CS 6320 Fall 2021 — Ziyad Amir Dhuka (zxd200000)
Project — James Amato (jca120030)

Natural Language, Processed Unnaturally

Hello Ximeng! Please note that this is a very preliminary draft. We will certainly be updating this, but wanted to give you a preview.

1 Problem Description

Given a limited knowledge set, answer several questions pertaining to it.

The data is a portion of the Stanford Question Answering Dataset (SQuAD). It consists of 30 articles and over 2500 questions. We were provided some data with which to develop our solutions. The solution will be tested with unseen questions pertaining to those 30 articles. Thus, in the test environment, preprocessed articles are able to be used.

2 Proposed Solution

Our solution really does a decent job with task 1 and 3, and has a thought-through-for-many-hours-but-still-nonfunctioning answerer for task 2. Currently, the best performing method uses a simple keyword search, with keywords broken into individual tokens.

We use established libraries and software for processing and storing NLP information from the articles. Each question is similarly processed. We hope to establish a good method for processing questions based on `Lasso`, 2000 [5]. For finding keywords,

3 Full Implementation Details

blah

3.1 Programming Tools

All work is performed in Python.

The relational database used is `solr` [2]. It is Java-based, open-source, enterprise-level, and developed by contributors to the Apache Software Foundation.

The NLP pipeline uses many tools. NLTK is used for tokenization and POS-tagging [1]. It is used in conjunction with WordNet for lemmatization and finding synonyms [8]. Named entities and parse trees are established using spaCy [4].

When reading sentence, the RAKE (Rapid Automatic Keyword Extraction) algorithm is used. It was described by Rose, et al. in 2010 [6]. The library implemention by Sharma leverages the strengths of nltk [7].

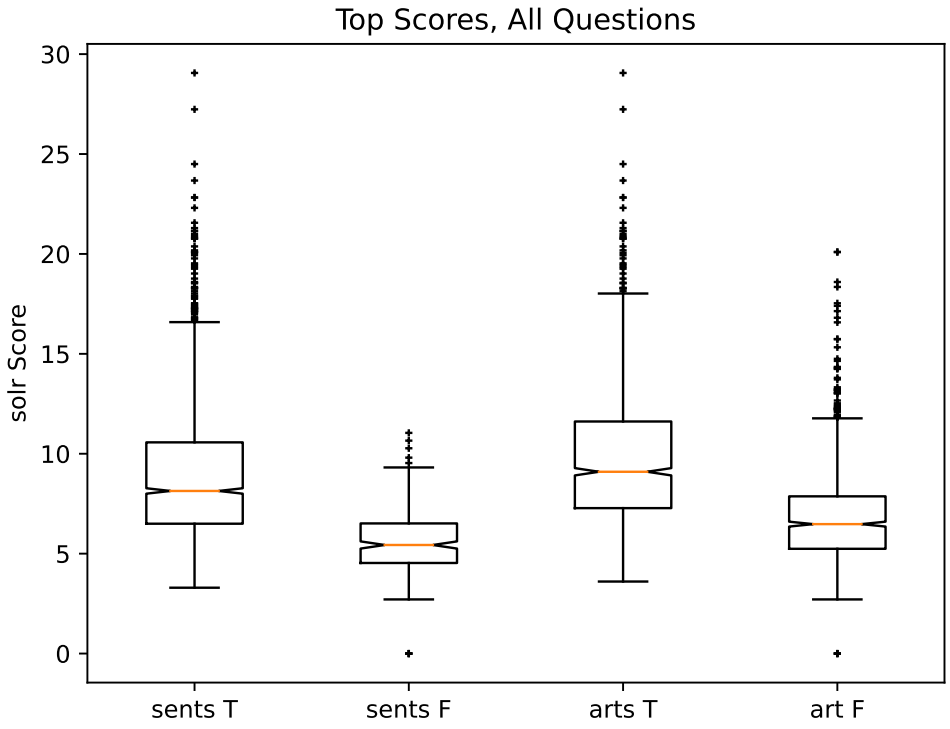
3.2 Architectural Diagram

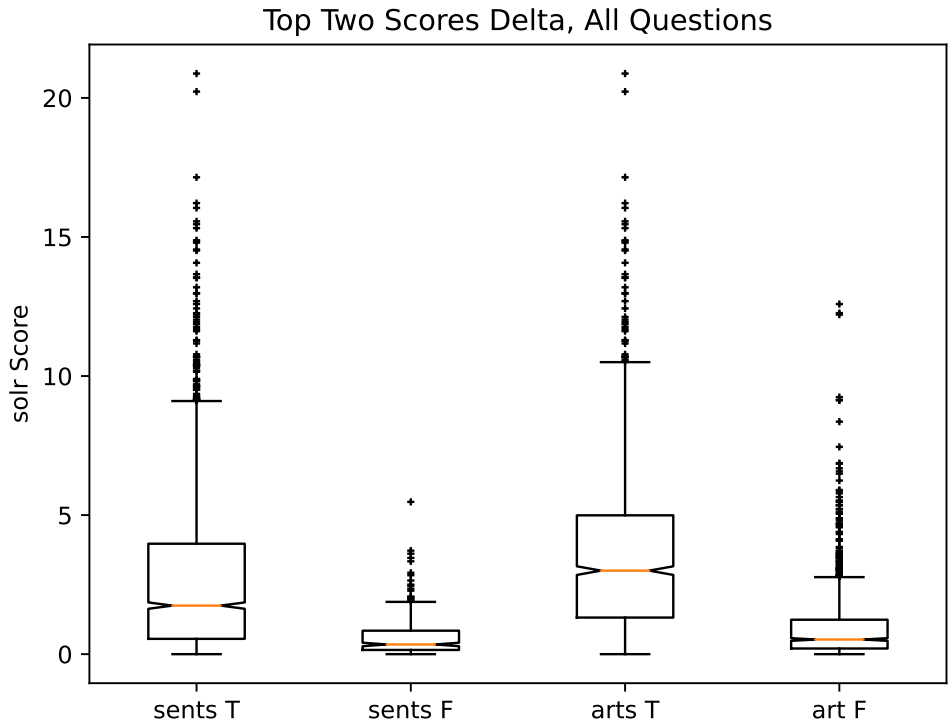
[Architectural Diagram to be developed]

3.3 Results and Error Analysis

As development proceeded we found the need to determine when correct articles are found. Our analysis simply counts instances of the correct answer in the target sentence. The first system we built was meant to deliberately get things wrong, and simply returned the first result in the whole solr database, when searching for type:'sentence'. By default, this happened to be the first sentence in article 109, "Bird migration is the regular seasonal movement, often north and south along a flyway, between breeding and wintering grounds." We counted number of times the article was correctly identified, and number of times the provided answer appeared in the sentence. Interestingly, because the terms "north" and "south" are answers to two unrelated questions, this method counted those as correct. Another question asked something similar, "What is the most common direction of migration in autumn?"; the provided answer "south" appears in the provided answer. So we see it is an imperfect method.

Nevertheless, we proceed. We tried to make data-driven decisions. As such, when we found a reasonably good, albeit very simple answerer, we began building tools for analysis. One such analysis looks at the solr score for a given result, and compares it when the answer is correct or incorrect, and when the article is determined correctly or incorrectly. We examined both the top score in those cases, and the deltas between scores. We did this analysis both on a small subset of questions (used for development) and the full set provided.





3.4 Problems: Encountered and Resolved

Initially, Jim had difficulty reading the data files into python, though Ziyad did not. The solution is always setting encoding='utf-8' when reading and writing files.

We built a BadAnswerer in hopes of getting the most improved award. This method simply returned the first sentence found in any article. In the set of questions, 89 are for this article, so those are correct by default.

Interestingly, for BadAnswerer, the results were better than expected:

BadAnswerer: Of 2505 total questions, the correct article was found 89 times and the correct sentence was found 4 times.

Next method simply used RAKE keyword search. Of 2505 questions, got 1453 correct articles and 769 correct sentences.

KeywordAnswerer: Of 2505 total questions, the correct article was found 1453 times and the correct sentence was found 769 times.

Next method - broke RAKE keywords into component pieces. For instance, from the question: "On what did Skousen analyze ink and pencil remnants?" we previously found only 2 compound keywords ("skousen analyze ink" "pencil remnants"); now we find 5 ("skousen" "analyze" "ink" "pencil" "remnants") Of 2505 total questions, the correct article was found 2206 times and the correct sentence was found 1380 times. (55%)

Next method - blacklisting just two words ("one", "type"), which seem to be particular to questions, did not improve results.

While preparing to add NE utiliation, explored how question words are used. There is a list of questions, below, with multiple question words in a single question. Need to use better system (POS tagging? syntatic analysis?)

Some difficulty when trying to find meronyms and holonyms... because it's not quite as simple as that. More hours than I'm proud to admit were spent debugging, when in fact the proper solution is that wordNet breaks meronyms and holonyms into substance and part types.

3.5 Pending Issues

We are trying to apply scipy statistics to the score results, but get ridiculous and plainly incorrect p-values and confidence intervals (overestimating precision).

3.6 Potential Improvements

Time management skills, for real, am I right?!

Further work would employ more of the techniques found in `Falcon`, 2000 [3].

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009. Accessed: 2021-10-20.

[2] The Apache Software Foundation. Apache solr. https://solr.apache.org/, 2021. Accessed: 2021-11-07.

[3] Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. Falcon: Boosting knowledge for answer engines. *In Proc. of the 9th Text Retrieval Conference (Trec-9)*, pages 479–488, 2000.

[4] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[5] Dan Moldovan, A Harabagiu, Rada Mihalcea, Richard Goodrum, and Vasile Rus. Lasso: A tool for surfing the answer net. *In Proc. of the Text Retrieval Conference (TREC-8)*, 1999.

[6] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, chapter 1, pages 1–20. John Wiley & Sons, Ltd, 2010.

[7] Vishwas B Sharma. rake-nltk. https://csurfer.github.io/rake-nltk/_build/html/index.html, 2018. Accessed: 2021-11-07.

[8] Princeton University. About wordnet. https://wordnet.princeton.edu/, 2010. Accessed: 2021-10-20.