Valentin Gegoux

# LSI model report

## Introduction

Latent Semantic Indexing (LSI) model has been applied to 1000 song lyrics, with ranking of results using cosine similarity.

Github repository: https://github.com/Valegox/IR-System

## How to use

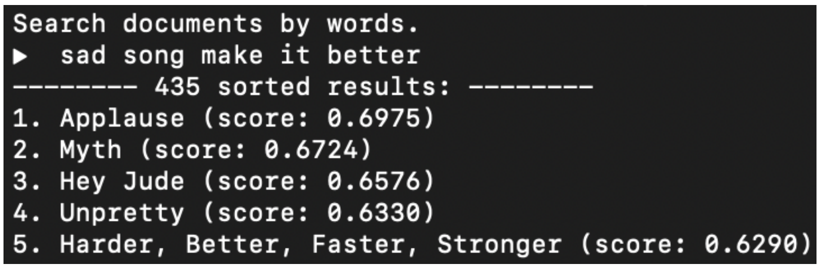
Please refer to the readme of the Github repository to see how to use the program.

## Experimentations

The K rank approximation has an impact on results: the lower the value, the more general the model is. The higher the value, the more precise it is. It should not be either too low or too high.

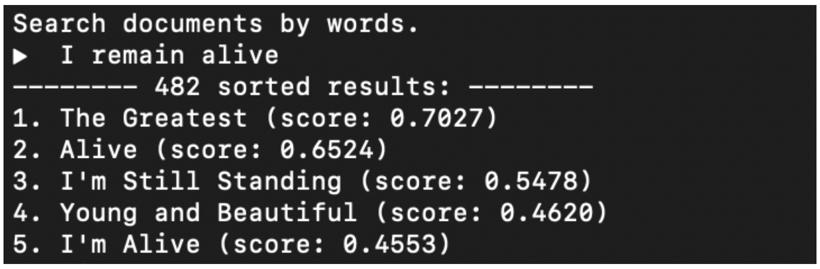
Across multiple test queries, we can consistently observe a high rate of **false positives**. For all queries, many documents are ranked highly despite bearing little real relevance to the query intent.

For example, the query *"sad song make it better"* intends to retrieve the "*Hey Jude*" song written by the Beatles. It's the 3rd result among a 1000 ones, which is not bad. But unrelated songs ("*Applause*", "*Myth*"...) blur the result.


```
Search documents by words.
▶  sad song make it better
-------- 435 sorted results: --------
1. Applause (score: 0.6975)
2. Myth (score: 0.6724)
3. Hey Jude (score: 0.6576)
4. Unpretty (score: 0.6330)
5. Harder, Better, Faster, Stronger (score: 0.6290)
```

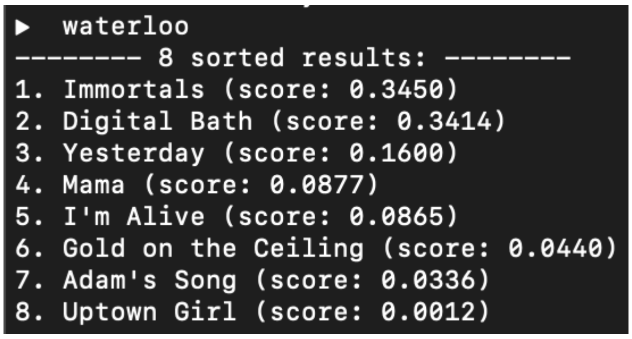
However, the goal of the LSI model is to find the meaning behind words in a collection of texts. Instead of just matching exact words, it looks at patterns of how words appear together and builds a map of related meanings.

The query “*I remain alive*” intends to retrieve “*I’m Still Standing”* by Elton John. The query has a similar meaning to the song. We can see it works: It’s (again) the 3rd result among 1000 songs. Note that the words “remain” and “alive” are not included in the lyrics of the song. So despite too many false positives, this model shows interesting results.



```
Search documents by words.
▶  I remain alive
-------- 482 sorted results: --------
1. The Greatest (score: 0.7027)
2. Alive (score: 0.6524)
3. I'm Still Standing (score: 0.5478)
4. Young and Beautiful (score: 0.4620)
5. I'm Alive (score: 0.4553)
```

Some obvious requests, such as “*Waterloo”* , show very low performances as the song “*Waterloo*” by ABBA doesn’t appear in the top results (despite the fact this word is present many times in the document):



```
▶  waterloo
-------- 8 sorted results: --------
1. Immortals (score: 0.3450)
2. Digital Bath (score: 0.3414)
3. Yesterday (score: 0.1600)
4. Mama (score: 0.0877)
5. I'm Alive (score: 0.0865)
6. Gold on the Ceiling (score: 0.0440)
7. Adam's Song (score: 0.0336)
8. Uptown Girl (score: 0.0012)
```

## Difference with other models and conclusion

The LSI model shows a lot more incorrect and false-positive results than previously seen models. It can’t be used alone. However, it seems to capture the meaning of some queries (for example - “*I remain alive*” retrieves “*I’m Still Standing*”).

A solution could be to combine LSI with Vector Space Models to build a both viable and meaningful IR system.