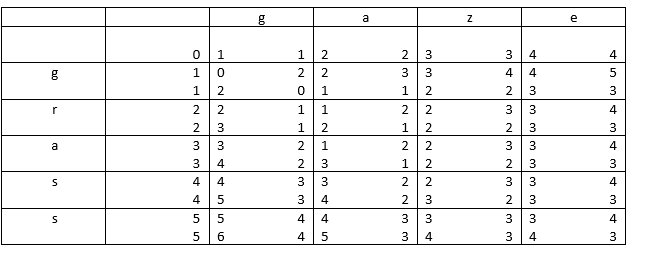
Collaborators: Ben Hlewka, Calvin Lee, Dan Cam, Derrick Wai, Cam McIntrye  
Consultations: Class slides, An Introduction to Information Retrieval by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze   
  
1.  
a) lemon -> lemon$, emon$l, mon$le, on$lem, n$lemo, $lemon  
 monster -> monster$ onster$m, nster$mo, ster$mon, ter$mons, er$monst, r$monste, $monster  
b) mon\* returns: monster. Terms: ster$mon, , ter$mons, er$monst, r$monste, and $monster, exist in the index and are returned because $mon\* is searched up. The $ coming before mon, along with characters, indicates that there are characters that exist after mon, and that mon is the start of the term.  
 m\*n returns: nothing. n$m\* is searched and since there are no matches in the index, nothing is returned.  
 \*mon returns: lemon. Term: mon$le exists in the index and is returned because mon$\* is looked up in the index. Mon appearing before $ indicates that it is not the start of the term, and the le following indicates that there are characters that appear before mon in the term  
c) lemon -> $le lem emo mon on$  
 monster -> $mo mon ons nst ste ter er$  
d) mon\* returns: monster. Trigrams: $mo and mon are searched for, indicating monster as a result. The terms are then compared with the original query and match, thus resulting in monster.   
 m\*n: returns: nothing. Trigrams: $m and n$ are searched for, with no returning matches.   
 \*mon: returns lemon. Trigrams: mon and on$ are searched for, indicating lemon as a resulting match. These terms are compared with the original query and do indeed match, therefore lemon is a result.

1/1

2.  
a)   
this: 1  
assignment: 1  
requires: 1  
work: 1, 3, 4  
i: 2, 3, 4  
can’t: 2  
more: 2  
than: 2  
10: 2  
hours: 2  
a: 2  
day: 2  
don’t: 3  
think: 3  
your: 3  
approach: 3  
will: 4  
you: 4  
do: 4  
my: 4  
work: 1, 3, 4  
while: 4  
relax: 4  
  
b) work AND I OR assignment: 1, 2, 3, 4  
c) work AND NOT work: no return  
d) will AND work AND NOT I: no return  
1/1

3.  
a)

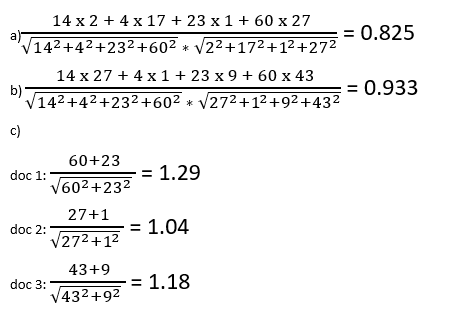


b)  
delete r -> gass  
replace s -> gazs  
replace s -> gaze

1/1

4.  
a) doc id’s 19 and 20 are skipped thought process is right, but if you look closely at the algorithm, it will advance both pointers after the match at 17 and never take the skip from 17 to 31  
b) no documents are skipped  
c) The results are different since the query is asking for Alice OR Bob. All documents in the postings list for Alice must be included as each document includes only Alice. Skipping over documents would result in a loss of documents that contained Alice even if the skip pointer would be less than the current pointer for Bob

0.9/1

5.  


Double-check your calculations because the results should never be greater than 1

0.7/1

Total: 4.6 / 5