

► **Lucene and LSI**

Natural Language Processing and
Information Retrieval



► Summary

- Indexing with Lucene
- Queries with Lucene
- LSI: pros and cons
- Conclusion



Lucene

Indexing with Lucene

► How does Lucene work?

- Word Break rules from the Unicode Text Segmentation algorithm:
<http://unicode.org/reports/tr29/>
- TF-IDF

```
Why this movie?
0.10779239 = weight(content:alien in 11) [BM25Similarity], result of:
  0.10779239 = score(freq=56.0), computed as boost * idf * tf from:
    0.10863384 = idf, computed as log(1 + (N - n + 0.5) / (n + 0.5)) from:
      30 = n, number of documents containing term
      33 = N, total number of documents with field
    0.9922543 = tf, computed as freq / (freq + k1 * (1 - b + b * dl / avgdl)) from:
      56.0 = freq, occurrences of term within document
      1.2 = k1, term saturation parameter
      0.75 = b, length normalization parameter
      18456.0 = dl, length of field (approximate)
      121116.484 = avgdl, average length of field
```

► How to execute my code?

I used an IDE: IntelliJ. I added three Lucene libraries: lucene-core-8.5.1.jar, lucene-analyzers-common-8.5.1.jar, lucene-queryparser-8.5.1.jar. Using an IDE is easier. Or you should update CLASSPATH with the libraires. Then, you can run java file with option:

- First option can be '-v' for activate verbose to have explanation about score.
- Or, first option can be anything else for activating indexing
- Or you can use both option ('-v' and anything else) to activate verbose and indexing.
- If there is not parameter it will just request for a query and then do the search normally.

Performance

Assignment 2 (with python)

```
maeva@maeva-VirtualBox:~/Documents/NaturalLanguage/Project_IR_Evaluation$ python3 sortFiles.py  
113.1 seconds for indexing.  
maeva@maeva-VirtualBox:~/Documents/NaturalLanguage/Project IR Evaluation$
```

With Lucene:

```
If you want to index files, just add an argument (for example '1') when executing the java file.  
Process to indexing... may takes up to 1 min  
Time for indexing: 51 s.
```



Lucene

Queries with Lucene

► **Queries example**

- Donkey
- Dead
- May the force be with you
- Mars VS planet Mars



► **Expected results**

- Shrek
- Horror movies
- Star Wars 2
- The martian and Watchmen

► Performance

Assignment 2 (with python)

```
maeva@maeva-VirtualBox:~/Documents/NaturalLanguage/Project_IR_Evaluation$ python  
3 query.py  
Query (at least one word): yoko  
  
Time required for this query: 481 ms  
  
*****  
| Rank | Genre | Movie title | Score |  
-----  
| 1 | Horror | The Grudge | 48.0 |  
-----
```

With Lucene

```
Enter a query and tap Enter (just tap Enter to exit)  
yoko  
1. TheGrudge (Horror) score = 2.214093  
2. TheHauntingOfHillHouse (Horror) score = 2.2055194  
3. TheMummy (Horror) score = 2.1970122  
Time for query :4 ms.
```


« donkey »

Enter a query and tap Enter (just tap Enter to exit)

donkey

```
1. Shrek (Family)          score = 0.99718744
2. Ted (Family)            score = 0.99661344
3. TheBrothersBloom (Family) score = 0.99623114
4. TheIncredibles (Family) score = 0.9958492
5. TheMask (Family)        score = 0.9954675
6. TheProposal (Family)    score = 0.99470496
7. JurassicPark2 (Scifi)   score = 0.4648893
8. MenInBlack (Scifi)      score = 0.44000137
9. Prometheus (Scifi)      score = 0.39744663
10. StarWars2 (Scifi)      score = 0.3791136
Time for query :5 ms.
```

- Expected result: Shrek
- P&R:

$$P = \frac{1}{10} = 0.1$$

$$R = \frac{1}{1} = 1$$

$$F = \frac{2}{\frac{1}{1} + \frac{10}{1}} = 0.18$$

Old: $F = 0.67$

« dead »

```
Enter a query and tap Enter (just tap Enter to exit)
dead
1. FridayThe13th (Horror)      score = 0.014706891
2. Halloween (Horror)         score = 0.0147001175
3. EvilDead2 (Horror)          score = 0.014696173
4. Hannibal (Horror)           score = 0.014691505
5. Insidious (Horror)          score = 0.014686035
6. TheMummy (Horror)           score = 0.014685125
7. TheGrudge (Horror)          score = 0.014682103
8. It (Horror)                 score = 0.014679245
9. TheHauntingOfHillHouse (Horror) score = 0.014674117
10. EvilDead (Horror)          score = 0.014635224
Time for query :34 ms.
```

- Expected result: all horror movies
- P&R:

$$P = \frac{10}{10} = 1$$

$$R = \frac{10}{11} = 0.91$$

$$F = \frac{2}{\frac{10}{10} + \frac{11}{10}} = 0.95$$

Old: $F = 0.5$

«may the force be with you »

Enter a query and tap Enter (just tap Enter to exit)

may the force be with you

1. EvilDead2 (Horror)	score = 0.1801891
2. FridayThe13th (Horror)	score = 0.1799631
3. Insidious (Horror)	score = 0.17978473
4. Hannibal (Horror)	score = 0.17975242
5. Halloween (Horror)	score = 0.17947873
6. It (Horror)	score = 0.17944859
7. TheGrudge (Horror)	score = 0.17924237
8. TheHauntingOfHillHouse (Horror)	score = 0.1788467
9. TheMummy (Horror)	score = 0.1786535
10. EvilDead (Horror)	score = 0.1782693

Time for query :79 ms.

Enter a query and tap Enter (just tap Enter to exit)

"may the force be with you"

1. StarWars2 (Scifi)	score = 0.1004063
2. TheMarsian (Scifi)	score = 0.09410986
3. Watchmen (Scifi)	score = 0.09124878

Time for query :29 ms.

- Expected result: Star Wars 2. We look at the second case.
- P&R:

$$P = \frac{1}{3} = 0.33$$

$$R = \frac{1}{1} = 1$$

$$F = \frac{2}{\frac{3}{1} + \frac{1}{1}} = 0.5$$

Old: $F = 0.25$

« Mars » VS « Planet Mars »

```
Enter a query and tap Enter (just tap Enter to exit)
Mars
1. Watchmen (Scifi)      score = 1.0642593
2. TheMarsian (Scifi)    score = 1.0620387
3. 2001ASpaceOdyssey (Scifi) score = 1.002471
4. Cube (Scifi)         score = 0.99130994
5. EternalSunshineOfTheSpotlessMind (Scifi) score = 0.967084
6. Ghostbuster2 (Scifi)  score = 0.93656653
7. HotTubeTimeMachine (Scifi) score = 0.9172695
8. Prometheus (Scifi)    score = 0.91116047
9. StarWars2 (Scifi)     score = 0.89965236
10. JurassicPark2 (Scifi) score = 0.88096666
Time for query :4 ms.
```

```
Enter a query and tap Enter (just tap Enter to exit)
Planet Mars
1. Watchmen (Scifi)      score = 1.107897
2. TheMarsian (Scifi)    score = 1.1057162
3. 2001ASpaceOdyssey (Scifi) score = 1.0434271
4. Cube (Scifi)         score = 1.031708
5. EternalSunshineOfTheSpotlessMind (Scifi) score = 1.0062804
6. Ghostbuster2 (Scifi)  score = 0.97426766
7. HotTubeTimeMachine (Scifi) score = 0.95403564
8. Prometheus (Scifi)    score = 0.9538797
9. StarWars2 (Scifi)     score = 0.94315207
10. JurassicPark2 (Scifi) score = 0.9159953
Time for query :5 ms.
```

- Expected result: Watchmen and The Marsian

- P&R:

Mars

$$P = \frac{2}{10} = 0.2, R = \frac{2}{2} = 1 \text{ so } F = \frac{\frac{2}{10} + \frac{2}{2}}{2} = 0.33$$

Old: $F = 0.67$

Planet Mars

$$P = \frac{2}{10} = 0.2, R = \frac{2}{2} = 1 \text{ so } F = \frac{\frac{2}{10} + \frac{2}{2}}{2} = 0.33$$

Old: $F = 0.44$

This time, same results for both requests!



**What is it? How does
it work?**

► What is it?

- Indexing and IR method that uses SVD.
- Identify patterns between terms and concepts
- Query about concepts not only words

► Pros and cons

► Pros

- Increases recall
- Independent from words

► Cons

- Costly
- Choosing accurate number of dimensions

► **Difficulties**

- Using a new library in 5 days.
- Some docs were old and unusable.
- Understand principles and how it works.
- Exploiting results

► **Conclusion**

- Lucene is very powerful.
- LSI seems to be the best alternative for query system.
- But it's heavy to include this in a system for beginners.

► Sources

- https://www.tutorialspoint.com/lucene/lucene_adddocument.htm
- <https://stackoverflow.com/questions/5694385/getting-the-filenames-of-all-files-in-a-folder>
- <https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html>
- https://www.w3schools.com/java/java_user_input.asp
- <https://examples.javacodegeeks.com/core-java/apache/lucene/lucene-query-parser-example/>
- https://en.wikipedia.org/wiki/Latent_semantic_analysis
- And mostly lucene docs included in the zip files

Conclusion
Thank you

Thank you !