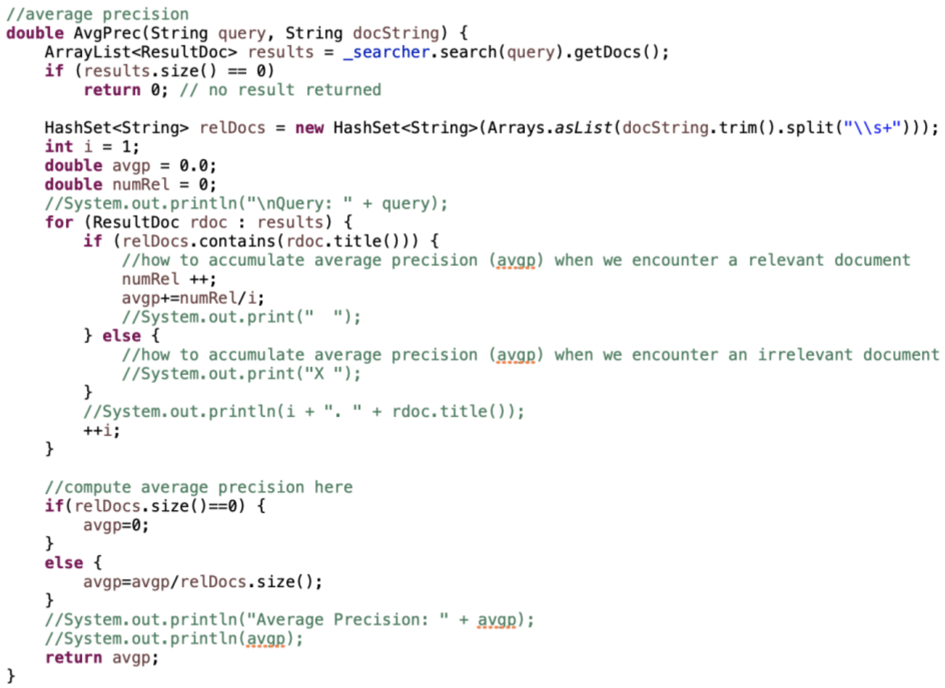
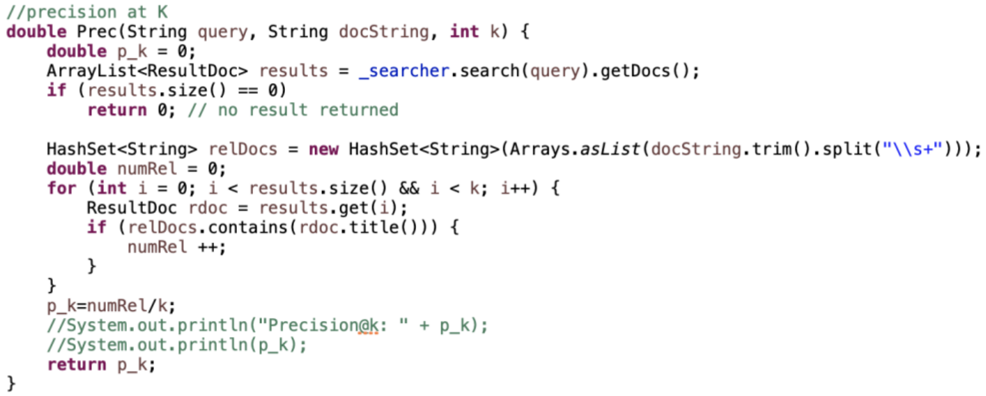
Yunlu Li
CS 4780
MP2

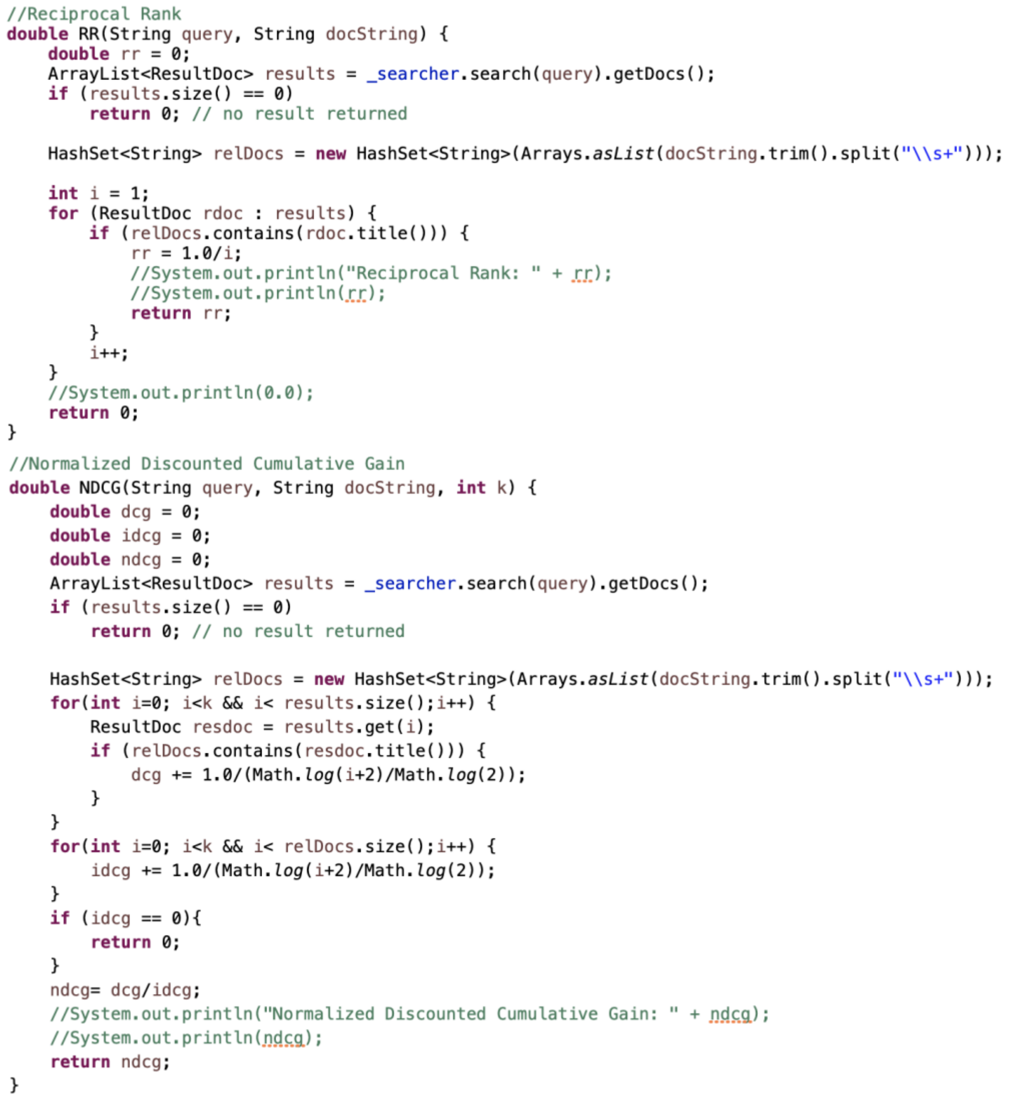
Question 1



```
//average precision
double AvgPrec(String query, String docString) {
    ArrayList<ResultDoc> results = _searcher.search(query).getDocs();
    if (results.size() == 0)
        return 0; // no result returned

    HashSet<String> relDocs = new HashSet<String>(Arrays.asList(docString.trim().split("\\s+")));
    int i = 1;
    double avgp = 0.0;
    double numRel = 0;
    //System.out.println("\nQuery: " + query);
    for (ResultDoc rdoc : results) {
        if (relDocs.contains(rdoc.title())) {
            //how to accumulate average precision (avgp) when we encounter a relevant document
            numRel ++;
            avgp+=numRel/i;
            //System.out.print("  ");
        } else {
            //how to accumulate average precision (avgp) when we encounter an irrelevant document
            //System.out.print("X ");
        }
        //System.out.println(i + ". " + rdoc.title());
        ++i;
    }

    //compute average precision here
    if(relDocs.size()==0) {
        avgp=0;
    }
    else {
        avgp=avgp/relDocs.size();
    }
    //System.out.println("Average Precision: " + avgp);
    //System.out.println(avgp);
    return avgp;
}
```



```
//precision at K
double Prec(String query, String docString, int k) {
    double p_k = 0;
    ArrayList<ResultDoc> results = _searcher.search(query).getDocs();
    if (results.size() == 0)
        return 0; // no result returned

    HashSet<String> relDocs = new HashSet<String>(Arrays.asList(docString.trim().split("\\s+")));
    double numRel = 0;
    for (int i = 0; i < results.size() && i < k; i++) {
        ResultDoc rdoc = results.get(i);
        if (relDocs.contains(rdoc.title())) {
            numRel ++;
        }
    }
    p_k=numRel/k;
    //System.out.println("Precision@k: " + p_k);
    //System.out.println(p_k);
    return p_k;
}
```



```
//Reciprocal Rank
double RR(String query, String docString) {
    double rr = 0;
    ArrayList<ResultDoc> results = _searcher.search(query).getDocs();
    if (results.size() == 0)
        return 0; // no result returned

    HashSet<String> relDocs = new HashSet<String>(Arrays.asList(docString.trim().split("\\s+")));

    int i = 1;
    for (ResultDoc rdoc : results) {
        if (relDocs.contains(rdoc.title())) {
            rr = 1.0/i;
            //System.out.println("Reciprocal Rank: " + rr);
            //System.out.println(rr);
            return rr;
        }
        i++;
    }
    //System.out.println(0.0);
    return 0;
}
//Normalized Discounted Cumulative Gain
double NDCG(String query, String docString, int k) {
    double dcg = 0;
    double idcg = 0;
    double ndcg = 0;
    ArrayList<ResultDoc> results = _searcher.search(query).getDocs();
    if (results.size() == 0)
        return 0; // no result returned

    HashSet<String> relDocs = new HashSet<String>(Arrays.asList(docString.trim().split("\\s+")));
    for(int i=0; i<k && i< results.size();i++) {
        ResultDoc resdoc = results.get(i);
        if (relDocs.contains(resdoc.title())) {
            dcg += 1.0/(Math.log(i+2)/Math.log(2));
        }
    }
    for(int i=0; i<k && i< relDocs.size();i++) {
        idcg += 1.0/(Math.log(i+2)/Math.log(2));
    }
    if (idcg == 0){
        return 0;
    }
    ndcg= dcg/idcg;
    //System.out.println("Normalized Discounted Cumulative Gain: " + ndcg);
    //System.out.println(ndcg);
    return ndcg;
}
```

The final MAP/P@10/MRR/NDCG@10 performance from each ranking function:

	BM25	Dot Product
MAP	0.27524149426286965	0.28405788399801385
P@10	0.34623655913978496	0.35053763440860214
MRR	0.674808203578191	0.6905643241833787
NDCG@10	0.42603032215453884	0.4312359204708965

Note: the *defaultNumResults* are 1000.

Question 2

First, I only removed the lower-case filter or length filter. The result is shown in the table below. As we can see, there is almost no effect on the effectiveness after removing the filters. I realized that this retrieval is about academic paper, so titles are already standardized. This means the lower-case or length filter does not have much effect under such circumstances.
(Note: the *defaultNumResults* are 1000).

	Original BM25	BM25 without Lower-case Filter	BM25 without Length Filter
MAP	0.27524149426286965	0.27524149426286965	0.2773372647192524
P@10	0.34623655913978496	0.34623655913978496	0.3483870967741936
MRR	0.674808203578191	0.674808203578191	0.6880613997715862
NDCG@10	0.42603032215453884	0.42603032215453884	0.4300451674422876

Then, I tested the effect of document analyzer on retrieval effectiveness by disabling stopword removal. Overall, lack of stopword removal makes retrieval less effective because all four metrics decrease, but the drop is very small. To further test the conclusion, I disabled stopword removal and stem. As we can see from the table below, all metrics showed a great drop. As we know, the BM25 uses bag-of-word as foundation, so lack of stemming makes it hard to transform words into original forms and thus BM25 becomes less effective. Therefore, we conclude that the filter pipeline built in the document analyzer makes the retrieval effective. Without those filters, the retrieval becomes less effective.
(Note: the *defaultNumResults* are 1000).

	Original BM25	BM25 without Stopword Removal	BM25 without Stopword Removal and Stem
MAP	0.27524149426286965	0.27133160186921407	0.20617530855162403
P@10	0.34623655913978496	0.3344086021505377	0.2752688172043011
MRR	0.674808203578191	0.6739268398734564	0.6487186601847095
NDCG@10	0.42603032215453884	0.4129613899798967	0.3501253435060279

Question 3

One-tail p-value:

	MAP	P@10	MRR	NDCG@10
Paired t-test	0.032627082	0.294474856	0.160087207	0.246149344
Wilcoxon signed-rank test	0.0116	0.31918	0.22363	0.32636

Two-tail p-value:

	MAP	P@10	MRR	NDCG@10
Paired t-test	0.065254165	0.588949712	0.320174413	0.492298688
Wilcoxon signed-rank test	0.0232	0.63836	0.44726	0.65272

From the table above, except MAP, we can see that all two-tail p-values are bigger than the significance level 0.05, which suggests that we cannot reject the null hypothesis. This means that the observed difference in P@10, MRR and NDCG@10 is insignificant, so we cannot determine which ranking algorithm is better for those matrices.

For MAP, the two-tail p-value of paired t-test is bigger than 0.05 whereas Wilcoxon signed-rank test has two-tail p-value less than 0.05. Then let us take a closer look at one-tail p-value. The one-tail p-value from both paired t-test and Wilcoxon signed-rank test are less than 0.05. This means we reject the null hypothesis. Dot Product is better than BM25 under MAP in both paired t-test and Wilcoxon signed-rank test.