigrams, bigrams, and part of speech bigrams from the question and answer text. Using logistic regression with L2 normalization, we were able to predict labels with an accuracy of 0.85 and a kappa of 0.69, the latter of which accounts for skewed label distributions.

To improve performance, we added one additional feature: the first word of the question. This is indented to identify "Why" or "What" or "Which" questions. While it didn't change accuracy, it did give kappa a slight boost, pushing it to the 0.70 barrier that we generally consider desirable.

In our actual system, we rely on a set of keywords that we hand-picked to accurately describe each question type. We decided our efforts could be spent elsewhere, rather than making LightSide or another machine learning platform play nice with UIMA. Our actual system achieves a surprising 97% accuracy, though is likely a bit less general than the logistic regression. For our purposes, however, the hand-written rules should be more than sufficient.

Answer Weighting

One advantage of labeling the question type lies in that we can filter some unreasonable candidate answers. Similar to the question above on “how many people on earth”, we could delete some answers which are not numbers. This can be regarded as given the weight of the corresponding answer with weight zero. Besides, we believe answers for certain type of questions are supposed to be biased. For instance, consider the binary question on “Is it possible for scientists to produce incentives memory?”. Given the candidate answers such as “sometimes yes”,”most frequently” and “never”, we tend to believe the third answer should be right since the boundary between “sometimes yes” and “most frequently” can be hard to tell when the information is limited. So we use the developing dataset to get the coefficient for the answers such as binary type questions and factoid types which could help us improve the performance to some extent.

Annotators

We incorporated all of the annotators from the base work, including the text segmenter, sentence extractor, noise filter, named entity recognizer, noun-phrase extractor, synonym expander, and dependency extractor. See the original paper (Patel et al., 2013) for more details about these annotators. We now describe our custom annotators.

Fanse Token Annotator: In this part we annotated the tokens in the original documents based on the fanse annotation models. For each token, we have added features like coarse Pos, lexical sense, head dependency relation, child dependency relation, head semantic relations and child semantic relations.

StanfordNLP Annotator: In the StanfordNLP annotator, we mainly annotate the basic properties for the tokens such as token id, sentence id, dependency node and dependency relations, entity mention.

DBpedia Annotator: In this annotator, we have annotated text span in the document that can be identified as DBpedia resources. This annotation will provide additional knowledge about the terms, thus bringing in new knowledge such as Uri, similarity score, resource type and abstract.

Morphology Annotator: Annotate the base form of morphology changed terms, also annotate the shortened token here.

Relevant Information Annotator: Since the raw, original training data is very messy, we decided to clean it up. This included detecting dates, citations, and authors, which eliminated some of the patently unhelpful information. This eliminated some of the original noise from the training data, which helped later components in the pipeline.

Question Type Annotator: Based on the question types and results of the machine learning, we manually created an annotator that uses keywords and phrases to identify questions. This included using the phrase "how many" to identify *quantity* types, and checking for questions starting with "is" to identify *binary* types.

Answer Selection and Ranking Annotators: We have several answer selection and ranking annotators, the detail of how to choose answers are written in respect section.

Results

As the gold standard for the final blind test set is not provided, we show the results and improvements in the 2012 testing data set.

Run	Baseline	Local Background Corpus	Softvote	Softvote+ Backoff	Hardvote+Backoff
C@1 at Doc 1	0.22	0.22	0.22	0.30	0.30
C@1 at Doc 1	0.55	0.55	0.30	0.30	0.40
C@1 at Doc 1	0.33	0.22	0.44	0.40	0.30
C@1 at Doc 1	0.20	0.40	0.44	0.40	0.50
Average C@1	0.325	0.3475	0.35	0.35	0.375

Specifically, the "Local Background Corpus" run is what we get by using our local index of the articles instead of using the background corpus server provided by the baseline and as it yields better results, we stick to this setting in the all following 4 runs on the 2012 test sets; the"Softvote" run is what the result is when we get several candidate answers for the question, we aggreate by their scores; the "Hardvote" run is what the result is when we get several candidate answers for the question, we aggreate by their times being selected for the answer in each candidate sentences. The "Backoff" refers to the above two techniques we introduced in the above "Retrieval Strategy" section.

Breakdown of individual contribution

Some of the work distribution has been marked clearly in the section title, there are several aspects that cannot be reflected by this writing, which are included in the following table.

Contributor	**Contributions**
Zhengzhong (Hector) Liu	Preprocessing; Type system design; Workflow design; DBpedia annotator; Morphology annotator; Output consumer; Experiments on training data (2012 test); Search strategy development; Whole system integration; Documentation maintainer; Report write up;
Ryan Carlson	Question type definition and design; Question type annotation (gold standard); Question type classifier; Type system design; Design question type based methods; Documentation maintainer; Relevant Info annotator; Report write up;
Guoqing Zheng	Indexing setup; Local background corpus server maintenance; Search strategy development; Search field design and implementation; Experiments on training data (2012 test); Documentation maintainer; Report write up;
Xiawen Chu	Fanse annotator; Stanford annotator; Answer weighting strategy based on question type; Experiments on final data (blind test set); Report write up;
Da Teng	Opennlp annotator; Answer weighting strategy based on question type; Final answer validation;

Conclusions & Future Work

We have discussed our general approach, building on an existing question-answering pipeline. We combined several techniques from Natural Language Processing, Information Retrieval and Machine Learning, discussed our scheme to build enhanced NLP annotators, to use robust statistics from the index to reliably retrieve possible correct answers from the index, and to annotate questions to help narrow the answer choices from the answer candidates. We showed a 15% improvement over the baseline, achieving 0.375 on the 2012 testing data set.

Besides from direct contributing to better performance, we also contribute to the community by coming up with a more reasonable question type annotation manual, and question type annotation. Our question type classifier is quite accurate on our designed question types, which give a substantial boost to answer reranking.

We have several directions for our future work. We could do more integrate our NLP and IR tools into our pipeline. We could also apply the machine-learned question-type model to our pipeline, which might help our system generalize, especially to new sources of data. Other possible directions of this word could be trying to incorporate richer external information sources to better facilitate answer selection and answer ranking for the biomedical question answering task.

Bibliography

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pp. 423-430.

Alkesh Patel, Zi Yang, Eric Nyberg, Teruko Mitamura. Building Optimal Question Answering System Automatically using Configration Space Exploration (CSE) for QA4MRE 2013 Tasks.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*(EMNLP '11). *Association for Computational Linguistics*, Stroudsburg, PA, USA, 1257-1268.

Mendes P.N., Jakob M., García-Silva A., Bizer C. DBpedia Spotlight: Shedding Light on the Web of Documents. *In the Proceedings of the 7th International Conference on Semantic Systems (I-Semantics 2011)*. Graz, Austria, September 2011.