**Introduction:**

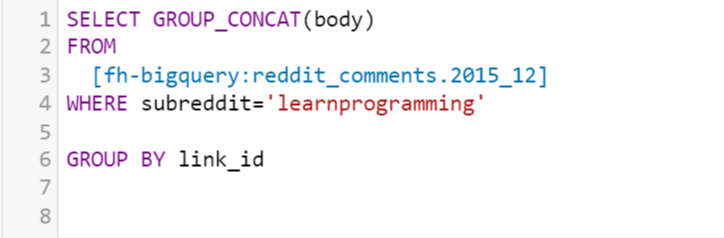
We created an information retrieval system tailored to programming related queries. The system has an ElasticSearch backend. Queries are made through a flask supported web app and their results are also delivered through this same web app. We utilized machine learning and the vector space model (powered by ElasticSearch)

**Quick Start:**

1. Install nltk (3.1), beautiful soup (11.4), jinja2 (2.8) and flask (0.10.1)
2. Download corpus (refactor if necessary or stick to default naming scheme)
3. Run ElasticSearch
4. To build index, in terminal call ‘python build\_index.py’
5. To run app, in terminal call ‘run\_index.py’

**Corpus (Corpusi) acquisition:**

We used Google BigQuery on a reddit dataset to obtain our corpus. We took all comments from the ‘/learnprogramming’ subreddit and performed a GROUP\_CONCAT by their link\_id in order to combine all posts in the same thread together (context is valuable).



For our second two corpuses, the ones consisting of documentation we pulled the documentation right from the official websites.

See es\_mapping.json for schema. Two mapping types 1) Reddit mapping type named ‘posts’ 2) Java/Python mapping type named ‘methods’

For the first corpus (the reddit corpus), all parsing was done in build\_index.py. It was very minimal and included iterating over the set cvs corresponding to our corpus and joining them by a new line symbol. Classification was done as well.

For the second corpus parsing was done with a number of heuristics (see doc\_parser.py) making heavy use of the beautiful soup library

Any attempts to change the corpuses should make sure that all directory names in the code are refactored according and the schemas will match up!

**How to run:**

**Directory Structure:**

*Templates*

All the html templates, for the homepage, the search results page and the error page are in this

*Classfication*

java\_train.py: training files for Java

java\_test.py: training files for Python

[the same format applies for Python and English training

*ProgrammersAssistant*

build\_index.py: pulls corpus data, parses necessary fields, classifies and indexes the corpus

run\_index.py: runs the app, maintains query functions

classification.py: trains and classifies documents. Supports testing also

es\_mapping.json: schema for index