# Spark-based Information Retrieval System

Haocheng li
Zongming Yang

## Boolean Retrieval System

Before doing the project, in order to improve performance, we did these:

1. Save all indexed results as object file instead of TXT file. So program can read index directly without changing format
2. Split index files into several pieces, which can reduce the time of generating index files.
3. Uploading java library com.cotdp.hadoop.ZipFileInputFormat to read zip files directly instead of uncompressing all zip files into XML files, which make system more effective.

```
val zipfiles = sc.newAPIHadoopFile(rawDataPath, classOf[ZipFileInputFormat],
classOf[Text], classOf[BytesWritable],new Job().getConfiguration())
val files = zipfiles.map(s => (s._1.toString, new String(s._2.getBytes)))
```

Contents

Stage 1

Stage 2

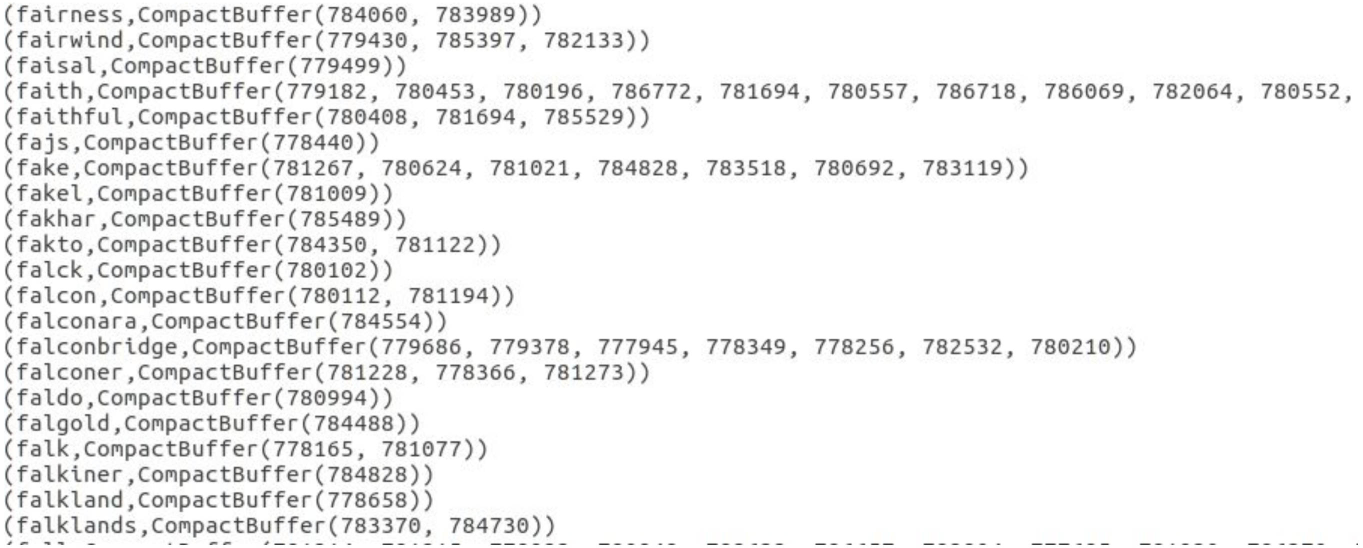
Stage 3

## Boolean Retrieval System

For Boolean retrieval, we need to create three indices: uniword index, biword index and uniword positional index.

Here are the content of three indices:

Uniword index



```
(fairness,CompactBuffer(784060, 783989))
(fairwind,CompactBuffer(779430, 785397, 782133))
(faisal,CompactBuffer(779499))
(faith,CompactBuffer(779182, 780453, 780196, 786772, 781694, 780557, 786718, 786069, 782064, 780552,
(faithful,CompactBuffer(780408, 781694, 785529))
(fajs,CompactBuffer(778440))
(fake,CompactBuffer(781267, 780624, 781021, 784828, 783518, 780692, 783119))
(fakel,CompactBuffer(781009))
(fakhar,CompactBuffer(785489))
(fakto,CompactBuffer(784350, 781122))
(falck,CompactBuffer(780102))
(falcon,CompactBuffer(780112, 781194))
(falconara,CompactBuffer(784554))
(falconbridge,CompactBuffer(779686, 779378, 777945, 778349, 778256, 782532, 780210))
(falconer,CompactBuffer(781228, 778366, 781273))
(faldo,CompactBuffer(780994))
(falgold,CompactBuffer(784488))
(falk,CompactBuffer(778165, 781077))
(falkiner,CompactBuffer(784828))
(falkland,CompactBuffer(778658))
(falklands,CompactBuffer(783370, 784730))
```

<word, txt1:txt2:txt3>, in files it shows like this:<word, CompactBuffer(txtid)>

Contents

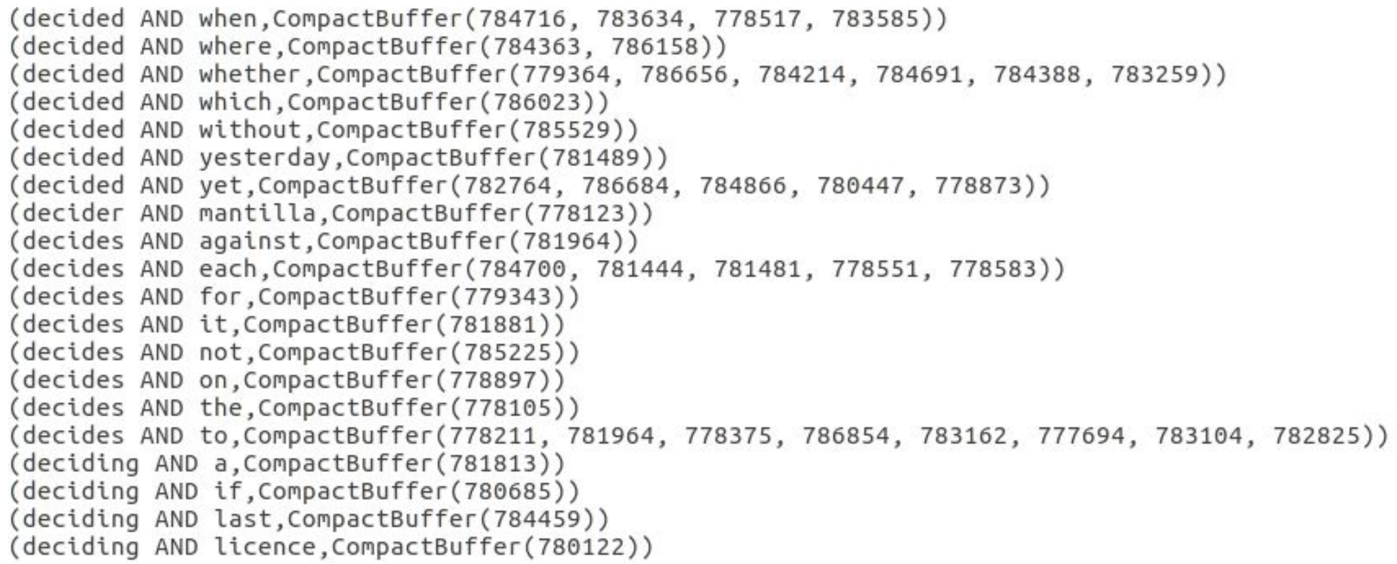
Stage 1

Stage 2

Stage 3

## Boolean Retrieval System

biword index



```
(decided AND when,CompactBuffer(784716, 783634, 778517, 783585))
(decided AND where,CompactBuffer(784363, 786158))
(decided AND whether,CompactBuffer(779364, 786656, 784214, 784691, 784388, 783259))
(decided AND which,CompactBuffer(786023))
(decided AND without,CompactBuffer(785529))
(decided AND yesterday,CompactBuffer(781489))
(decided AND yet,CompactBuffer(782764, 786684, 784866, 780447, 778873))
(decider AND mantilla,CompactBuffer(778123))
(decides AND against,CompactBuffer(781964))
(decides AND each,CompactBuffer(784700, 781444, 781481, 778551, 778583))
(decides AND for,CompactBuffer(779343))
(decides AND it,CompactBuffer(781881))
(decides AND not,CompactBuffer(785225))
(decides AND on,CompactBuffer(778897))
(decides AND the,CompactBuffer(778105))
(decides AND to,CompactBuffer(778211, 781964, 778375, 786854, 783162, 777694, 783104, 782825))
(deciding AND a,CompactBuffer(781813))
(deciding AND if,CompactBuffer(780685))
(deciding AND last,CompactBuffer(784459))
(deciding AND licence,CompactBuffer(780122))
```

<biword, txt1:txt2:txt3>, in files it shows like this: <word, CompactBuffer(txtid)>

Contents

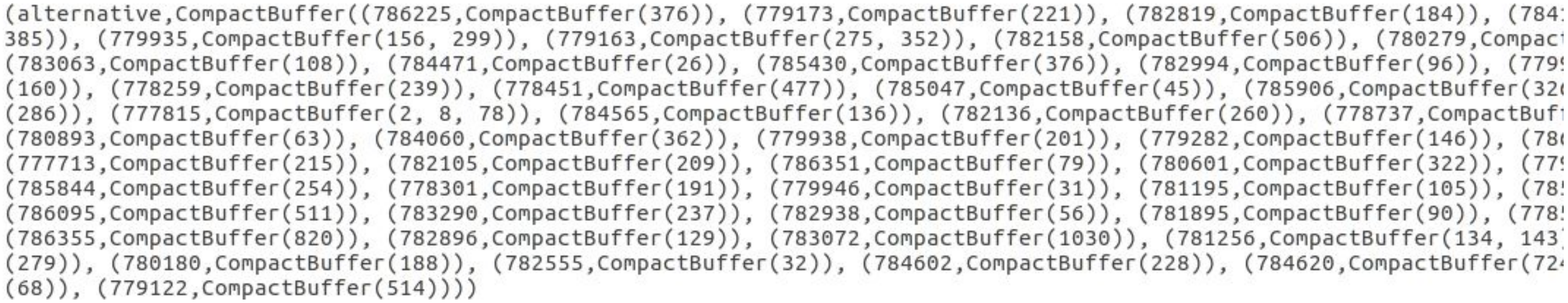
Stage 1

Stage 2

Stage 3

## Boolean Retrieval System

uniword positional index



```
(alternative,CompactBuffer((786225,CompactBuffer(376)), (779173,CompactBuffer(221)), (782819,CompactBuffer(184)), (784
385)), (779935,CompactBuffer(156, 299)), (779163,CompactBuffer(275, 352)), (782158,CompactBuffer(506)), (780279,Compac
(783063,CompactBuffer(108)), (784471,CompactBuffer(26)), (785430,CompactBuffer(376)), (782994,CompactBuffer(96)), (779
(160)), (778259,CompactBuffer(239)), (778451,CompactBuffer(477)), (785047,CompactBuffer(45)), (785906,CompactBuffer(32
(286)), (777815,CompactBuffer(2, 8, 78)), (784565,CompactBuffer(136)), (782136,CompactBuffer(260)), (778737,CompactBuf
(780893,CompactBuffer(63)), (784060,CompactBuffer(362)), (779938,CompactBuffer(201)), (779282,CompactBuffer(146)), (78
(777713,CompactBuffer(215)), (782105,CompactBuffer(209)), (786351,CompactBuffer(79)), (780601,CompactBuffer(322)), (77
(785844,CompactBuffer(254)), (778301,CompactBuffer(191)), (779946,CompactBuffer(31)), (781195,CompactBuffer(105)), (78
(786095,CompactBuffer(511)), (783290,CompactBuffer(237)), (782938,CompactBuffer(56)), (781895,CompactBuffer(90)), (778
(786355,CompactBuffer(820)), (782896,CompactBuffer(129)), (783072,CompactBuffer(1030)), (781256,CompactBuffer(134, 143
(279)), (780180,CompactBuffer(188)), (782555,CompactBuffer(32)), (784602,CompactBuffer(228)), (784620,CompactBuffer(72
(68)), (779122,CompactBuffer(514))))
```

<word <txt1 pos1:pos2:pos3>,<txt2 pos1:pos2:pos3>...>
in files it shows like this:<word, CompactBuffer<txtid, CompactBuffer(position)>>

Contents

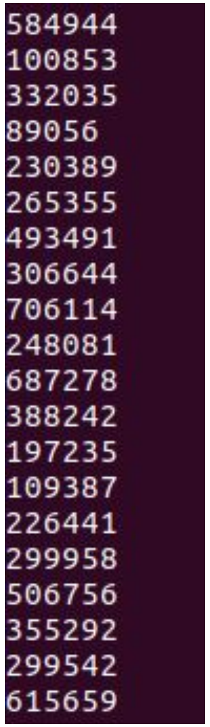
**Stage 1**

**Stage 2**

**Stage 3**

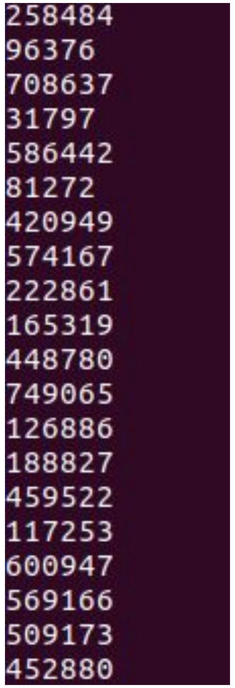
## Boolean Retrieval System

Here are the results of stage 1:



```
584944
100853
332035
89056
230389
265355
493491
306644
706114
248081
687278
388242
197235
109387
226441
299958
506756
355292
299542
615659
```

Result of query “volkswagen”



```
258484
96376
708637
31797
586442
81272
420949
574167
222861
165319
448780
749065
126886
188827
459522
117253
600947
569166
509173
452880
```

Result of query “mexico economy”

# Contents

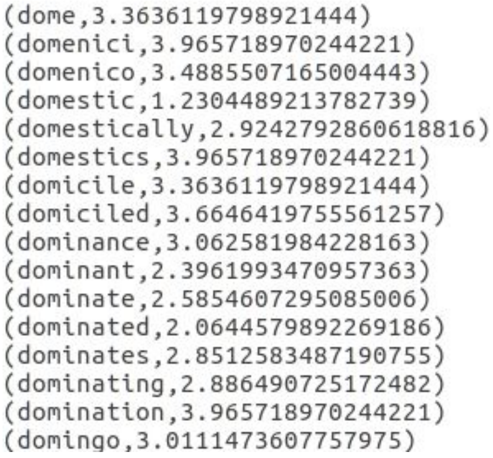
Stage 1

Stage 2

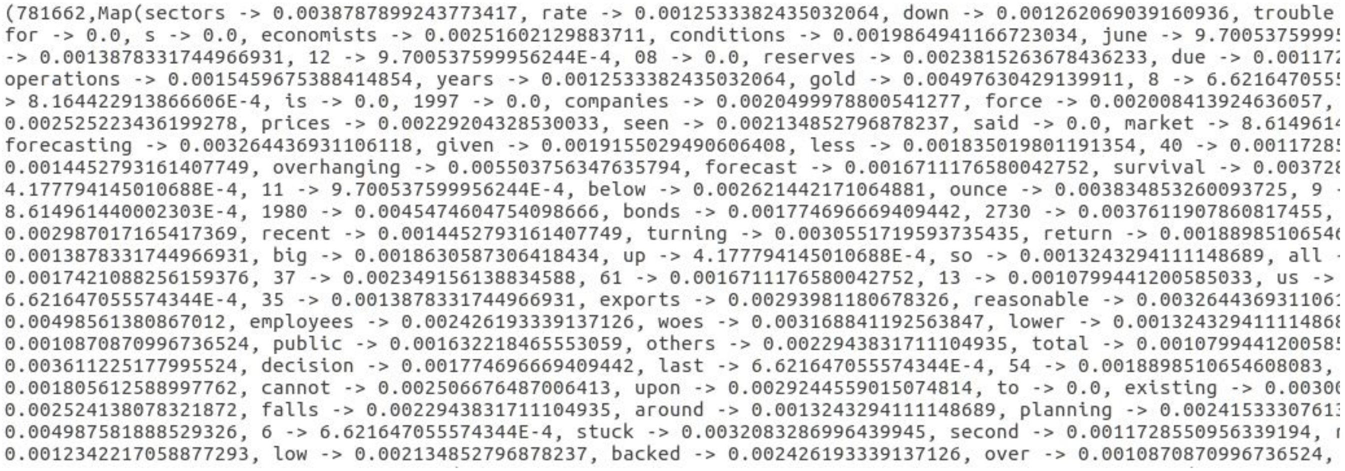
Stage 3

## Ranked Retrieval System

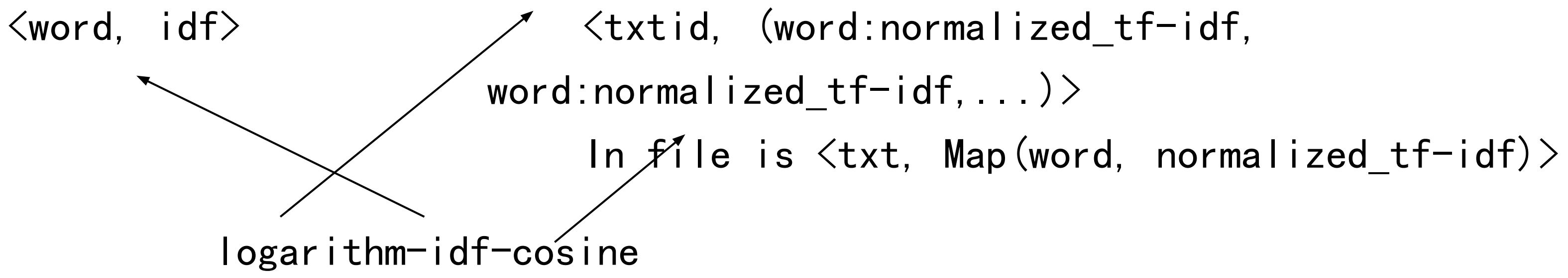
We use ltc (logarithm-idf-cosine).ltc for query and documents to compute their weight. To get tf-idf weights of terms in documents and normalize the document, we created two indices:



```
(dome,3.3636119798921444)
(domenici,3.965718970244221)
(domenico,3.4885507165004443)
(domestic,1.2304489213782739)
(domestically,2.9242792860618816)
(domestics,3.965718970244221)
(domicile,3.3636119798921444)
(domiciled,3.6646419755561257)
(dominance,3.062581984228163)
(dominant,2.3961993470957363)
(dominate,2.5854607295085006)
(dominated,2.0644579892269186)
(dominates,2.8512583487190755)
(dominating,2.886490725172482)
(domination,3.965718970244221)
(domingo,3.0111473607757975)
```



```
(781662,Map(sectors -> 0.0038787899243773417, rate -> 0.0012533382435032064, down -> 0.001262069039160936, trouble
for -> 0.0, s -> 0.0, economists -> 0.0025160212988371 1, conditions -> 0.0019864941166723034, june -> 9.7005375999
-> 0.0013878331744966931, 12 -> 9.700537599956244E-4, 08 -> 0.0, reserves -> 0.0023815263678436233, due -> 0.00117
operations -> 0.0015459675388414854, years -> 0.0012533382435032064, gold -> 0.004976304291399 11, 8 -> 6.621647055
> 8.164422913866606E-4, is -> 0.0, 1997 -> 0.0, companies -> 0.0020499978800541277, force -> 0.002008413924636057,
0.002525223436199278, prices -> 0.00229204328530033, seen -> 0.002134852796878237, said -> 0.0, market -> 8.614961
forecasting -> 0.003264436931106118, given -> 0.0019155029490606408, less -> 0.0018350198011913 54, 40 -> 0.0011728
0.0014452793161407749, overhanging -> 0.005503756347635794, forecast -> 0.0016711176580042752, survival -> 0.00372
4.177794145010688E-4, 11 -> 9.700537599956244E-4, below -> 0.002621442171064881, ounce -> 0.003834853260093725, 9
8.614961440002303E-4, 1980 -> 0.0045474604754098666, bonds -> 0.001774696669409442, 2730 -> 0.0037611907860817455,
0.002987017165417369, recent -> 0.0014452793161407749, turning -> 0.003055171959373543 5, return -> 0.0018898510654
0.0013878331744966931, big -> 0.0018630587306418434, up -> 4.177794145010688E-4, so -> 0.0013243294111148689, all
0.0017421088256159376, 37 -> 0.002349156138834588, 61 -> 0.0016711176580042752, 13 -> 0.0010799441200585033, us ->
6.621647055574344E-4, 35 -> 0.0013878331744966931, exports -> 0.00293981180678326, reasonable -> 0.003264436931106
0.00498561380867012, employees -> 0.002426193339137126, woes -> 0.003168841192563847, lower -> 0.001324329411114868
0.0010870870996736524, public -> 0.001632218465553059, others -> 0.0022943831711104935, total -> 0.001079944120058
0.003611225177995524, decision -> 0.001774696669409442, last -> 6.621647055574344E-4, 54 -> 0.0018898510654608083,
0.001805612588997762, cannot -> 0.002506676487006413, upon -> 0.0029244559015074814, to -> 0.0, existing -> 0.0030
0.002524138078321872, falls -> 0.0022943831711104935, around -> 0.0013243294111148689, planning -> 0.0024153330761
0.004987581888529326, 6 -> 6.621647055574344E-4, stuck -> 0.0032083286996439945, second -> 0.0011728550956339194,
0.0012342217058877293, low -> 0.002134852796878237, backed -> 0.002426193339137126, over -> 0.0010870870996736524,
```



<word, idf>

<txtid, (word:normalized_tf-idf, word:normalized_tf-idf,...)>

In file is <txt, Map(word, normalized_tf-idf)>

logarithm-idf-cosine

Contents

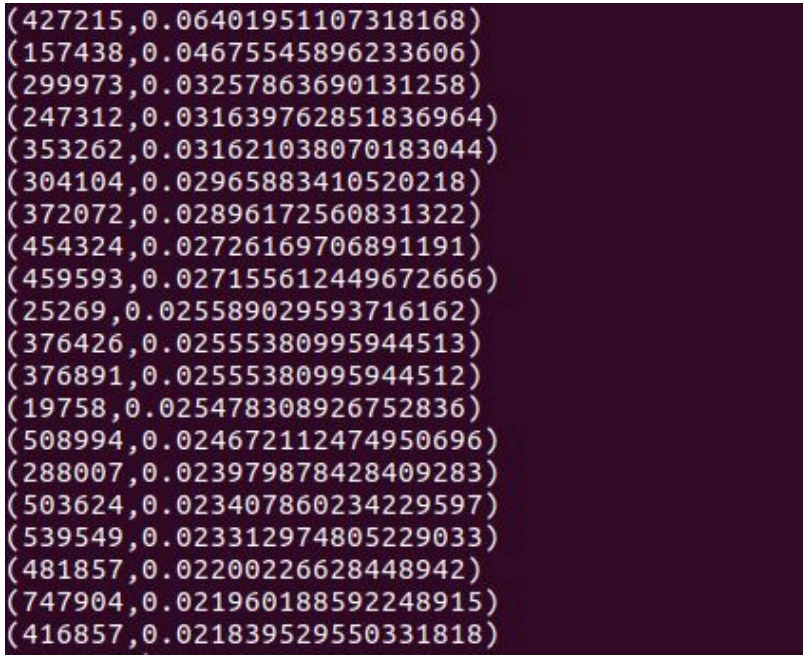
Stage 1

Stage 2

Stage 3

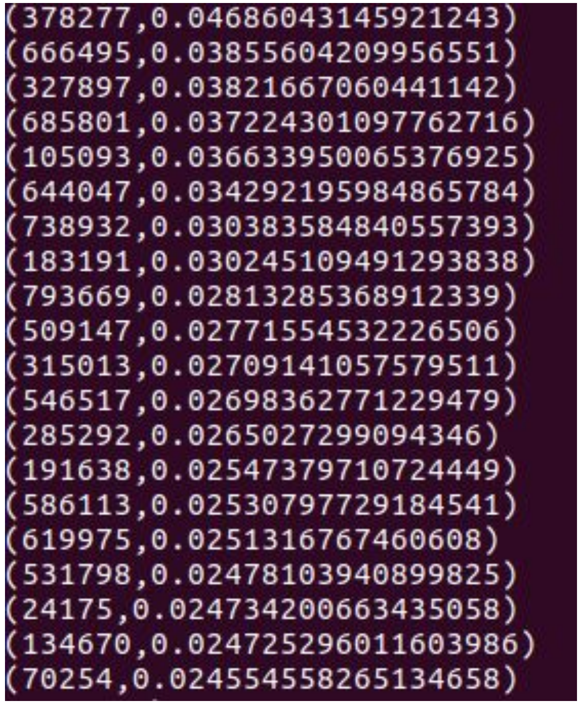
## Ranked Retrieval System

Here are the results of stage 2:



```
(427215,0.06401951107318168)
```

Result of query “volkswagen”



```
(378277,0.04686043145921243)
(666495,0.03855604209956551)
(327897,0.03821667060441142)
(685801,0.037224301097762716)
(105093,0.036633950065376925)
(644047,0.034292195984865784)
(738932,0.030383584840557393)
(183191,0.030245109491293838)
(793669,0.02813285368912339)
(509147,0.02771554532226506)
(315013,0.02709141057579511)
(546517,0.02698362771229479)
(285292,0.0265027299094346)
(191638,0.02547379710724449)
(586113,0.02530797729184541)
(619975,0.0251316767460608)
(531798,0.02478103940899825)
(24175,0.02473420066435058)
(134670,0.024725296011603986)
(70254,0.024554558265134658)
```

Result of query “mexico economy”

# Contents

Stage 1

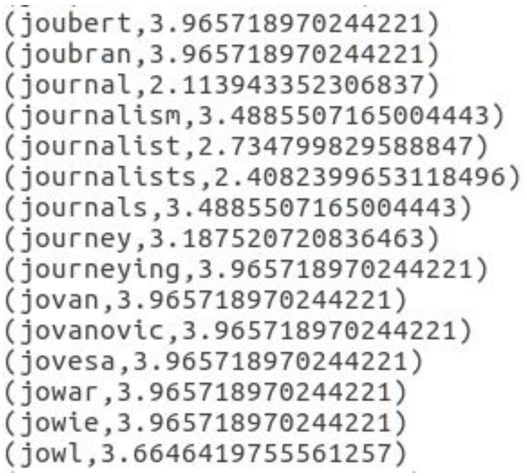
Stage 2

Stage 3

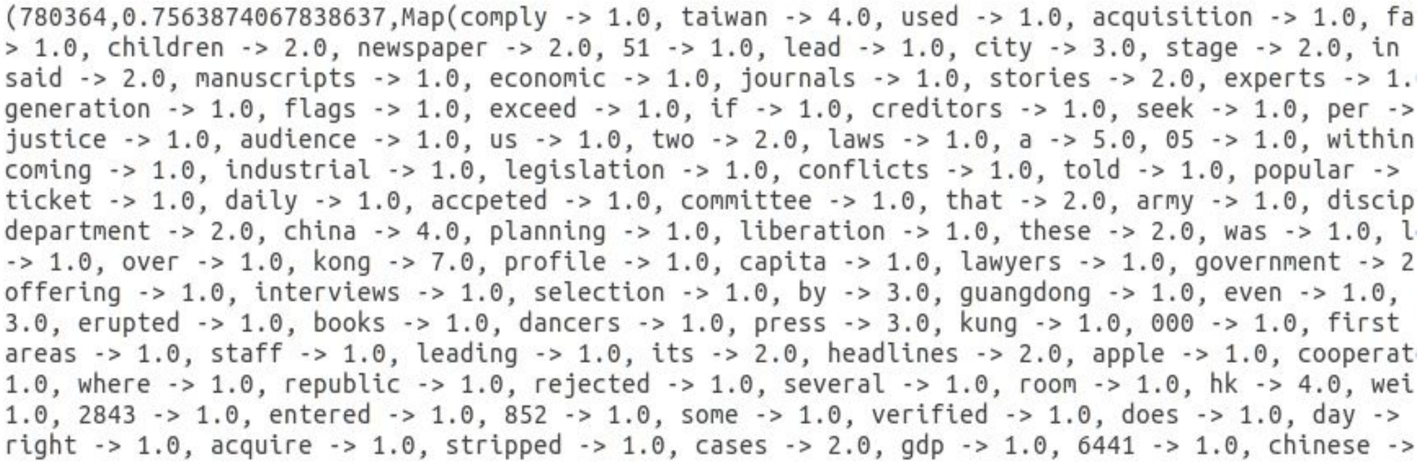
## BM25 Retrieval System

$$RSV_d = \sum_{t \in q} \left[ \log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{ave})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

We use formula mentioned above to calculate $RSV_d$ , here we set $k_1$=1.5, $k_3$=1.5 and b=0.75. To get result, we need to know idf, $\mathrm{tf}_{td}$, $\mathrm{tf}_{tq}$ of each term, also length of each document and average document length. Thus, we created two indices:



```
(joubert,3.965718970244221)
(joubran,3.965718970244221)
(journal,2.113943352306837)
(journalism,3.4885507165004443)
(journalist,2.734799829588847)
(journalists,2.4082399653118496)
(journals,3.4885507165004443)
(journey,3.18752072083646З)
(journeying,3.965718970244221)
(jovan,3.965718970244221)
(jovanovic,3.965718970244221)
(jovesa,3.965718970244221)
(jowar,3.965718970244221)
(jowie,3.965718970244221)
(jowl,3.6646419755561257)
```



```
(780364,0.7563874067838637,Map(comply -> 1.0, taiwan -> 4.0, used -> 1.0, acquisition -> 1.0, fa
> 1.0, children -> 2.0, newspaper -> 2.0, 51 -> 1.0, lead -> 1.0, city -> 3.0, stage -> 2.0, in
said -> 2.0, manuscripts -> 1.0, economic -> 1.0, journals -> 1.0, stories -> 2.0, experts -> 1.
generation -> 1.0, flags -> 1.0, exceed -> 1.0, if -> 1.0, creditors -> 1.0, seek -> 1.0, per ->
justice -> 1.0, audience -> 1.0, us -> 1.0, two -> 2.0, laws -> 1.0, a -> 5.0, 05 -> 1.0, within
coming -> 1.0, industrial -> 1.0, legislation -> 1.0, conflicts -> 1.0, told -> 1.0, popular ->
ticket -> 1.0, daily -> 1.0, accpeted -> 1.0, committee -> 1.0, that -> 2.0, army -> 1.0, discip
department -> 2.0, china -> 4.0, planning -> 1.0, liberation -> 1.0, these -> 2.0, was -> 1.0, l
-> 1.0, over -> 1.0, kong -> 7.0, profile -> 1.0, capita -> 1.0, lawyers -> 1.0, government -> 2
offering -> 1.0, interviews -> 1.0, selection -> 1.0, by -> 3.0, guangdong -> 1.0, even -> 1.0,
3.0, erupted -> 1.0, books -> 1.0, dancers -> 1.0, press -> 3.0, kung -> 1.0, 000 -> 1.0, first
areas -> 1.0, staff -> 1.0, leading -> 1.0, its -> 2.0, headlines -> 2.0, apple -> 1.0, cooperat
1.0, where -> 1.0, republic -> 1.0, rejected -> 1.0, several -> 1.0, room -> 1.0, hk -> 4.0, wei
1.0, 2843 -> 1.0, entered -> 1.0, 852 -> 1.0, some -> 1.0, verified -> 1.0, does -> 1.0, day ->
right -> 1.0, acquire -> 1.0, stripped -> 1.0, cases -> 2.0, gdp -> 1.0, 6441 -> 1.0, chinese ->
```

<word, idf>

Similar with stage

<txt, length_Ratio, Map(word,tf)>

Length_Ratio=$L_d$ / $L_{avg}$

Contents

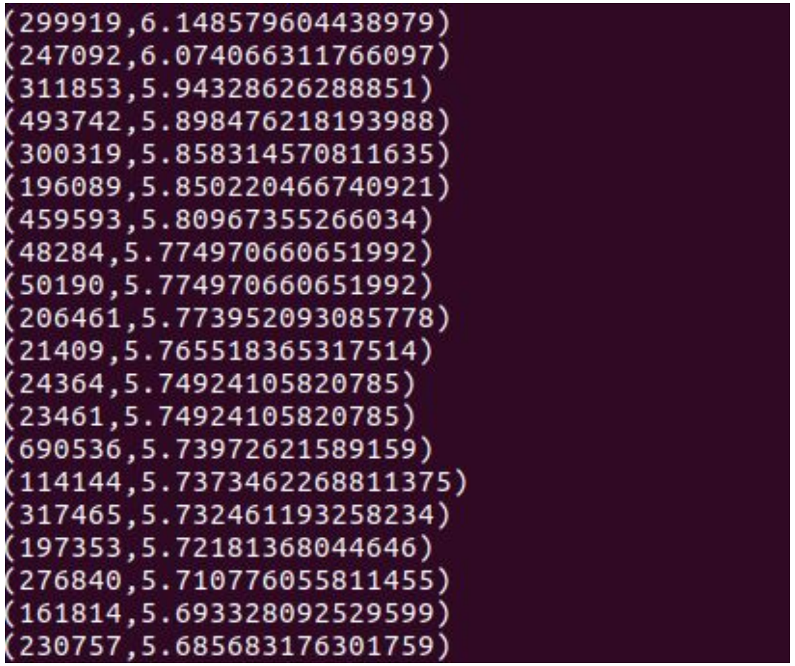
Stage 1

Stage 2

Stage 3

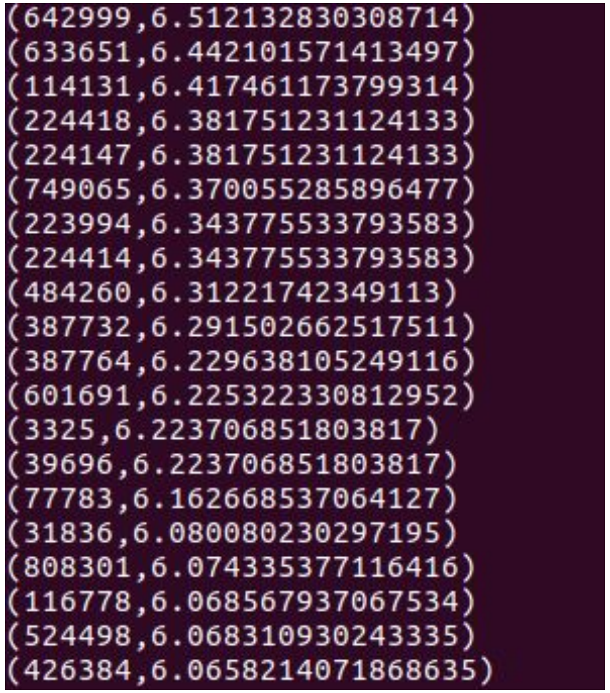
## BM25 Retrieval System

Here are the results of stage 3:



```
(299919,6.148579604438979)
(247092,6.074066311766097)
(311853,5.94328626288851)
(493742,5.898476218193988)
(300319,5.858314570811635)
(196089,5.850220466740921)
(459593,5.80967355266034)
(48284,5.774970660651992)
(50190,5.774970660651992)
(206461,5.773952093085778)
(21409,5.765518365317514)
(24364,5.74924105820785)
(23461,5.74924105820785)
(690536,5.73972621589159)
(114144,5.7373462268811375)
(317465,5.732461193258234)
(197353,5.72181368044646)
(276840,5.710776055811455)
(161814,5.693328092529599)
(230757,5.685683176301759)
```

Result of query “volkswagen”



```
(642999,6.512132830308714)
(633651,6.442101571413497)
(114131,6.417461173799314)
(224418,6.381751231124133)
(224147,6.381751231124133)
(749065,6.370055285896477)
(223994,6.343775533793583)
(224414,6.343775533793583)
(484260,6.31221742349113)
(387732,6.291502662517511)
(387764,6.229638105249116)
(601691,6.225322330812952)
(3325,6.223706851803817)
(39696,6.223706851803817)
(77783,6.162668537064127)
(31836,6.080080230297195)
(808301,6.074335377116416)
(116778,6.068567937067534)
(524498,6.068310930243335)
(426384,6.0658214071868635)
```

Result of query “mexico economy”

Contents

Stage 1

Stage 2

Stage 3

## Comparison between three stages

For query "volkswagen"

| | Uniword | Ranked | BM25 |
|---|---|---|---|
| Precision | 0.3 | 0.75 | 0.90 |

For query "mexico economy"

| | Uniword | Ranked | BM25 |
|---|---|---|---|
| Precision | 0.35 | 0.75 | 0.85 |

The precision of result of uniword retrieval system is really low compared to the other two method.
Also, the relevance between query and results of uniword is much less than the other two method because it is boolean retrieval.

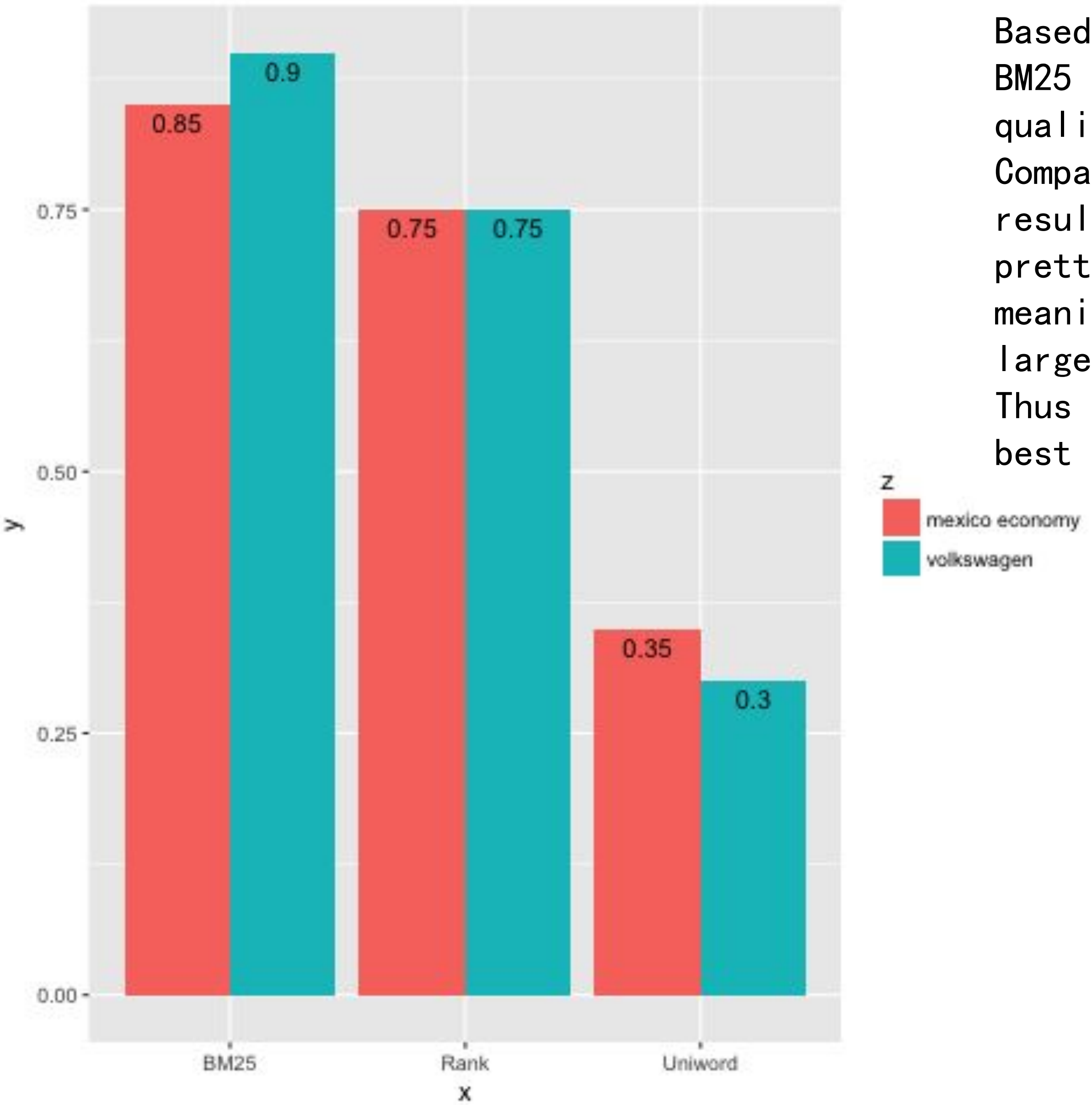
# Contents

Stage 1

Stage 2

Stage 3

## Comparison between three stages



Based on the manual tests, the results of BM25 have highest precision and best quality.

Compared with BM25, the length of top results of ranked retrieval system is pretty short so the results are kind meaningless. Because article’s length has large influence on ranked weight.

Thus we draw conclusion that BM25 is the best in these three method.

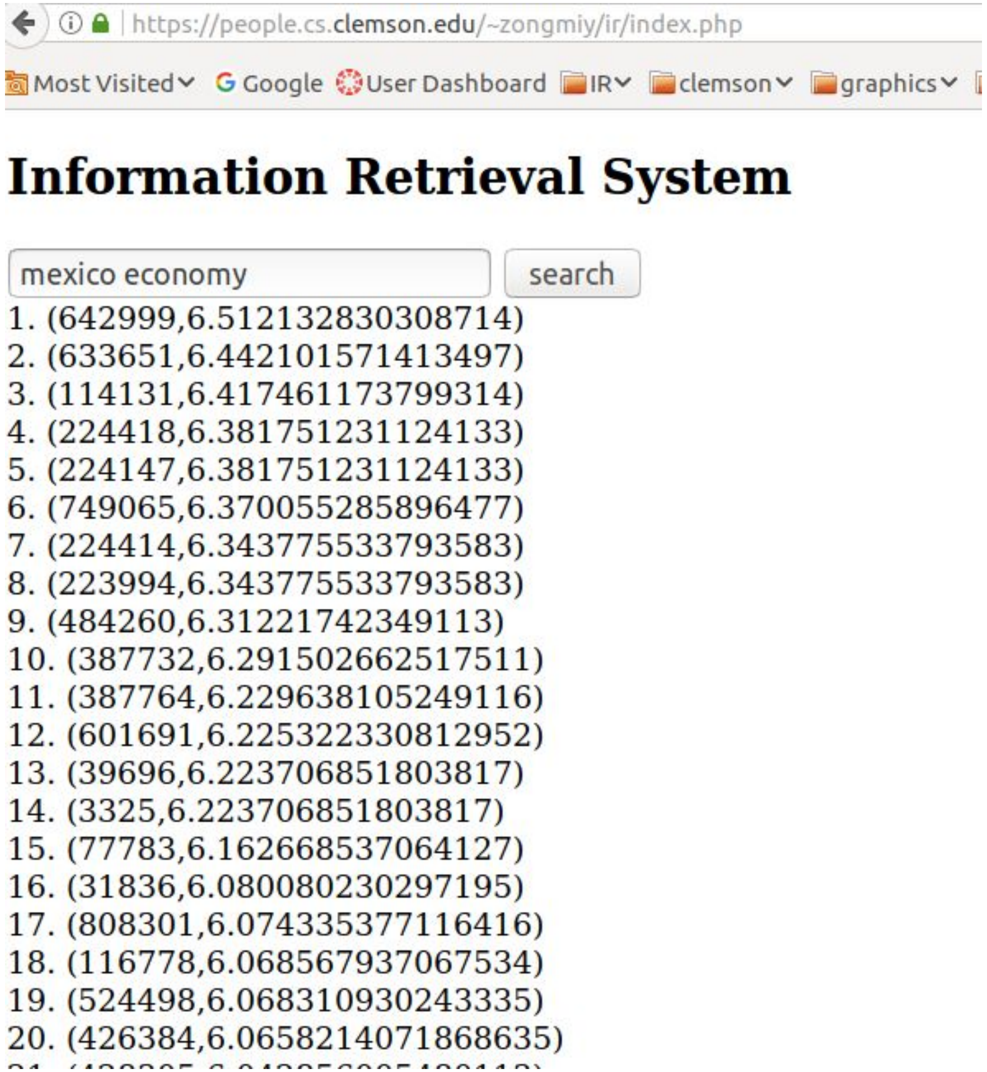
Contents

Stage 1

Stage 2

Stage 3

# Retrieval System Interface



## Information Retrieval System

mexico economy

```
1. (642999,6.512132830308714)
2. (633651,6.442101571413497)
3. (114131,6.417461173799314)
4. (224418,6.381751231124133)
5. (224147,6.381751231124133)
6. (749065,6.370055285896477)
7. (224414,6.343775533793583)
8. (223994,6.343775533793583)
9. (484260,6.31221742349113)
10. (387732,6.291502662517511)
11. (387764,6.229638105249116)
12. (601691,6.225322330812952)
13. (39696,6.223706851803817)
14. (3325,6.223706851803817)
15. (77783,6.162668537064127)
16. (31836,6.080080230297195)
17. (808301,6.074335377116416)
18. (116778,6.068567937067534)
19. (524498,6.068310930243335)
20. (426384,6.0658214071868635)
```

# Thanks!

# Question?