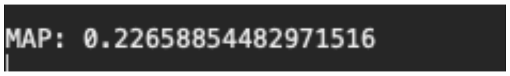
MP2—Retrieval Evaluations

1. Copy and paste your implementation of each evaluation function into your report, together with the corresponding final MAP/P@10/MRR/NDCG@10 performance you get from each ranking function. (40pts + 20pts)

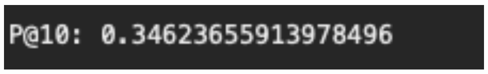
   MAP:

```
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;
        if (i < Evaluate.k + 1) {
          System.out.print("  ");}}
      else {
        //how to accumulate average precision (avgp) when we encounter an
irrelevant document
        if ( i < Evaluate.k + 1) {
        System.out.print("X ");}
      }

      if (i < Evaluate.k + 1) {
        System.out.println(i + ". " + rdoc.title());}
      ++i;
    }

    //compute average precision here
    // avgp = ?
    if (numRel > 0) {
      avgp = avgp / relDocs.size();}
    else {
      avgp = 0.0;}
    System.out.println("Average Precision: " + avgp);
    return avgp;
  }
```


```
MAP: 0.22658854482971516
```

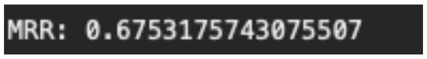
P@10:

```
//precision at K
  double Prec(String query, String docString, int k) {
    double p_k = 0;

    k = Evaluate.k;
    //your code for computing precision at K here

    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.split(" ")));

    double relevantNum = 0;
    int i = 0;
    System.out.println("\nQuery: " + query);
    for (ResultDoc rdoc : results) {
      if (relDocs.contains(rdoc.title())) {
        relevantNum ++;
        System.out.print("  ");
      } else {
        System.out.print("X ");
      }
      System.out.println(i + ". " + rdoc.title());
      ++i;

      if (i == k) {
        break;}
    }
    p_k = relevantNum / k;
    return p_k;
  }
```


```
P@10: 0.34623655913978496
```

MRR:

```
//Reciprocal Rank
  double RR(String query, String docString) {
    double rr = 0;
    //your code for computing Reciprocal Rank here
    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.split(" ")));

    int OccurTime = 1;
    System.out.println("\nQuery: " + query);
    for (ResultDoc rdoc : results) {
      if (relDocs.contains(rdoc.title())) {
        break;
      } else {
        System.out.print("X ");
      }
      System.out.println(OccurTime + ". " + rdoc.title());
      ++OccurTime;
    }
    rr = 1.0 / OccurTime;

    return rr;
  }
```

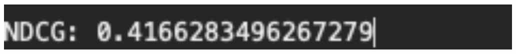


```
MRR: 0.6753175743075507
```

NDCG@10:

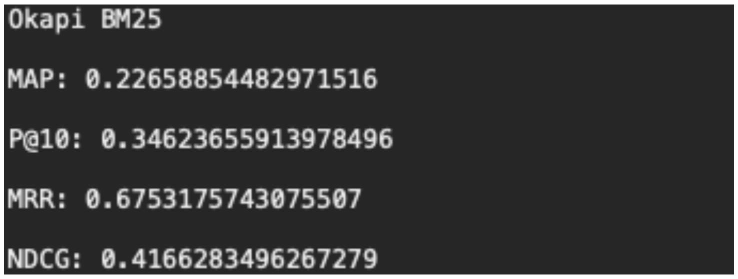
```
  //Normalized Discounted Cumulative Gain
  double NDCG(String query, String docString, int k) {

    double ndcg = 0.0;
    double dcg = 0.0;
    double idcg = 0.0;
    k = Evaluate.k;
    //your code for computing Normalized Discounted Cumulative Gain here

    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.split(" ")));
```

```
        // compute ideal dcg.
        if (relDocs.size() == 0) {
          return 0;}
        else if (relDocs.size() < k) {
          for (int i = 1; i < relDocs.size() + 1; i++) {
            idcg += Math.log(2.0) / Math.log(1 + i);}}
        //idcg += (Math.pow(2, 1) - 1.0)  / Math.log(i + 1) * Math.log(2.0);
        else {
          for (int i = 1; i < k + 1; i++) {
            idcg += Math.log(2.0) / Math.log(1 + i);}}


        // compute dcg.
        int i = 1;
        System.out.println("\nQuery: " + query);
        for (ResultDoc rdoc : results) {
          if (relDocs.contains(rdoc.title())) {
            dcg += Math.log(2.0) / Math.log(1 + i);
            //dcg += (Math.pow(2, 1) - 1.0)  / Math.log(i + 1) * Math.log(2.0);
            System.out.print("  ");
          } else {
            System.out.print("X ");
          }
          System.out.println(i + ". " + rdoc.title());
          ++i;
          if (i > k) break;
        }

        ndcg = dcg / idcg;
        if(idcg == 0)
          ndcg = 0;

        return ndcg;
      }
```


```
NDCG: 0.4166283496267279
```

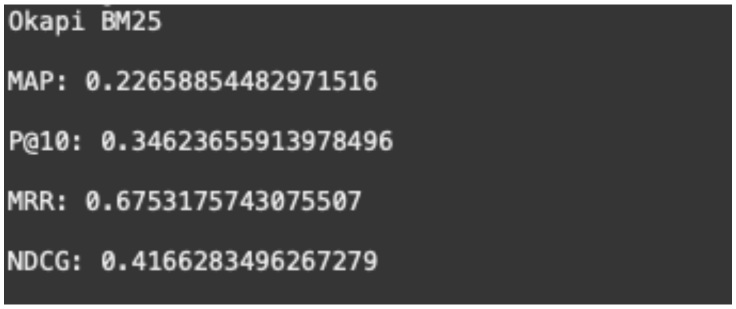
2. In `edu.virginia.cs.index.SpecialAnalyzer.java`, we defined a special document analyzer to process the document/query for retrieval purpose. Basically, we built up a pipeline with filters of `LowerCaseFilter`, `LengthFilter`, `StopFilter`, and `PorterStemFilter`. Please disable some of the filters, e.g., without stopword removal or stemming, and test the new analyzer with the BM25 model. What is your conclusion about the effect of document analyzer on retrieval effectiveness? (20pts) *Note: this analyzer has to be used in both indexing time and query time!*

   Baseline:


```
Okapi BM25

MAP: 0.22658854482971516

P@10: 0.34623655913978496

MRR: 0.6753175743075507

NDCG: 0.4166283496267279
```

I. Remove LowerCaseFilter

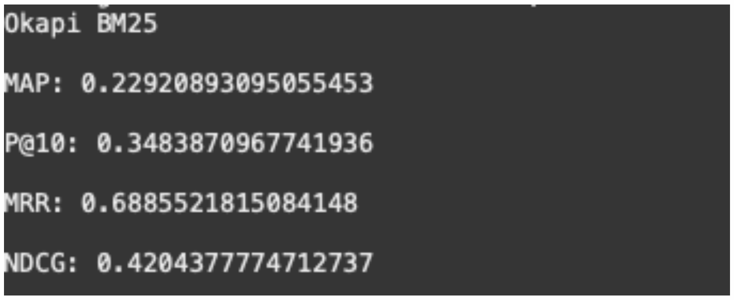
Evaluate:


```
Okapi BM25

MAP: 0.22658854482971516

P@10: 0.34623655913978496

MRR: 0.6753175743075507

NDCG: 0.4166283496267279
```

No effect, because there is no upper letter in the txt.

II. Remove LengthFilter

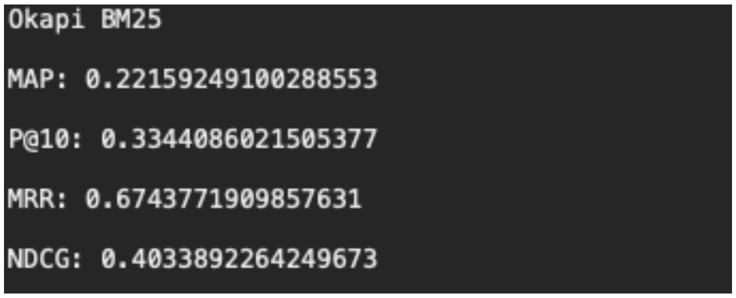
Evaluate:


```
Okapi BM25

MAP: 0.22920893095055453

P@10: 0.3483870967741936

MRR: 0.6885521815084148

NDCG: 0.4204377774712737
```

All increase a bit. The LengthFilter may filter some right answers.

III. Remove StopFilter

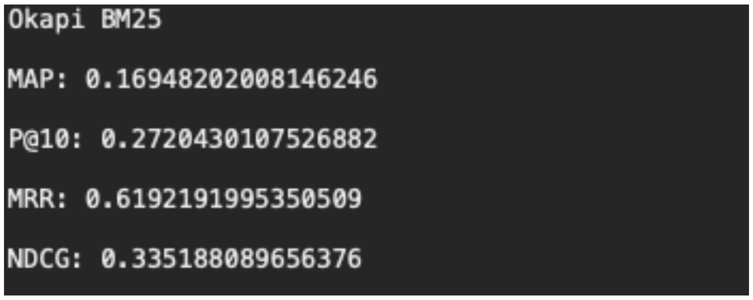
Evaluate:


```
Okapi BM25

MAP: 0.22159249100288553

P@10: 0.3344086021505377

MRR: 0.6743771909857631

NDCG: 0.4033892264249673
```

All decreas a bit. The stopword may connect some right answers.

IV. Remove PorterStemFilter

Evaluate:



```
Okapi BM25

MAP: 0.16948202008146246

P@10: 0.2720430107526882

MRR: 0.6192191995350509

NDCG: 0.335188089656376
```

All decrease, and NDCG decreases more. The stemmer may produce some right answers.

3. In question 1, we compared the ranking model BM25 with TFIDF only by the mean value of the four evaluation metrics. As we already know that statistical test is necessary when we only have a small evaluation data set (93 queries in our case). Let's compute and report the p-value from paired t-test and Wilcoxon signed-rank test for the comparison over all four metrics. Based on your statistical test results, which is a better ranking algorithm? (20pts) *Note: you do not need to implement the calculation of those tests. You can find any Java/Python/Matlab implementation for this purpose, and just prepare the required input for it.*

Paired t-test:

MAP:

	0.1733941	**0.2527667**
Mean	0.22716675	0.23329796
Variance	0.03024771	0.03016648
Observations	92	92
Pearson Correlation	0.96346611	
Hypothesized Mean Difference	0	
df	91	
t Stat	-1.2517496	
P(T<=t) one-tail	0.10693457	
t Critical one-tail	1.66177116	
P(T<=t) two-tail	0.21386914	
t Critical two-tail	1.98637715	

These two algorithm performs equally in MAP.

P@10:

t-Test: Paired Two Sample for Means		
	0.3	0.4
Mean	0.34673913	0.35
Variance	0.06185738	0.06230769
Observations	92	92
Pearson Correlation	0.95318613	
Hypothesized Mean Difference	0	
df	91	
t Stat	-0.4102148	
P(T<=t) one-tail	0.3413063	
t Critical one-tail	1.66177116	
P(T<=t) two-tail	0.68261261	
t Critical two-tail	1.98637715	

These two algorithm performs equally in MAP.

MRR:

t-Test: Paired Two Sample for Means		
	0.4	1
Mean	0.35	0.68765255
Variance	0.06230769	0.1484242
Observations	92	92
Pearson Correlation	0.67411277	
Hypothesized Mean Difference	0	

df	91
t Stat	-11.373983
P(T<=t) one-tail	1.8022E-19
t Critical one-tail	1.66177116
P(T<=t) two-tail	3.6044E-19
t Critical two-tail	1.98637715

The two-tailed p-value is < 0.05.

This result shows there is significant difference between these two ranking algorithm. The one-tailed p value is also < 0.05, which means the TFIDF better than BM25 in RR.

NDCG:

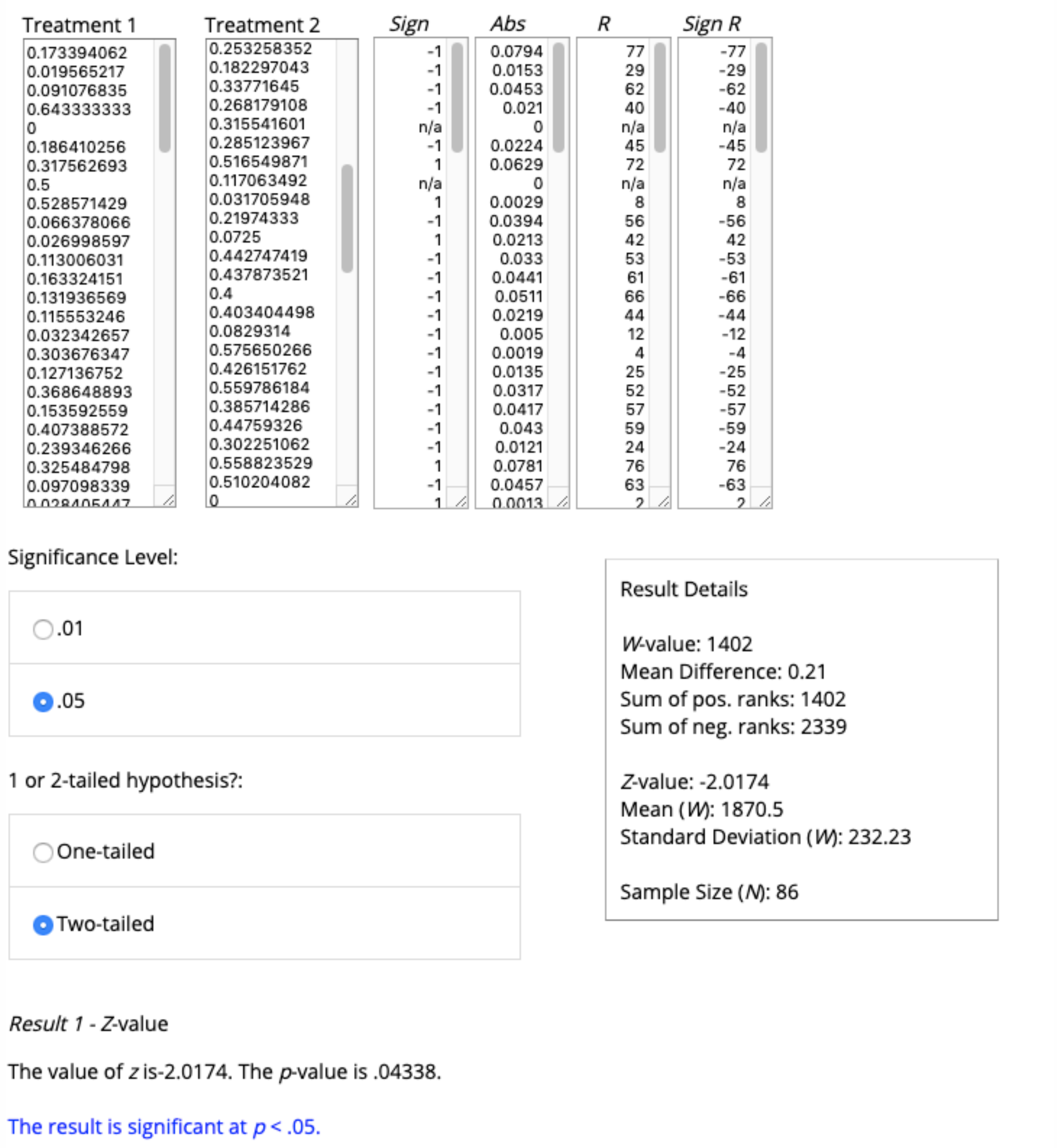
t-Test: Paired Two Sample for Means		
	0.424926	0.4943572
Mean	0.41653816	0.42084282
Variance	0.07229955	0.07222937
Observations	92	92
Pearson Correlation	0.96490746	
Hypothesized Mean Difference	0	
df	91	
t Stat	-0.5797592	
P(T<=t) one-tail	0.28175484	
t Critical one-tail	1.66177116	
P(T<=t) two-tail	0.56350967	
t Critical two-tail	1.98637715	

These two algorithm performs equally in NDCG.

Wilcoxon signed-rank test:

I use this website: https://www.socscistatistics.com/tests/signedranks/default2.aspx

MAP:



Treatment 1	Treatment 2	*Sign*	*Abs*	*R*	*Sign R*
0.173394062	0.253258352	-1	0.0794	77	-77
0.019565217	0.182297043	-1	0.0153	29	-29
0.091076835	0.33771645	-1	0.0453	62	-62
0.643333333	0.268179108	-1	0.021	40	-40
0	0.315541601	n/a	0	n/a	n/a
0.186410256	0.285123967	-1	0.0224	45	-45
0.317562693	0.516549871	1	0.0629	72	72
0.5	0.117063492	n/a	0	n/a	n/a
0.528571429	0.031705948	1	0.0029	8	8
0.066378066	0.21974333	-1	0.0394	56	-56
0.026998597	0.0725	1	0.0213	42	42
0.113006031	0.442747419	-1	0.033	53	-53
0.163324151	0.437873521	-1	0.0441	61	-61
0.131936569	0.4	-1	0.0511	66	-66
0.115553246	0.403404498	-1	0.0219	44	-44
0.032342657	0.0829314	-1	0.005	12	-12
0.303676347	0.575650266	-1	0.0019	4	-4
0.127136752	0.426151762	-1	0.0135	25	-25
0.368648893	0.559786184	-1	0.0317	52	-52
0.153592559	0.385714286	-1	0.0417	57	-57
0.407388572	0.44759326	-1	0.043	59	-59
0.239346266	0.302251062	-1	0.0121	24	-24
0.325484798	0.558823529	1	0.0781	76	76
0.097098339	0.510204082	-1	0.0457	63	-63
0.028405447	0	1	0.0013	2	2

Significance Level:

( ) .01 [✓ .05]

1 or 2-tailed hypothesis?:

( ) One-tailed [✓ Two-tailed]

Result Details

W-value: 1402
Mean Difference: 0.21
Sum of pos. ranks: 1402
Sum of neg. ranks: 2339

Z-value: -2.0174
Mean (*W*): 1870.5
Standard Deviation (*W*): 232.23

Sample Size (*N*): 86

Result 1 - Z-value

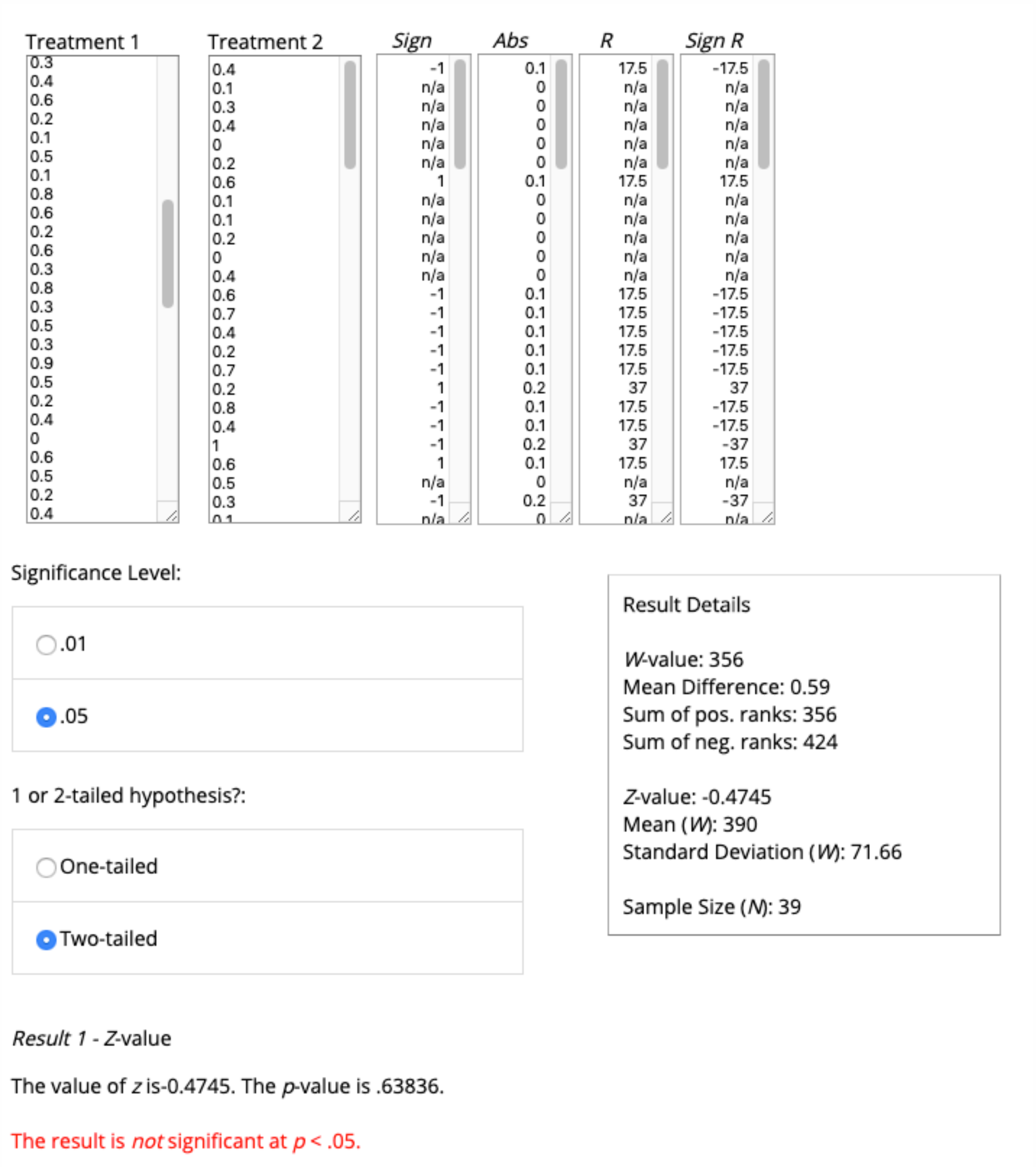
The value of *z* is-2.0174. The *p*-value is .04338.

The result is significant at *p* < .05.

The two-tailed p-value is 0.04338

This result shows there is significant difference between these two ranking algorithm. The one-tailed p value is 0.02169 < 0.05, which means the TFIDF better than BM25 in MAP.

P@10:



Treatment 1	Treatment 2	*Sign*	*Abs*	*R*	*Sign R*
0.3	0.4	-1	0.1	17.5	-17.5
0.4	0.1	n/a	0	n/a	n/a
0.6	0.3	n/a	0	n/a	n/a
0.2	0.4	n/a	0	n/a	n/a
0.1	0	n/a	0	n/a	n/a
0.5	0.2	n/a	0	n/a	n/a
0.1	0.6	1	0.1	17.5	17.5
0.8	0.1	n/a	0	n/a	n/a
0.6	0.1	n/a	0	n/a	n/a
0.2	0.2	n/a	0	n/a	n/a
0.6	0	n/a	0	n/a	n/a
0.3	0.4	n/a	0	n/a	n/a
0.8	0.6	-1	0.1	17.5	-17.5
0.3	0.7	-1	0.1	17.5	-17.5
0.5	0.4	-1	0.1	17.5	-17.5
0.3	0.2	-1	0.1	17.5	-17.5
0.9	0.7	-1	0.1	17.5	-17.5
0.5	0.2	1	0.2	37	37
0.2	0.8	-1	0.1	17.5	-17.5
0.4	0.4	-1	0.1	17.5	-17.5
0	1	-1	0.2	37	-37
0.6	0.6	1	0.1	17.5	17.5
0.5	0.5	n/a	0	n/a	n/a
0.2	0.3	-1	0.2	37	-37
0.4	0.1	n/a	0	n/a	n/a

Significance Level:

.01 [✓ .05]

1 or 2-tailed hypothesis?:

One-tailed [✓ Two-tailed]

Result Details

W-value: 356
Mean Difference: 0.59
Sum of pos. ranks: 356
Sum of neg. ranks: 424

Z-value: -0.4745
Mean (*W*): 390
Standard Deviation (*W*): 71.66

Sample Size (*N*): 39

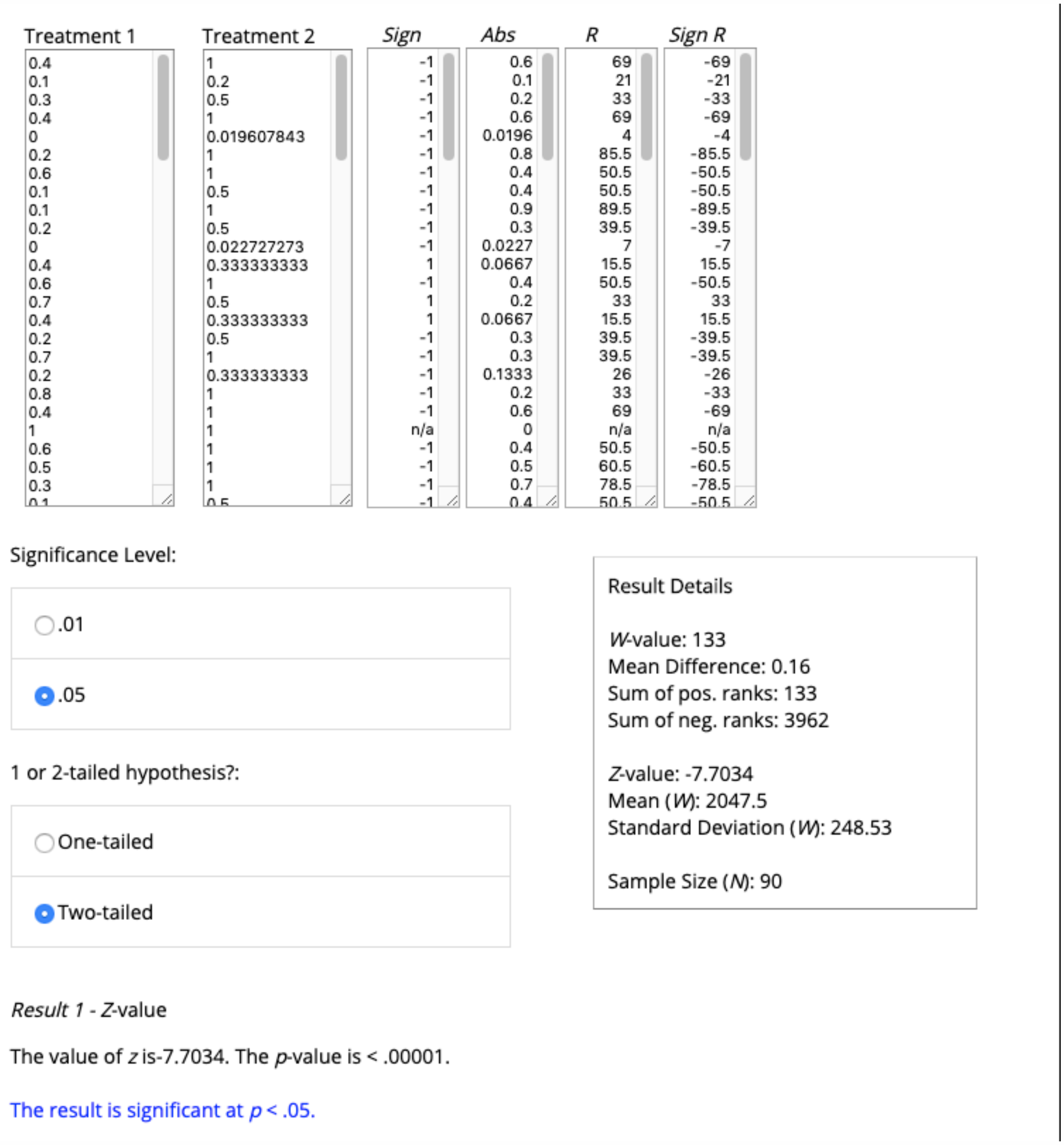
Result 1 - Z-value

The value of *z* is-0.4745. The *p*-value is .63836.

The result is *not* significant at *p* < .05.

The p-value is 0.63836. These two algorithm performs equally in P@10.

MRR:



Treatment 1	Treatment 2	*Sign*	*Abs*	*R*	*Sign R*
0.4	1	-1	0.6	69	-69
0.1	0.2	-1	0.1	21	-21
0.3	0.5	-1	0.2	33	-33
0.4	1	-1	0.6	69	-69
0	0.019607843	-1	0.0196	4	-4
0.2	1	-1	0.8	85.5	-85.5
0.6	1	-1	0.4	50.5	-50.5
0.1	0.5	-1	0.4	50.5	-50.5
0.1	1	-1	0.9	89.5	-89.5
0.2	0.5	-1	0.3	39.5	-39.5
0	0.022727273	-1	0.0227	7	-7
0.4	0.333333333	1	0.0667	15.5	15.5
0.6	1	-1	0.4	50.5	-50.5
0.7	0.5	1	0.2	33	33
0.4	0.333333333	1	0.0667	15.5	15.5
0.2	0.5	-1	0.3	39.5	-39.5
0.7	1	-1	0.3	39.5	-39.5
0.2	0.333333333	-1	0.1333	26	-26
0.8	1	-1	0.2	33	-33
0.4	1	-1	0.6	69	-69
1	1	n/a	0	n/a	n/a
0.6	1	-1	0.4	50.5	-50.5
0.5	1	-1	0.5	60.5	-60.5
0.3	1	-1	0.7	78.5	-78.5
0.1	0.5	-1	0.4	50.5	-50.5

Significance Level:

.01 [✓ .05]

1 or 2-tailed hypothesis?:

One-tailed [✓ Two-tailed]

Result Details

W-value: 133
Mean Difference: 0.16
Sum of pos. ranks: 133
Sum of neg. ranks: 3962

Z-value: -7.7034
Mean (*W*): 2047.5
Standard Deviation (*W*): 248.53

Sample Size (*N*): 90

Result 1 - Z-value

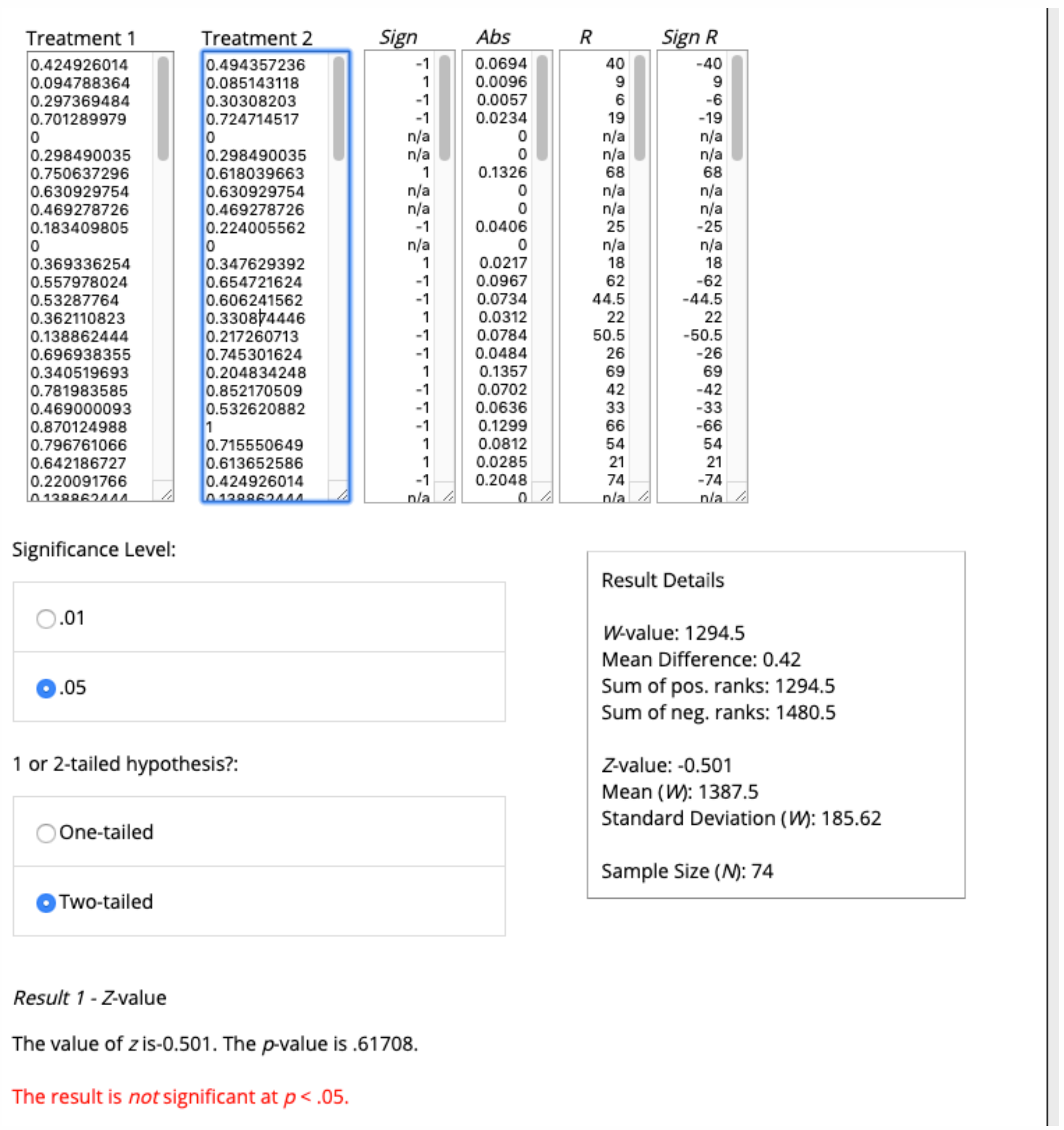
The value of *z* is-7.7034. The *p*-value is < .00001.

The result is significant at *p* < .05.

The one-tailed *p*-value is < .00001.

This result shows there is significant difference between these two ranking algorithm. The Two-tailed p value is also < 0.05, which means the TFIDF better than BM25 in RR.

NDCG:



Treatment 1	Treatment 2	*Sign*	*Abs*	*R*	*Sign R*
0.424926014	0.494357236	-1	0.0694	40	-40
0.094788364	0.085143118	1	0.0096	9	9
0.297369484	0.30308203	-1	0.0057	6	-6
0.701289979	0.724714517	-1	0.0234	19	-19
0	0	n/a	0	n/a	n/a
0.298490035	0.298490035	n/a	0	n/a	n/a
0.750637296	0.618039663	1	0.1326	68	68
0.630929754	0.630929754	n/a	0	n/a	n/a
0.469278726	0.469278726	n/a	0	n/a	n/a
0.183409805	0.224005562	-1	0.0406	25	-25
0	0	n/a	0	n/a	n/a
0.369336254	0.347629392	1	0.0217	18	18
0.557978024	0.654721624	-1	0.0967	62	-62
0.53287764	0.606241562	-1	0.0734	44.5	-44.5
0.362110823	0.330874446	1	0.0312	22	22
0.138862444	0.217260713	-1	0.0784	50.5	-50.5
0.696938355	0.745301624	-1	0.0484	26	-26
0.340519693	0.204834248	1	0.1357	69	69
0.781983585	0.852170509	-1	0.0702	42	-42
0.469000093	0.532620882	-1	0.0636	33	-33
0.870124988	1	-1	0.1299	66	-66
0.796761066	0.715550649	1	0.0812	54	54
0.642186727	0.613652586	1	0.0285	21	21
0.220091766	0.424926014	-1	0.2048	74	-74
0.138862444	0.138862444	n/a	0	n/a	n/a

Significance Level:

( ) .01

(●) .05

1 or 2-tailed hypothesis?:

( ) One-tailed

(●) Two-tailed

Result Details

W-value: 1294.5
Mean Difference: 0.42
Sum of pos. ranks: 1294.5
Sum of neg. ranks: 1480.5

Z-value: -0.501
Mean (*W*): 1387.5
Standard Deviation (*W*): 185.62

Sample Size (*N*): 74

Result 1 - Z-value

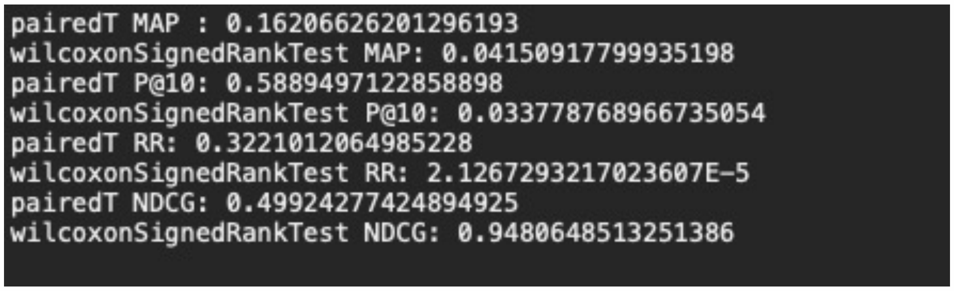
The value of *z* is-0.501. The *p*-value is .61708.

The result is *not* significant at *p* < .05.

The p-value is 0.30854. These two algorithm performs equally in NDCG.

I also test in Java:

```
import org.apache.commons.math3.stat.inference.TTest;
import org.apache.commons.math3.stat.inference.WilcoxonSignedRankTest;

TTest t = new TTest();
WilcoxonSignedRankTest w = new WilcoxonSignedRankTest();

    //double p = t.tTest(ev.ok_MAP, ev.tfidf_MAP);
      //System.out.println("t-Test: p=" + p);
    System.out.println("pairedT MAP : " + t.pairedTTest(ev.ok_MAP,
ev.tfidf_MAP));
    System.out.println("wilcoxonSignedRankTest MAP: " +
w.wilcoxonSignedRankTest(ev.ok_MAP, ev.tfidf_MAP, false));
    System.out.println("pairedT P@10: " + t.pairedTTest(ev.ok_PK, ev.tfidf_PK));
    System.out.println("wilcoxonSignedRankTest P@10: " +
w.wilcoxonSignedRankTest(ev.ok_PK, ev.tfidf_PK, false));
    System.out.println("pairedT RR: " + t.pairedTTest(ev.ok_RR, ev.tfidf_RR));
    System.out.println("wilcoxonSignedRankTest RR: " +
w.wilcoxonSignedRankTest(ev.ok_RR, ev.tfidf_RR, false));
    System.out.println("pairedT NDCG: " + t.pairedTTest(ev.ok_NDCG,
ev.tfidf_NDCG));
    System.out.println("wilcoxonSignedRankTest NDCG: " +
w.wilcoxonSignedRankTest(ev.ok_NDCG, ev.tfidf_NDCG, false));
```



```
pairedT MAP : 0.16206626201296193
wilcoxonSignedRankTest MAP: 0.04150917799935198
pairedT P@10: 0.5889497122858898
wilcoxonSignedRankTest P@10: 0.033778768966735054
pairedT RR: 0.3221012064985228
wilcoxonSignedRankTest RR: 2.1267293217023607E-5
pairedT NDCG: 0.49924277424894925
wilcoxonSignedRankTest NDCG: 0.9480648513251386
```

But this result seems not right.

Inclusion, the TFIDF

TFIDF might better than BM25 in MAP under Wilcoxon signed rank.

TFIDF might better than BM25 in RR under paired t-test and Wilcoxon signed rank.