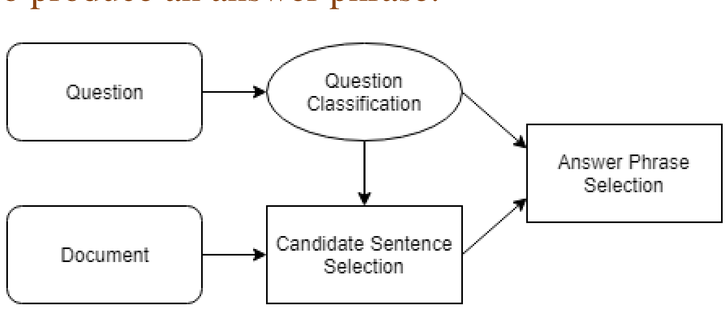
Question Answering with 2-Stage Candidate Selection

Adam Davies, Carlos Jimenez

Introduction

Our objective was to design a **single-document question answering (QA) system**.

Our system follows a 2-stage approach: first, candidate sentence selection; second, answer phrase identification. This approach was inspired by that of Wang, Liu, Xiao et al.[1] We began by performing question classification (via constituency parse); next, we used lexical analysis and a rule-based approach for candidate sentence selection; finally, we utilized syntactic argument analysis, guided by dependency/constituency parsing and semantic cues, to produce an answer phrase.



Tools Used

Natural Language Tool-Kit (NLTK): Used directly for tokenization, lemmatization, POS tagging, some NER, and chunking.

Stanford CoreNLP: Accessed via NLTK to provide constituency and dependency parses.

Gensim, Wordnet: Both assessed word similarity (Gensim via pre-trained embeddings, Wordnet via 'synsets').

SpaCy: Provided some forms of NER and dependency parsing.

Stage 1: Candidate Sentence Selection

Our first stage requires question classification into questions types. This allows us to treat each case differently, including specific sub-cases.

For example, if we know we are dealing with a 'how-much' type question, we can often identify candidate sentences by those containing MONEY patterns, or CARDINAL entities. After narrowing down the pool of candidates, we filter our candidate pool through more general sentence selection subprocesses.

The "general" case: Our system has two generalized processes to rank candidate sentences.

For example, one process involves extracting sentences' lemmatized words, while preserving POS tags, and comparing them with those words in the question processed the same way. Sentences are then ranked by similarity (via Wordnet 'synsets' and Gensim's word embeddings), and passed on to the second stage (answer identification).

Stage 2: Answer Phrase Selection

The second stage begins with dependency parsing to determine relevant syntactic arguments (e.g. subject, direct object, etc.). Next, we use Wordnet 'synsets', Gensim vector similarity measures, and NER to determine the extent to which various arguments satisfy the semantic properties expected based on the type of question. For example: 'who' questions expect ENTITY types, or words related to 'person' or 'organization', in subject or direct object positions; 'where' questions expect LOCATION types, or words related to various synonyms of 'location', as prepositional objects; and so on. (Compare Na, Kang, Lee, and Lee[4].)

Finally, we look for the constituents nested within the most relevant arguments (or, in some cases, arguments nested within relevant constituents) of the expected form for the given question-type, and return them.

Conclusions

What Worked Well:

- Using a three-pronged approach of NER, word embeddings, and Wordnet 'synsets' made sure desired information was almost always captured.
- Rule-based additions and heuristics often improved performance significantly, and were easy to implement.

What Did Not:

- Relying too heavily on parsing sometimes resulted in a lack of robustness (given answers occurring in unexpected syntactic structures).

[1] Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. Joint training of candidate extraction and answer selection for reading comprehension. https://arxiv.org/abs/1805.06145, 2018.

[2] Seung-Hoon Na, In-Su Kang, Sang-yool Lee, and Jong-Hyeok Lee. Question answering approach using a wordnet-based answer type taxonomy. 03 2003.