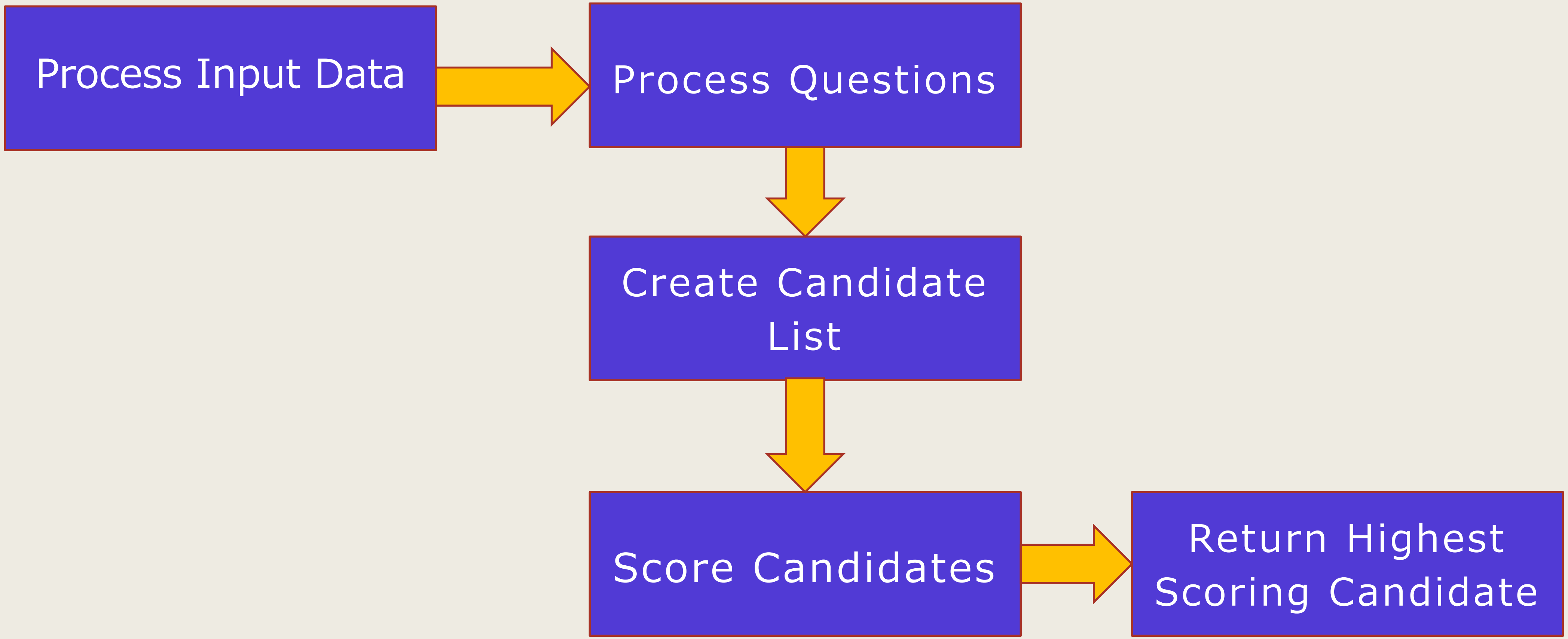
Question answering system

by

Janaan Lake

System Architecture



Process Input Data

- Resolve coreferences in story
- Create BERT embeddings for each sentence and for each question
- Store processed information in story and question dicts for later reference

Process Questions

- Create a set of keywords for each question

  Remove stopwords and lemmatize the words

- Determine question type based on first word, i.e. “Where”, “Who”, “What”, “When”, “How”, “What”, “Why”, and Other

Where Questions

Question candidates are drawn from

- NER tags: LOC, FAC, GPE
- Prepositional phrases that start with ”in”, “at” “on”, ”into”, “inside”, “outside”

Who Questions

Question candidates are drawn from

- NER tags: GPE, NORP, ORG, PERSON

When Questions

Question candidates are drawn from

- NER tags: DATE, EVENT, TIME
- Prepositional phrases that start with ”during”, “after” “before”, ”while”, “as”, “when”
- Phrases that include “from ___ to ___”

How Many Questions

Question candidates are drawn from

- NER tags: CARDINAL, PERCENT, QUANTITY

How Much Questions

Question candidates were drawn from

- NER tags: MONEY, PERCENT

How Questions (Other)

If word following “How” is an ADV or ADJ, then question candidates are drawn from:

- NER tags: CARDINAL, ORDINAL, QUANTITY

How Questions (Other)

If word following “How” is an AUX or VERB, then question candidates are drawn from:

- phrases that contain the “nsubj” found in the question

Why, What, Other Questions

Question candidates are drawn from:

- phrases that contain the “nsubj” found in the question
- if no “nsubj” is found, then the entire sentence that has the highest cosine similarity score based on BERT embeddings

Update Candidates

- Candidates where 75% of the text appears in the question are removed.
- Noun chunks, or noun phrases, that include the candidate are added and the substring candidate is removed

Score Candidates

- Each candidate is given a score.
- The score is a combination of a cosine similarity score and a keyword distance score.
- The candidate with the highest score is returned as the answer.

Cosine Similarity Score

Cosine similarity of the BERT embedding for the question and the BERT embedding for the sentence in which the candidate is found.

Keyword Distance Score

- The minimum total distance from each keyword in the set to the candidate
- The keyword that is the root of the question is given extra weight
- Possessive keywords are given extra weight
- The final score is the inverse of the total distance.

Emphasis/Originality

- Using BERT embeddings for cosine similarity score
- Coreference resolution using Neuralcoref library

Performance

	Dev Set	Test Set 1	Test Set 2
Recall	0.4362	0.5394	0.4752
Precision	0.2882	0.3579	0.3172
F-Score	0.3471	0.4303	0.3804

Performance

	Where	Who	When	How	What	Why
Recall	0.3783	0.3451	0.5905	0.3418	0.3102	0.2453
Precision	0.2992	0.3234	0.6522	0.2549	0.184	0.1586
F-Score	0.3341	0.3339	0.6198	0.292	0.231	0.1927

Results from the Dev Set data

Lessons Learned

- Starting with a smaller and more simple scope and building upon this foundation is much better and more efficient than starting with a complicated model. This approach also allows for measuring impact of each improvement.
- Computing performance for each individual question type was very helpful for knowing where and how to focus more effort.
- Learning how to use the spaCy library, which is very useful for NLP tasks.

Regrets/Complications

A lot of time was initially spent researching ML techniques, specifically predicting spans of answers in a dataset. Because the dataset wasn't similar to the SQUAD dataset and because the dataset was so small, this approach was abandoned. However, after seeing the results of the top team more persistence down this direction should have been pursued!

Regrets/Complications

Working with different libraries and getting them to play nicely with each other was complicated! I used the nueralcoref library for coreferencing, but I had to install an older version of spaCy to make this work. I also spent a lot of time working with semantic role labeling. I made a completely new model that incorporated role labeling using the AllenNLP library and the Forte framework. I had success in extracting some thematic roles, but this library was not compatible with BERT because they required different versions of the transformer library. Because of time limitations and because my first model had some better performance, I submitted that model as my final project. However, with more time I would have incorporated the thematic roles into my first model and then found a new way to calculate cosine similarity scores without using BERT embeddings.

External Resources

- NLTK toolkit

  https://nltk.org

- spaCy NLP library

  https://spacy.io

- Nueralcoref library

  https://spacy.io/universe/project/neuralcoref

- BERT Transformer Model

  https://huggingface.co/docs/transformers/model_doc/bert