UIMA Components

Input

- OpenTrivaQA parser
    - Parses files in OpenTriviaQA format
    - Each question is a document
    - For each question the following annotations are added
        - Question → the question text
        - Answer → the correct answer
        - Answer candidates → all four answer candidates; each candidate has its own annotation

Kommentiert [TK1]: This may be dropped because I don't see a use case in our system for the answer candidates…

Question Processing

- Keyword extraction
    - The question should be analyzed by other components (e.g. POS tagger) before
    - Extract all key words from question
    - Beyond the baseline: the component may add an importance score to each keyword

Tag Cloud Enrichment

- Category and hypernym detection
    - This is just a single component which is internally multi-threaded
    - For each resource (e.g. Wikipedia, WordNet) a separate thread is started
    - Each resource creates a list of categories/hypernyms
    - After the research for each resource is finished, the list is set as the corresponding feature in the answer annotation (thereby building the "large tag cloud")
    - **Parameters**
        - Annotation type: over which annotation to iterate; this will be used to run the analysis engine on the correct answer first and later on the answer candidates

Kommentiert [TK2]: As of my research, it seems to be non-trivial to parallelize UIMA analysis engines

Kommentiert [TK3]: That's why there are the two distinct types "correct answer" and "candidate answer"

Candidate Extraction

- Category ranking
    - Iterates over all categories and hypernyms (possibly in parallel) of the correct answer
    - Uses a measurement to derive a score for each category which reflect its relevance
        - For instance, page rank could be used for Wikipedia categories
        - The measurement may change to allow different system configurations
    - The scores are assigned to the categories
    - **Parameters**

        - Relevance measure
- Candidate extraction
    - Extract categories/hypernyms from the correct answer with the highest relevance score
    - For each of the extracted categories, retrieve candidates
    - Create an "answer candidate" annotation for each retrieved candidate
    - **Parameters**
        - How many categories/hypernyms to use
- Synonym resolution
    - Iterate over all "answer candidate" annotations
    - Perform synonym resolution with the correct answer
        - E.g. by using WordNet
    - Remove all synonymous candidate answers by removing the corresponding "answer candidate" from the index

Kommentiert [TK4]: This is probably not easy

Kommentiert [TK5]: Maybe add the correct answer's keywords to it?

Similarity Detection

- Similarity detection
    - Iterate over all "answer candidate" annotations
    - Perform the similarity measure between the candidate's tag cloud and the correct answer's tag cloud
    - Assign the resulting scores to the answer candidates
    - **Parameters**
        - The similarity measure

Kommentiert [TK6]: Maybe perform the similarity calculation in a separate thread for each answer candidate

CAS Consumer

- Candidate selection
    - Iterate over all "answer candidate" annotations
    - Select the candidates with the highest scores and output them
    - **Parameters**
        - Number of candidates to select

Kommentiert [TK7]: This may be used as an input for another system

UIMA Types

Question

- Standard UIMA annotation
- No special features

Answer

- Annotation for an answer (correct or candidate)
- Features
    - Keywords
    - Categories
    - Hypernyms
    - Answer type

Kommentiert [TK8]: All of them implement a common interface which allows unified access to them

Kommentiert [TK9]: This, however, would require an additional analysis engine

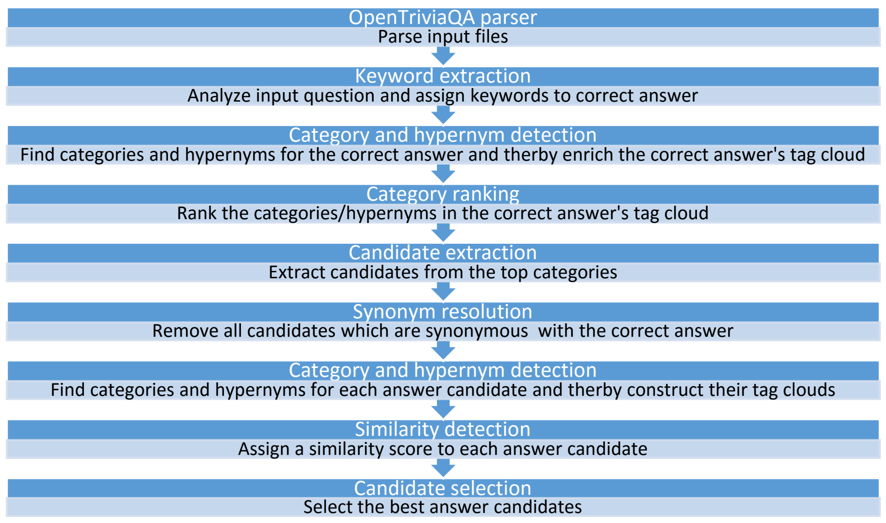
Correct Answer

- Inherits everything from the answer type
- Is used to distinguish the correct answer annotation from the candidate annotation

Answer Candidate

- Inherits everything from the answer type
- Features
    - Similarity score
- Is used for iterating over answer candidates

Final Pipeline


OpenTriviaQA parser
Parse input files

Keyword extraction
Analyze input question and assign keywords to correct answer

Category and hypernym detection
Find categories and hypernyms for the correct answer and therby enrich the correct answer's tag cloud

Category ranking
Rank the categories/hypernyms in the correct answer's tag cloud

Candidate extraction
Extract candidates from the top categories

Synonym resolution
Remove all candidates which are synonymous with the correct answer

Category and hypernym detection
Find categories and hypernyms for each answer candidate and therby construct their tag clouds

Similarity detection
Assign a similarity score to each answer candidate

Candidate selection
Select the best answer candidates