Allen Majewski
Weekly report
Week 1: 3/13-3/19

Work completed
I assisted Manh on the problem of assigning metadata tags from the (size ~40k) a4 and oasis databases to each of the ~200 readiness criteria specified by rhombus.

I worked closely with Manh and discussed his planned approach using lists of keywords to assist brute force selection by hand. It differed from my plan to use word vectors, but we concluded the two approaches are similar.

At his request I wrote a python script that filters the metadata variables in csv format by user defined keywords over user defined columns. It is flexible and allows expressiveness in its use. You can choose which fields to match over and pass arbitrary numbers of keys to match.

At first glance, it is a glorified implementation of grep. But it can also mutate the output printed on the standard output to show useful information. For example, the number of matches per line, or other metrics of match quality. The default behavior is to add a column containing the keywords that matched the given line of the output.

The code was optimized to be fast and written to facilitate batch processing since we may want to run it over 10k lists of keywords and compare outputs.

The code is attached as filter_by_keywords3.py.

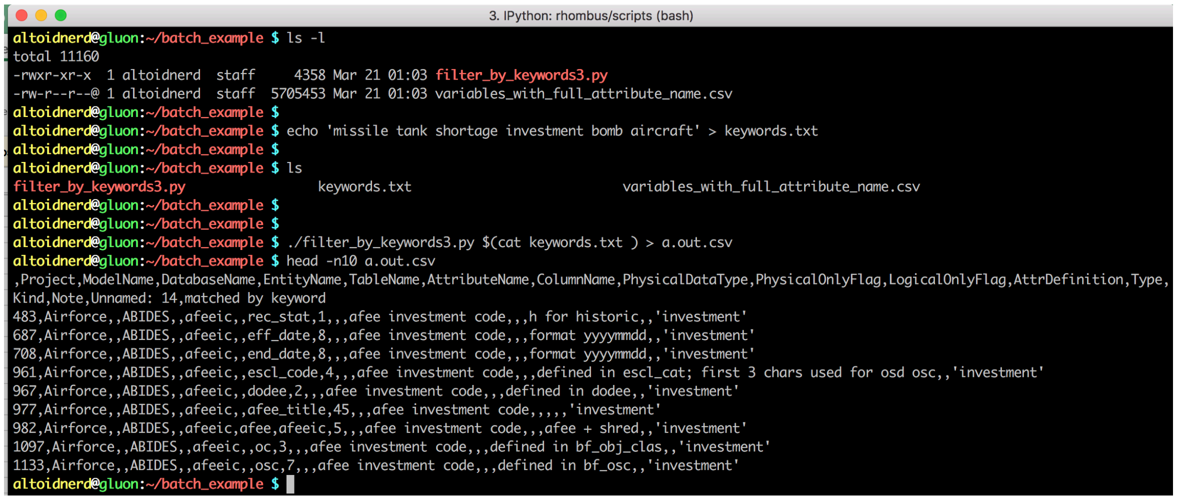
Some example uses for batch scripting:

```
EXAMPLE 1: run on keywords from command line
allen@computer ~/ > ./filter_by_keywords3.py missile fuel aircraft silo bomb person casualty




EXAMPLE 2: quickly count the number of matches for a given keyword
allen@computer ~/ > ./filter_by_keywords3.py active | grep -c .
334


EXAMPLE 3:  read from a keyword list contained in a text file and write from stdout to a new file

#!/bin/bash
echo missile fuel tank bomb shortage runway > keywords.txt
./filter_by_keywords3.py $(cat keywords.txt) > example.out.csv
```

```
EXAMPLE 4: iterate over a set of keyword files keywords1.txt, keywords2.txt, keywords3.txt,
keywords4.txt ... and generate corresponding csvs
```

```
#!/bin/bash
for keyfile in $(ls keywords*.txt); do
   ./filter_by_keywords3.py $(cat $keyfile) > $keyfile.out.csv
done
```

```
This will create keywords1.txt.out.csv, keywords2.txt.out.csv, ...one for each input.
```



```
altoidnerd@gluon:~/batch_example $ ls -l
total 11160
-rwxr-xr-x  1 altoidnerd  staff     4358 Mar 21 01:03 filter_by_keywords3.py
-rw-r--r--@ 1 altoidnerd  staff  5705453 Mar 21 01:03 variables_with_full_attribute_name.csv
altoidnerd@gluon:~/batch_example $
altoidnerd@gluon:~/batch_example $ echo 'missile tank shortage investment bomb aircraft' > keywords.txt
altoidnerd@gluon:~/batch_example $
altoidnerd@gluon:~/batch_example $ ls
filter_by_keywords3.py           keywords.txt                      variables_with_full_attribute_name.csv
altoidnerd@gluon:~/batch_example $
altoidnerd@gluon:~/batch_example $
altoidnerd@gluon:~/batch_example $ ./filter_by_keywords3.py $(cat keywords.txt ) > a.out.csv
altoidnerd@gluon:~/batch_example $ head -n10 a.out.csv
,Project,ModelName,DatabaseName,EntityName,TableName,AttributeName,ColumnName,PhysicalDataType,PhysicalOnlyFlag,LogicalOnlyFlag,AttrDefinition,Type,
Kind,Note,Unnamed: 14,matched by keyword
483,Airforce,,ABIDES,,afeeic,,rec_stat,1,,,afee investment code,,h for historic,,'investment'
687,Airforce,,ABIDES,,afeeic,,eff_date,8,,,afee investment code,,,format yyyymmdd,,'investment'
708,Airforce,,ABIDES,,afeeic,,end_date,8,,,afee investment code,,,format yyyymmdd,,'investment'
961,Airforce,,ABIDES,,afeeic,,escl_code,4,,,afee investment code,,,defined in escl_cat; first 3 chars used for osd osc,,'investment'
967,Airforce,,ABIDES,,afeeic,,dodee,2,,,afee investment code,,,defined in dodee,,'investment'
977,Airforce,,ABIDES,,afeeic,,afee_title,45,,,afee investment code,,,,,'investment'
982,Airforce,,ABIDES,,afeeic,afee,afeeic,5,,,afee investment code,,,afee + shred,,'investment'
1097,Airforce,,ABIDES,,afeeic,,oc,3,,,afee investment code,,,defined in bf_obj_clas,,'investment'
1133,Airforce,,ABIDES,,afeeic,,osc,7,,,afee investment code,,,defined in bf_osc,,'investment'
altoidnerd@gluon:~/batch_example $
```

Fig 1: illustration of use of the script from the command line.

I also assisted Manh with other simple automation tasks and presented some keyword lists for training readiness at his request.

Recommendations

Metrics for success in this problem generally should be clarified and tested. Orders of magnitude for the size of the desired output (such as, the number of db matches per ID) should be constrained.

I would feel more comfortable if the methodology relied less on human brute force labor. Currently, we are selecting keywords by inspection. Subjectivity would be reduced if we trained a keyword selector, or at least tested all/many subsets of the full text as keyword sets and converged the output results in a systematic way.

Alternatively, word association vectors could be trained by scraping the web for airforce related documents. My initial idea was to utilize word vectors to generate similarity scores between rhombus ID tags and variable definitions as a solution to the problem. This would eliminate the subjectivity that threatens to bias the results. Manh convinced me the standard word vector databases are not trained narrowly enough in this case and I agreed.