A UIMA-Based QA System

Chris Curtis, Nigel Kilmer, David McHugh

Approach

- UIMA text processing pipeline
    - DKPro suite of NLP modules
    - Custom query and answer processing modules
- Indri for indexing and passage retrieval

UIMA

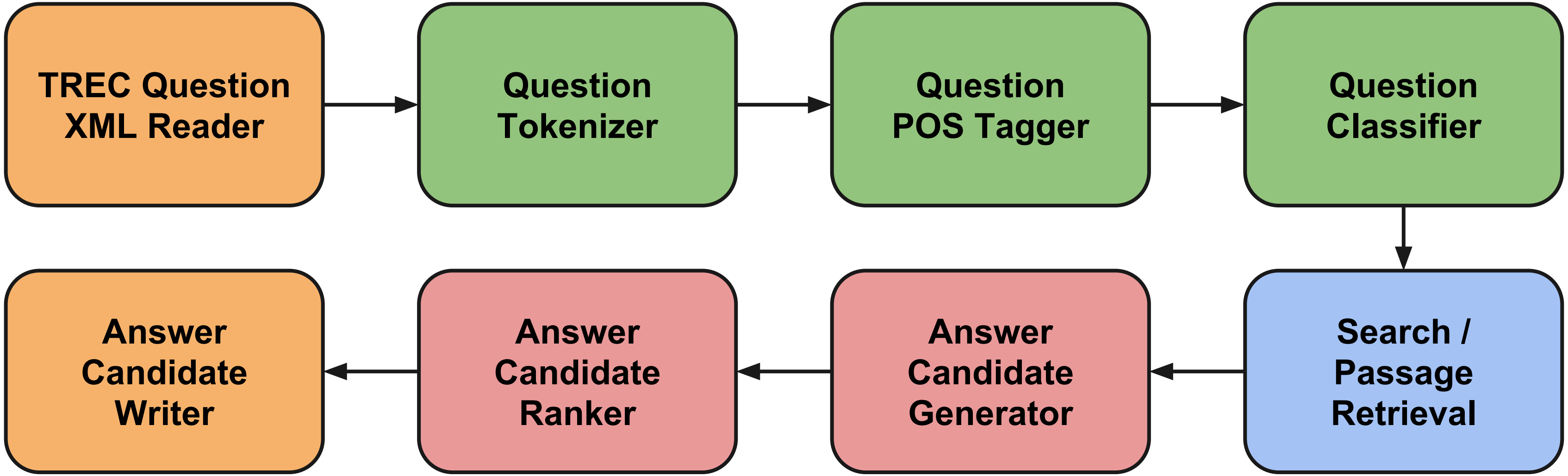
- **U**nstructured **I**nformation **M**anagement **A**rchitecture
- Provides structure for a pipeline of text processing components

- CAS - **C**ommon **A**nalysis **S**tructure
  - Stores original text and annotations (feature structures) produced by components
  - Annotations exist in the context of a type hierarchy

DKPro

- Suite of UIMA modules corresponding to open-source NLP toolsets
    - Stanford Segmenter
    - OpenNLP POS Tagger
    - (OpenNLP Chunker)

Pipeline


TREC Question XML Reader

Question Tokenizer

Question POS Tagger

Question Classifier

Search / Passage Retrieval

Answer Candidate Generator

Answer Candidate Ranker

Answer Candidate Writer

Question Classifier

- Simple classification based on *wh*-word in question:
    - “what” > *entity*
    - “who” > *person*
    - “when” > *time*
    - “why” > *reason*
    - “how” > *method*
- Annotation not used in baseline system

Search / Passage Retrieval

- Build a query using the NNs from question as keywords
  - *#combine*(NN_1, NN_2, …)
- Indri returns top 20 text snippets in windows around the matching terms
- Clearly we can improve on this approach

Answer Processing

- For each result returned by Indri, create a *CandidateAnswer* feature structure
  - Answer text
  - Score of the returned passage from Indri
  - Other features later (e.g. answer classification)

- Answers are ranked based on the Indri score

- No filtering of answers yet

Results

- Results calculated using TREC 2006 question set:
    - Strict MRR: 0.0176
    - Lenient MRR: 0.0510
- Low scores are due to placeholder versions of many components

Successes and Issues

- UIMA and DKPro allow us to easily create and integrate new modules into our pipeline
- Indexing using Indri was straightforward

- DKPro chunking modules producing warnings and errors, had to back off from using chunking of question text in our baseline
- Some UIMA feature structures are cumbersome to deal with (lists)
- No handling of question sets yet (apart from creating a feature structure type hierarchy for them)

UIMA CAS Example 1

```
[What is the name of the winning team?]
Sentence
  begin: 6
  end: 43
[What]
PR
  begin: 6
  end: 10
  PosValue: "WP"
[What]
Token
  begin: 6
  end: 10
  pos: PR
       begin: 6
       end: 10
       PosValue: "WP"
```

```
[is]
V
  begin: 11
  end: 13
  PosValue: "VBZ"
[is]
Token
  begin: 11
  end: 13
  pos: V
       sofa: _InitialView
       begin: 11
       end: 13
       PosValue: "VBZ"
```

UIMA CAS Example 2

```
Search
  queryString: "#combine( number  students )"
  searchResults: NonEmptyFSList
      head: SearchResult
      docId: "678"
      uri: "APW19980601.1143"
      score: -6.755781840076132
      rank: 4
      snippet: "...(text removed due to lack of space)..."
      tail: NonEmptyFSList
            head: SearchResult
            docId: "24"
            ...
```

References

David Ferrucci and Adam Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, September.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt Knowledge Processing repository based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology*, Tübingen, Germany, April.