# Information Retrieval

## Search Engine Bandits

#### Sara Asaad – 207674318

#### Ward Zidani – 318531944

#### Safwan Halabi – 213784127

[GitHub Link](https://github.com/Safwan-Halabi/Information-Retrieval.git): - <https://github.com/Safwan-Halabi/Information-Retrieval.git>

**Question 1.1:**

|D| = 100, Rel = 20, Non-Rel = 80, the system retrieved 8 R and 10 NR.

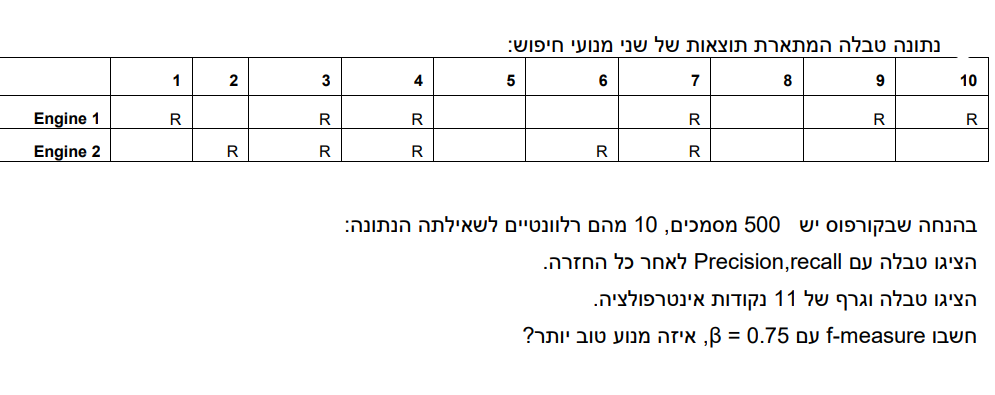
*Precision* – the percentage of **Relevant** documents from the overall **retrieved** documents.

*Recall* – the percentage of **Relevant** documents from the overall **Relevant** documents.

Thus, making the **precision** of the system in question: 8 / (8 + 10) = 8 / 18 =~ 0.444.

And the **recall** of the system in question: 8 / 20 = 0.4.

**Question 1.2:**



**Answer**:

|  |  |  |
| --- | --- | --- |
| Engine 1 | Precision | Recall |
| Document 1 | 1 / 1 = 1 | 1 / 10 = 0.1 |
| Document 2 | 1 / 2 = 0.5 | 1 / 10 = 0.1 |
| Document 3 | 2 / 3 = 0.667 | 2 / 10 = 0.2 |
| Document 4 | 3 / 4 = 0.75 | 3 / 10 = 0.3 |
| Document 5 | 3 / 5 = 0.6 | 3 / 10 = 0.3 |
| Document 6 | 3 / 6 = 0.5 | 3 / 10 = 0.3 |
| Document 7 | 4 / 7 =~ 0.571 | 4 / 10 = 0.4 |
| Document 8 | 4 / 8 = 0.5 | 4 / 10 = 0.4 |
| Document 9 | 5 / 9 =~ 0.555 | 5 / 10 = 0.5 |
| Document 10 | 6 / 10 = 0.6 | 6 / 10 = 0.6 |

|  |  |  |
| --- | --- | --- |
| Engine 2 | Precision | Recall |
| Document 1 | 0 / 1 = 0 | 0 / 10 = 0 |
| Document 2 | 1 / 2 = 0.5 | 1 / 10 = 0.1 |
| Document 3 | 2 / 3 = 0.667 | 2 / 10 = 0.2 |
| Document 4 | 3 / 4 = 0.75 | 3 / 10 = 0.3 |
| Document 5 | 3 / 5 = 0.6 | 3 / 10 = 0.3 |
| Document 6 | 4 / 6 = 0.667 | 4 / 10 = 0.4 |
| Document 7 | 5 / 7 =~ 0.714 | 5 / 10 = 0.5 |
| Document 8 | 5 / 8 = 0.625 | 5 / 10 = 0.5 |
| Document 9 | 5 / 9 =~ 0.555 | 5 / 10 = 0.5 |
| Document 10 | 5 / 10 = 0.5 | 5 / 10 = 0.5 |

Interpolation graph for both engines:

A graph of a graph with blue and orange lines

Description automatically generated

F-Measure with Beta = 0.75:

A black and white math equation

Description automatically generated

The overall **Precision** and **Recall** of engine 1 are: (0.6, 0.6)

The overall **Precision** and **Recall** of engine 2 are: (0.5, 0.5)

Thus, we get the following F-Measure scores:

Engine 1: (0.75\*0.75 + 1)\*0.6\*0.6 / (0.75\*0.75\*0.6 + 0.6) = 0.5625 / 0.9375 = 0.6.

Engine 2: (0.75\*0.75 + 1)\*0.5\*0.5 / (0.75\*0.75\*0.5 + 0.5) = 0.390625 / 0.78125 = 0.5.

**Engine 1 is better in all metrics possible**.

**Question 2**:

A screenshot of a text

Description automatically generated

**Question 2.1**:

|  |  |
| --- | --- |
| **Term** | **Documents** |
| dog | D1, D3 |
| love | D1, D3 |
| music | D1 |
| listen | D1 |
| roll | D1, D3 |
| stone | D1, D3, D4 |
| information | D2 |
| retrieval | D2 |
| course | D2 |
| can | D3 |
| throw | D3 |
| help | D4 |
| pick | D4 |
| up | D4 |
| road | D4 |

**Question 2.2**:

|  |  |
| --- | --- |
| **Term** | **Documents** |
| dogs | D1 |
| love | D1 |
| music | D1 |
| listen | D1 |
| rolling | D1, D3 |
| stones | D1, D3, D4 |
| information | D2 |
| retrieval | D2 |
| course | D2 |
| dog | D3 |
| can | D3 |
| roll | D3 |
| loves | D3 |
| throwing | D3 |
| help | D4 |
| pick | D4 |
| up | D4 |
| road | D4 |

**Question 2.3**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Term** | **Rank** | **Frequency** | **C** |
| stone | 1 | 3 | 3 |
| love | 2 | 2 | 4 |
| dog | 3 | 2 | 6 |
| roll | 4 | 2 | 8 |
| listen | 5 | 1 | 5 |
| music | 6 | 1 | 6 |
| information | 7 | 1 | 7 |
| retrieval | 8 | 1 | 8 |
| course | 9 | 1 | 9 |
| can | 10 | 1 | 10 |
| throw | 11 | 1 | 11 |
| help | 12 | 1 | 12 |
| pick | 13 | 1 | 13 |
| up | 14 | 1 | 14 |
| road | 15 | 1 | 15 |

A graph of a line

Description automatically generated

We see that the C values for each term **does not** follow Zipf’s Law.

According to the image found in the next page:

A math equations on a white background

Description automatically generated

**Question 3**:



**Question 3.1:**

Information needed to extract:

* We want to extract the number of job positions open for a specific skill, library or description (I.E ‘Software Development’, ‘Databases’, etc.).
* We want to extract skills needed for a specific job position.

For the first requirement we will use the following queries in tandem:

1. Given a search word we will execute the following (in our case we will execute these steps for a multitude of predefined job descriptions and save the results): f'https://www.linkedin.com/jobs/search/?currentJobId=3822734767&f\_AL=true&geoId=101620260&keywords={search\_word}&location=Israel&origin=JOB\_SEARCH\_PAGE\_SEARCH\_BUTTON&refresh=true'. (search\_word is simply the skill, library or job description we want to find).
2. soup.find('title'). This finds the title, the number of jobs is posted there in the title so we extract it.

For the second requirement we will use the following queries in tandem:

1. Given a job description\search word we will execute the following: soup.find('ul',class\_='jobs-search\_\_results-list').find\_all('li'), to find all the job listings
2. li.find('a')['href'], to find the link to the job listing.
3. soup\_job\_details.find('div',class\_='description\_\_text').find('section').find('div').find\_all('ul'). This will extract all unordered lists from the description section (which has the requirements).
4. We will then extract only the parts with “experience” or “years” written in them: li.find("years") != -1 or li.find("experience") != -1 or li.find("Experience") != -1.

**Question 3.2**

For the sake of completing this assignment on time we will solve all the 3 questions by executing the second query on the word “tensorflow”, and saved the results to “job\_requirements.csv”.

Our query yielded 47 results.

Here are the 15 most common words in the search results:

('experience', 232), ('learning', 61), ('data', 60), ('years', 43), ('machine', 37), ('tensorflow', 32), ('pytorch', 31), ('frameworks', 29), ('models', 28), ('working', 26), ('cloud', 26), ('science', 25), ('deep', 23), ('3', 23), ('ml', 23)

Here is the Inverted Index for said words in **all pages returned** (created using a custom-made code we will share along with the code for the crawler):

1. experience: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]
2. learning: [0, 1, 2, 3, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 21, 22, 24, 26, 27, 28, 31, 32, 33, 35, 38, 40, 41, 42, 43, 44]
3. data: [1, 2, 4, 6, 8, 9, 10, 11, 13, 15, 18, 19, 20, 21, 22, 23, 25, 27, 28, 31, 33, 35, 36, 37, 40, 42, 43, 44, 45]
4. years: [1, 2, 5, 7, 8, 9, 10, 12, 13, 14, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 42, 43, 44, 45, 46]
5. machine: [1, 2, 7, 8, 10, 11, 13, 15, 21, 22, 27, 28, 31, 32, 35, 38, 40, 42, 43, 44]
6. tensorflow: [0, 1, 2, 4, 6, 9, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 26, 27, 28, 31, 32, 33, 38, 40, 41, 42, 43, 45]
7. pytorch: [0, 1, 2, 4, 6, 9, 11, 13, 14, 15, 16, 17, 18, 20, 22, 23, 24, 26, 27, 28, 31, 32, 33, 38, 40, 41, 42, 43, 45]
8. frameworks: [0, 1, 2, 4, 6, 8, 11, 13, 14, 15, 16, 17, 22, 23, 24, 27, 28, 31, 33, 39, 40, 41, 42, 43, 46]
9. models: [1, 2, 7, 11, 12, 15, 17, 21, 23, 25, 27, 28, 29, 34, 36, 39, 40, 42, 43]
10. working: [1, 2, 4, 6, 7, 11, 15, 18, 20, 21, 22, 25, 31, 33, 38, 43]
11. cloud: [2, 7, 8, 11, 15, 18, 20, 21, 24, 27, 28, 30, 32, 33, 36, 37, 38, 39, 43, 45]
12. science: [1, 2, 5, 9, 11, 13, 15, 18, 20, 21, 27, 28, 31, 33, 37, 40]
13. deep: [0, 2, 3, 9, 11, 14, 15, 16, 17, 21, 24, 26, 27, 28, 33, 38, 41]
14. 3: [1, 2, 9, 11, 13, 14, 15, 18, 19, 20, 23, 24, 26, 27, 28, 31, 32, 33, 36]
15. ml: [4, 6, 7, 8, 9, 22, 23, 25, 26, 34, 36, 39, 42, 43, 45, 46]

Here is the inverted index for the 15 most common words for the first 20 pages only:

1. experience: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
2. learning: [0, 1, 2, 3, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17]
3. data: [1, 2, 4, 6, 8, 9, 10, 11, 13, 15, 18, 19]
4. years: [1, 2, 5, 7, 8, 9, 10, 12, 13, 14, 18, 19]
5. machine: [1, 2, 7, 8, 10, 11, 13, 15]
6. tensorflow: [0, 1, 2, 4, 6, 9, 11, 13, 14, 15, 16, 17, 18]
7. pytorch: [0, 1, 2, 4, 6, 9, 11, 13, 14, 15, 16, 17, 18]
8. frameworks: [0, 1, 2, 4, 6, 8, 11, 13, 14, 15, 16, 17]
9. models: [1, 2, 7, 11, 12, 15, 17]
10. working: [1, 2, 4, 6, 7, 11, 15, 18]
11. cloud: [2, 7, 8, 11, 15, 18]
12. science: [1, 2, 5, 9, 11, 13, 15, 18]
13. deep: [0, 2, 3, 9, 11, 14, 15, 16, 17]
14. 3: [1, 2, 9, 11, 13, 14, 15, 18, 19]
15. ml: [4, 6, 7, 8, 9]

**Question 3.2.2**

There are 4 csv files in the homework assignment:

1. “job\_requirements.csv” – which is the query result
2. “TermFrequecny.csv” – which has the term frequency calculation for each document and each term.
3. “InverseDocumentFrequency.csv” – which has the IDF calculation for each term.
4. “TF-IDF.csv” – which has the TF-IDF calculation for each term and each document.

The code that generated the csv files will also be provided with the crawler code.

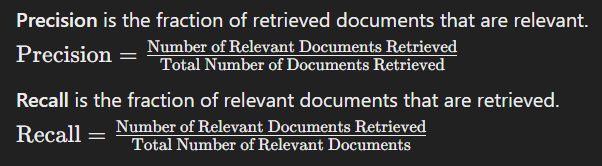
**Question 3.2.3**

Out of the first 10 results only 9 are actually relevant – all but document 4.

User 1 looked at the first 10 results and found that the following documents are relevant: [ 1, 2, 3, 5, 8, 9, 10 ] while the following are irrelevant: [ 4, 6, 7 ].

User 2 looked at the first 10 results and found that the following documents are relevant: [ 1, 2, 6, 7, 9, 10 ] while the following are irrelevant: [ 3, 4, 5, 8 ].

We will use the following formulas to calculate Precision and Recall:



1. User 1:

* Precision = 7 / 10 = 0.7.
* Recall = 7 / 9 =~ 0.77.

1. User 2:

* Precision = 6 / 10 = 0.6.
* Recall = 6 / 9 =~ 0.66.