Question Answering Using Answer Classification and Query Expansion

LING573

David Lin Edward Wong Chris Staley

Goals

- Improve upon our previous results and approaches.
- Develop a QA system that can be more readily tinkered with and improved upon by instituting unit tests and refactoring our code to make it more modular,

Approach

- Question Classification
  - Trained a Support Vector Machine (SVM) to classify into 6 coarse buckets
  - Used the associated probabilities to assign the classification to one of three likelihoods
- Web Search
  - Use Pattern package to return snippets from Google based on the queries
  - Separated answers into individual sentences and deduped
  - Web search is cached per query question, if the query question does not exist a web search is made

Approach

- Question Reformulation
  - Replaces question topics based on NER and POS tags
  - Topic NER is based on the most common NER type
  - Uses a topicMap hash table to map NER types to acceptable POS tags to replace
  - Includes logic to ensure topic hasn't already been replaced

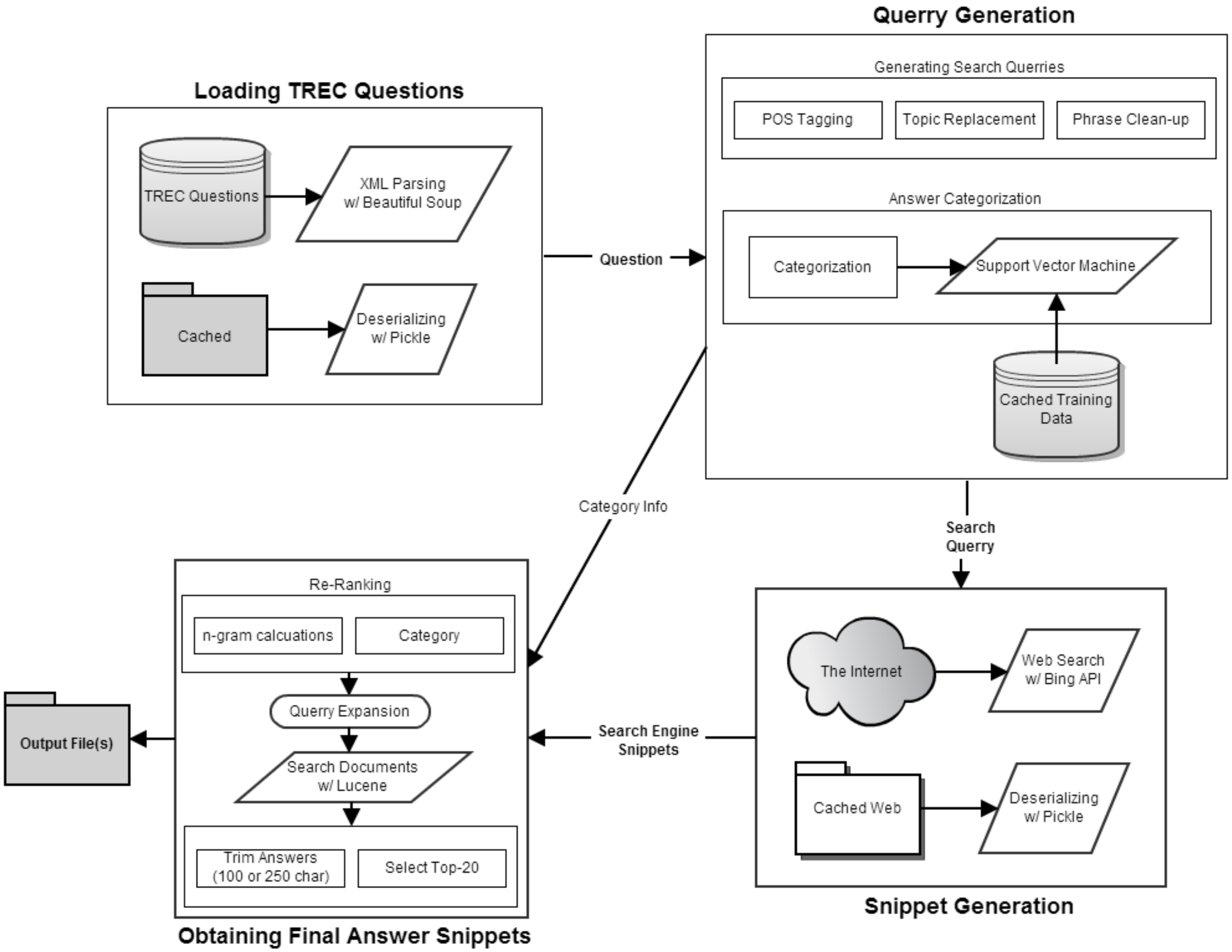
Approach

- Answer Extraction
  - N-gram redundancy method to return N-grams that appear most frequently in the document
  - Answer boosting based on predicted answer type
  - Heuristics to remove invalid answers
  - Removed answers with the topic as part of the answer
- Document Retrieval
  - Submit answers to Lucene-based IR engine to find relevant document

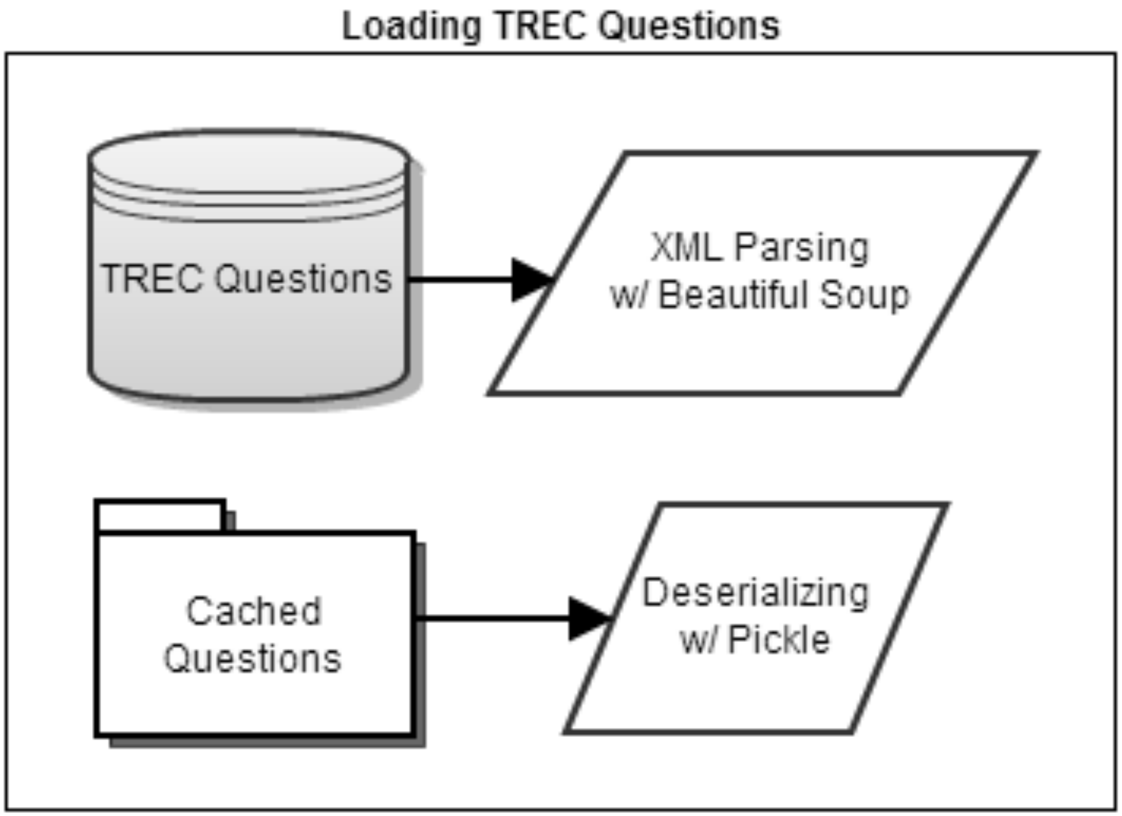
Approach

- Unit Test Cases
  - Borrowed from our combined work experiences, as well as previous deliverables.
  - Rather than risk programming something new, and risking additional breakage further down, we wanted to make sure our existing system was "bug free"
  - We were hoping that once we are confident with our code, we'd move on to additional features.

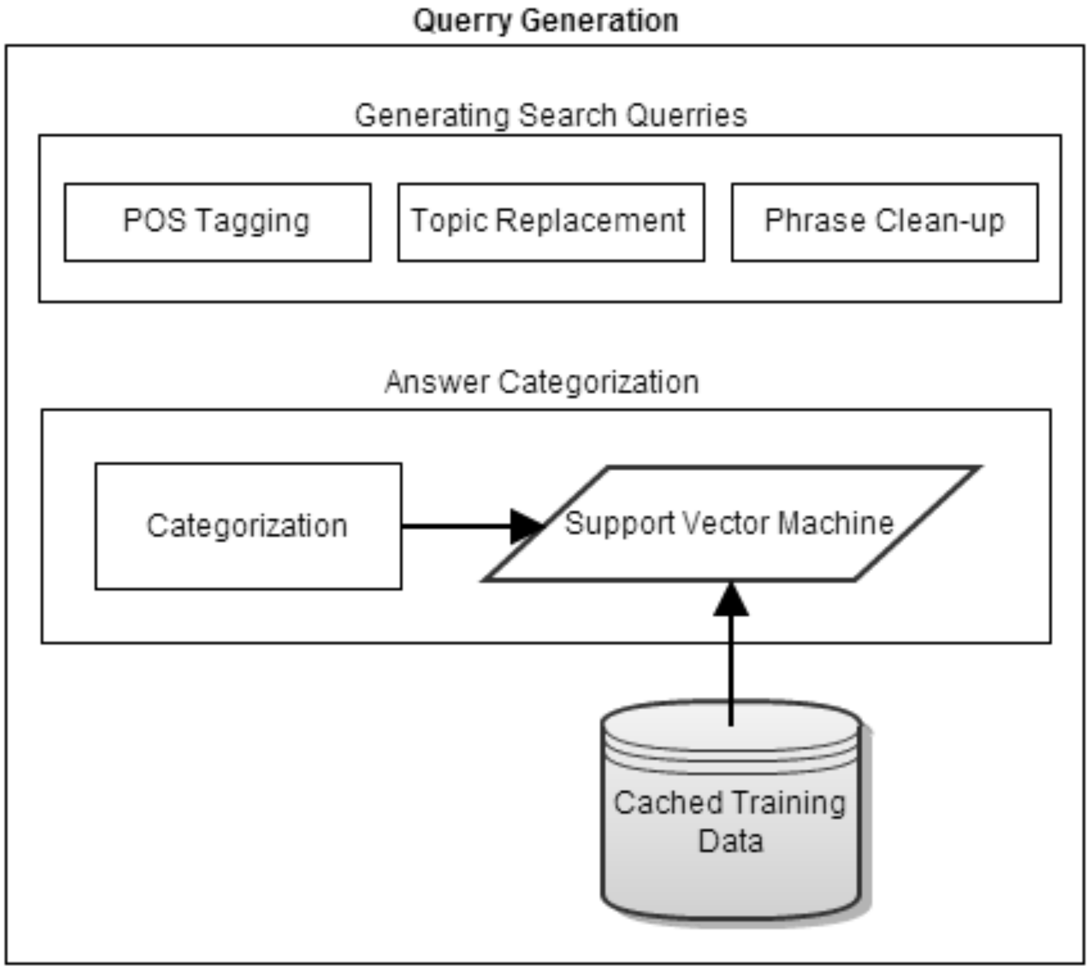
Implementation



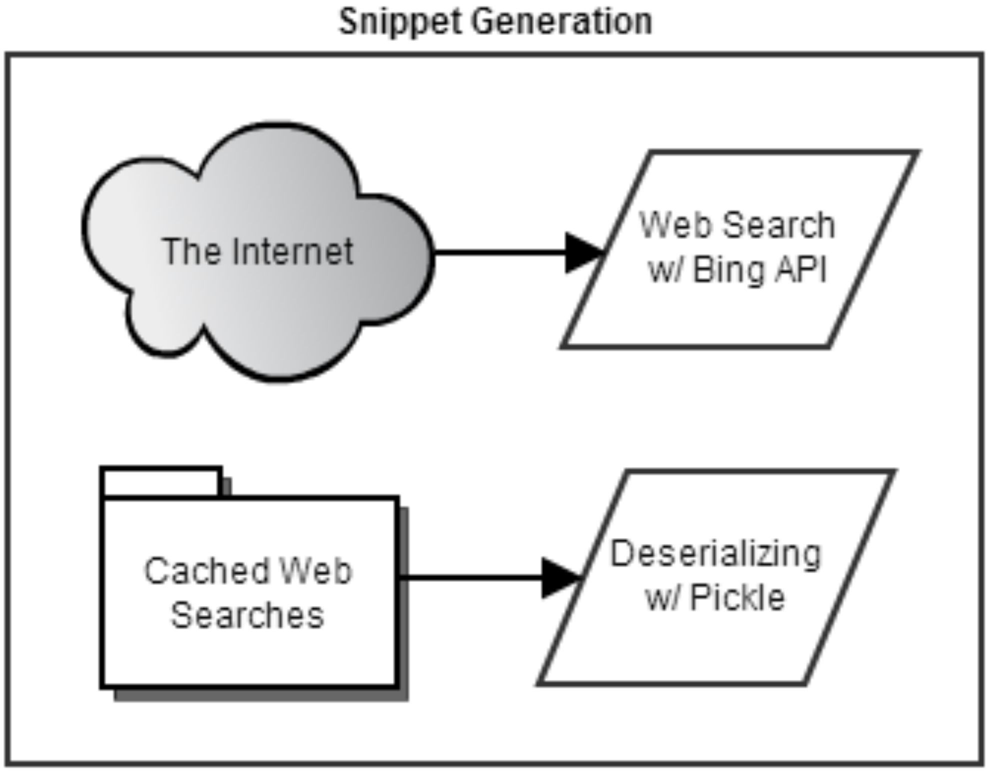
Implementation



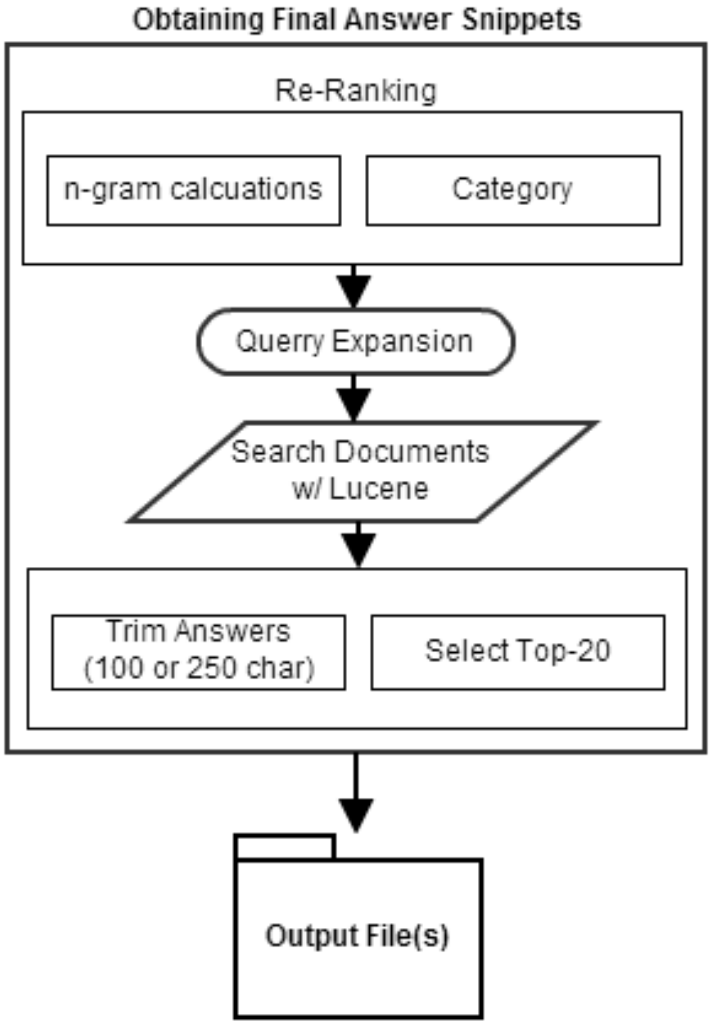
Implementation



Implementation



Implementation


Obtaining Final Answer Snippets
Re-Ranking
n-gram calcuations
Category
Querry Expansion
Search Documents w/ Lucene
Trim Answers (100 or 250 char)
Select Top-20
Output File(s)

Issues and Successes

- Attempts to improve results ended up yielding worse scores
- Code was structurally improved so that future iterations could be more easily undertaken
- Time spent working on individual test cases helped fix bugs at the function-call level; however there were still macroscopic issues that our cases still could not cover/forsee.

Unit Tests and beyond

Pros:

- Helped us isolate functions to a more testable metric
- Quicker tests for individual functions which didn't include a full run to fix
- Ability to refactor for performance

Cons:

- Should have started earlier/from the beginning...
- Didn't have full coverage, functional tests

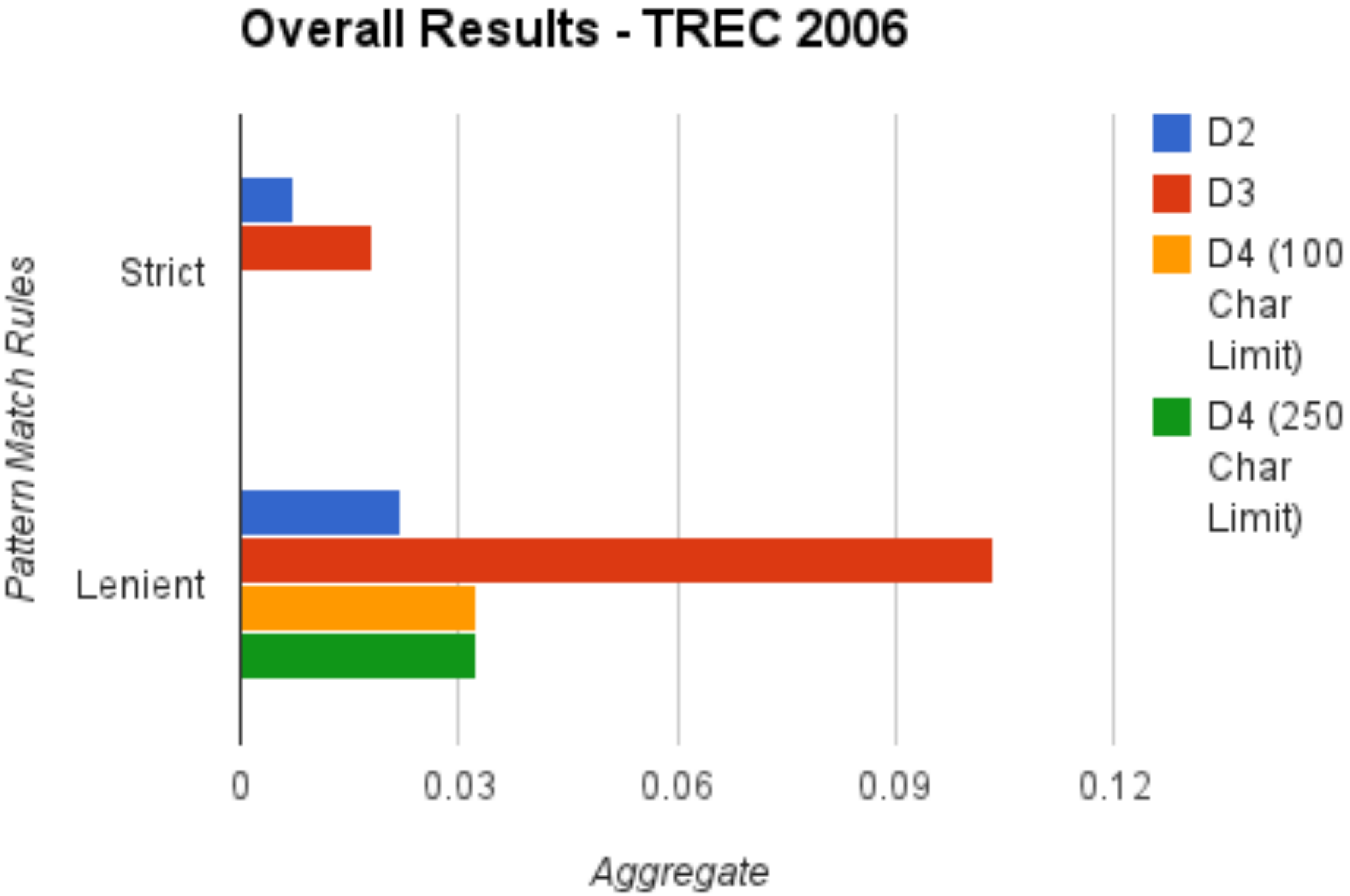
Results

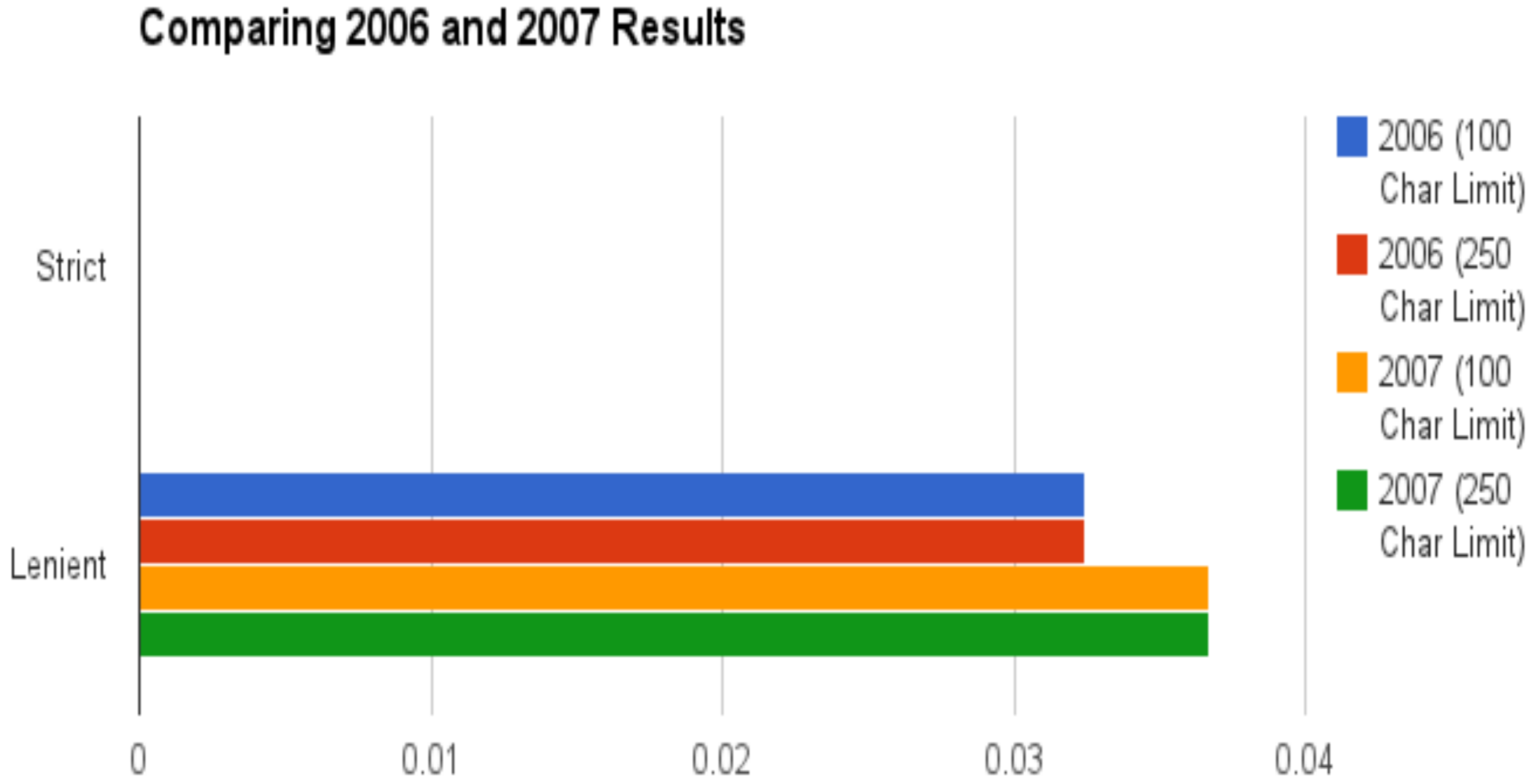
TREC-2006 Results

	100 Characters	250 Characters
Strict	0.0	0.0
Lenient	0.0324554783058	0.0324554783058

TREC-2007 Results

	100 Characters	250 Characters
Strict	0.0	0.0
Lenient	0.0366938487476	0.0366938487476


Overall Results - TREC 2006
D2
D3
D4 (100 Char Limit)
D4 (250 Char Limit)
Pattern Match Rules
Strict
Lenient
0
0.03
0.06
0.09
0.12
Aggregate


Comparing 2006 and 2007 Results
Strict
Lenient
0
0.01
0.02
0.03
0.04
2006 (100 Char Limit)
2006 (250 Char Limit)
2007 (100 Char Limit)
2007 (250 Char Limit)

Potential Improvements

- Query Formulation
  - More sophisticated parsing of the question
  - Tune the classification algorithm
- IR Engine
  - Use the documents as the main source of snippets to search rather than the web
  - Increase accuracy to improve Strict score
- Answer extraction
  - Investigate incorrect answers to improve methods for finding the answer in the candidate passages