CS 469 Information Retrieval Assignment 2

Group Number 8

Name	ID
Nimish Agrawal	2020B3A71857H
NIKHIL MOGALAPALLI	2020B1A72398H
SRIGOPAL SARDA	2020AAPS2099H
LINGUTLA SAI CHARAN	2020A7PS1296H

April 28, 2024

Introduction

1 Dataset

The datasets provided for this assignment offer a comprehensive collection of resources for conducting information retrieval tasks in medical information and nutrition-related research. The primary dataset that was represented and used in a major part of the assignment is doc dump, supplemented by the GENA Knowledge Graph, offering diverse information for exploration and analysis.

2 Queries Division and Significance

NFCorpus is the cornerstone of our information retrieval experiments, providing a rich collection of natural language queries, medical documents, and relevance judgments. We have extracted different queries from our dataset.

The NFCorpus dataset is divided into three distinct parts: training, validation, and test sets.

2.1 Training Queries

The training set constitutes the largest portion of the query data,comprising 80 percentage of the total queries in the NFCorpus dataset. This subset is utilized during the initial stages of model development, where the primary objective is to train the ranking models using supervised learning techniques.

2.2 Validation Queries

The validation set, comprising 10 percentage of the total queries, is an intermediate checkpoint during the model development process.

2.3 Test Queries

The test set, comprising the remaining 10 percentage of the queries, serves as the final evaluation benchmark for the developed ranking models.

Experiment 1: Indexing the datasets

1 Introduction

Experiment 1 aimed to create indexing. We have done indexing of the whole pre-processed data on doc dump.txt

Experiment 2: Vector-based Models

1 Introduction

The vector space model (VSM) is a common information retrieval (IR) model representing documents and queries as vectors in a multidimensional space. The VSM is an algebraic model that uses vectors to represent natural language documents formally. The distance between vectors represents the relevance between documents

2 General terminology

2.1 Term frequency tf

The term frequency tft,d of term t in document d is defined as the number of times that t occurs in d.

2.2 Log-frequency weighting

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

2.3 Document frequency D_f

D_f is the document frequency of t, which represents the number of documents that contain term t.

2.4 inverse document frequency idf

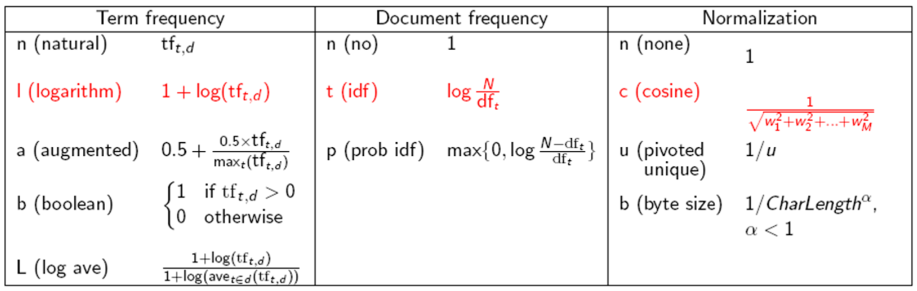
$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

3 Methodology

3.1 tf-idf weighting

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

3.1.1 Variants of tf-idf weighting



Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

3.2 Score

$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} \mathrm{tf.idf}_{t,d}$$

3.3 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discounted Cumulative Gain (NDCG) is a ranking quality metric. It compares rankings to an ideal order where all relevant items are at the top of the list.

$$\log[(1 - u_t)/u_t] = \log[(N - \mathrm{df}_t)/\mathrm{df}_t] \approx \log N/\mathrm{df}_t$$

4 Results Summary

NDCG scores are calculated based on the top 3 documents retrieved for each query.

```
---------------  file is   nfcorpus/train.nontopic-titles.queries

Average Query Time (nnn): 0.10727603011336063
Average Query Time (ntn): 0.10556768117297856
Average Query Time (ntc): 0.06121734314900966
Average NDCG (ntn): [0.173531989482909, 0.181580484954718, 0.17577414692564]
Average NDCG (nnn): [0.12094653812445229, 0.12829389424481438, 0.1285016678246797]
Average NDCG (ntc): [0.16988022202746117, 0.19563248612328377, 0.18532545181042318, ]

---------------  file is   nfcorpus/train.titles.queries

Average Query Time (nnn): 0.20985050603988636
Average Query Time (ntn): 0.21059568082725624
Average Query Time (ntc): 0.07737526548028267
Average NDCG (ntn): [0.20682343870470338, 0.20962477512207645, 0.20303284290019846]
Average NDCG (nnn): [0.1736700077101004, 0.17773708558211243, 0.1737503506481392]
Average NDCG (ntc): [0.21793883320483193, 0.2307761500899509, 0.21974824401833656]

 ---------------  file is   nfcorpus/train.vid-desc.queries

Average Query Time (nnn): 0.16648842519140009
Average Query Time (ntn): 0.16705658195054002
Average Query Time (ntc): 0.06975990827447676
Average NDCG (ntn): [0.18226600985221666, 0.2002463054187192, 0.19764965947065763]
Average NDCG (nnn): [0.11371100164203614, 0.12799671592775033, 0.1248722058588825]
Average NDCG (ntc): [0.2126436781609194, 0.22007389162561583, 0.21966432388094834]
```

```
 ---------------  file is   nfcorpus/train.vid-titles.queries

Average Query Time (nnn): 0.10302891725389829
Average Query Time (ntn): 0.10292079765808407
Average Query Time (ntc): 0.06193686823539546
Average NDCG (ntn): [0.1917077175697864, 0.1985426929392446, 0.189449766492694]
Average NDCG (nnn): [0.14121510673234822, 0.14488916256157622, 0.14221171199524305]
Average NDCG (ntc): [0.18657635467980288, 0.20909277504105095, 0.19876473176938755]
```

Experiment 3: Rocchio Feedback Algorithm for Query Expansion using Pseudo-Relevance Feedback

1 Introduction

In the Rocchio feedback algorithm for query expansion using pseudo-relevance feedback, the top 25 documents are selected based on their relevance to the initial query. These selected documents serve as the basis for determining which terms to add or weight differently in the expanded query, aiming to improve retrieval performance

2 Methodology

2.1 Centroid

Centroid Is centre of mass of a set of points

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d \in C} \vec{d}$$

2.2 Rocchio Algorithm

Rocchio use vsm to pick relevance feedback query¿it is use to increase Recall

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

3 Results Summary

```
 ----------------- file is :  nfcorpus/train.nontopic-titles.queries

average Query time =   0.1398141200243643
average update time =  0.15107729453773064
average ndcg before =  [0.16988022202746117, 0.19563248612328377, 0.18532545181042318]
average ndcg after =  [0.15205959684487286, 0.17421852176453415, 0.16811288771299054]

 ----------------- file is :  nfcorpus/train.titles.queries

average Query time =   0.13992770479197858
average update time =  0.14997415277897255
average ndcg before =  [0.21793883320483193, 0.2307761500899509, 0.21974824401833656]
average ndcg after =  [0.19641480339244435, 0.21017733230531976, 0.2002545347199929]

 ----------------- file is :  nfcorpus/train.vid-desc.queries

average Query time =   0.13621755392093376
average update time =  0.1443889977896742
average ndcg before =  [0.2126436781609194, 0.22007389162561583, 0.21966432388094834]
average ndcg after =  [0.19540229885057459, 0.20806650246305414, 0.20504992958330384]

 ----------------- file is :  nfcorpus/train.vid-titles.queries

average Query time =   0.13359281817093271
average update time =  0.1397666793151442
average ndcg before =  [0.18657635467980288, 0.20909277504105095, 0.19876473176938755]
average ndcg after =  [0.17220853858784899, 0.19041461412151053, 0.18337760495289995]
```

Experiment 4: Probabilistic Retrieval

1 Language models

Probability language models estimate the likelihood of word sequences in natural language

1.1 Methodology

1.1.1 Rank documents

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

1.1.2 P(q/d)

Probability of q given d.

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\mathrm{tf}_{t,q}}$$

2 Okapi BM25

The BM25 model, a probabilistic information retrieval model, assesses the relevance of documents to a given query based on term frequency and document length. It's widely used in search engines for ranking documents by their relevance to user queries

2.1 Methodology

2.1.1 A Nonbinary Model

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t}$$

$$f_{\mathrm{COOR}}(q,d) = \sum_{e:\vec{E}_q(e)>0} \mathbb{1}(\vec{E}_d(e) > 0)$$

$$\log[(1-u_t)/u_t] = \log[(N - \mathrm{df}_t)/\mathrm{df}_t] \approx \log N/\mathrm{df}_t$$

3 Results Summary

```
------------- file is  :  nfcorpus/train.nontopic-titles.queries

averageQuerytimeLang =  0.13522839420830127
averageQuerytimeBm25 =  0.0442301727615878
averageNdcgLang =  [0.17630733274905044, 0.1959684487291849, 0.18715340912321418]
averageNdcgBm25 =  [0.16038562664329525, 0.16526438796377443, 0.15417120392926742]

------------- file is  :  nfcorpus/train.titles.queries

averageQuerytimeLang =  0.09775524609991469
averageQuerytimeBm25 =  0.0353208024775697
averageNdcgLang =  [0.2375353379593937, 0.24300308404009266, 0.23449748192904865]
averageNdcgBm25 =  [0.2180030840400927, 0.21860704189154465, 0.20681550719900624]

------------- file is  :  nfcorpus/train.vid-desc.queries

averageQuerytimeLang =  0.3668275052690741
averageQuerytimeBm25 =  0.12425026693954844
averageNdcgLang =  [0.16830870279146143, 0.1685344827586205, 0.16404420789871815]
averageNdcgBm25 =  [0.048029556650246295, 0.045566502463054194, 0.04141290401170877]

------------- file is  :  nfcorpus/train.vid-titles.queries

averageQuerytimeLang =  0.12390832331380233
averageQuerytimeBm25 =  0.0422202190154879
averageNdcgLang =  [0.20176518883415426, 0.2163793103448274, 0.20521192743669736]
averageNdcgBm25 =  [0.18308702791461406, 0.18074712643678142, 0.16799252164185602]
```

Experiment 5: Entity-based retrieval models l

1 Introduction

Capture knowledge in the form of graphs

2 General terminology

2.0.1 Entities

In knowledge graphs and databases, entities are objects or things that exist and are distinguishable, such as people, places, concepts, or tangible objects.

2.0.2 Relations

Relations are connections or associations between entities in a knowledge graph or database. They describe how entities are related to each other and provide valuable contextual information.

2.0.3 Triples

Triples are basic building blocks in knowledge graphs, consisting of three parts: subject, predicate (or relation), and object. They represent a statement or fact about the relationship between two entities

3 Methodology

3.1 Bag of Entities

Instead of a vector of words, build a vector of entities The vocabulary of entities comes from some existing knowledge base

3.2 Coordinate Match

$$f_{\mathrm{COOR}}(q,d) = \sum_{e:\vec{E}_q(e)>0} \mathbb{1}(\vec{E}_d(e) > 0)$$

3.3 Entity Frequency

$$r_f(o_k) = \sum_{d_j \in D} tf(d_j, o_k) \log \frac{|F|}{df(o_k)}.$$

4 Results Summary

Experiment 6: Query Expansion using Knowledge Graphs

1 Introduction

objects are the entities in a knowledge graph

2 Methodology

2.1 Create a ranked list of objects

Create a ranked list of objects O = o1, o2, .., oK with ranking scores r(O) = r(o1), r(o2),..,r(oK)

2.2 Coordinate Match

$$r_f(o_k) = \sum_{d_j \in D} tf(d_j, o_k) \log \frac{|F|}{df(o_k)}.$$

2.3 Select expansion terms

$$s_p(t_i) = \sum_{o_k \in O} \frac{tf(e(o_k), t_i)}{|e(o_k)|} \times r(o_k) \times \log \frac{|E|}{df(t_i)}$$

3 Results Summary

Experiment 7: Learning to Rank models

1 Introduction

Learn how to combine predefined features for ranking by means of discriminative learning

2 TYPE

2.1 Pointwise

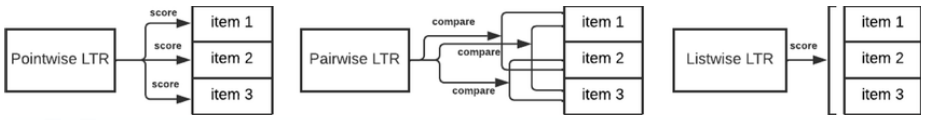
In pointwise learning to rank, each document is treated individually. The relevance score or label for each document is considered independently of other documents. Features are extracted from each document-query pair, and a model is trained to predict the relevance of each individual document. This method is straightforward and computationally efficient but may not capture the interdependence between documents.

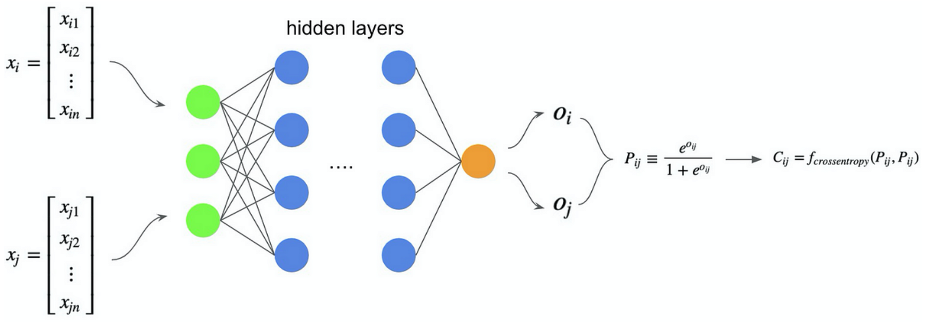
2.2 Pairwise

In pairwise learning to rank, the focus is on comparing pairs of documents within the same query. Rather than predicting the relevance of individual documents, the model learns to predict which document in a pair is more relevant to the query. This approach addresses the issue of capturing the relative relevance between documents but requires careful handling of ties and can be computationally more demanding.

2.3 Listwise

In listwise learning to rank, the entire ranked list of documents for a query is considered as a single unit. The model is trained to directly optimize a loss function that measures the quality of the entire ranked list. Listwise methods aim to directly optimize the ranking performance metric, such as Mean Average Precision (MAP) or Normalized Discounted Cumulative Gain (NDCG), and can potentially yield better performance than pairwise or pointwise methods. However, they can also be more complex to implement and computationally intensive.




$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{bmatrix}$
$x_j = \begin{bmatrix} x_{j1} \\ x_{j2} \\ \vdots \\ x_{jn} \end{bmatrix}$
hidden layers
....
o_i
o_j
$P_{ij} \equiv \frac{e^{o_{ij}}}{1 + e^{o_{ij}}}$
$C_{ij} = f_{crossentropy}(P_{ij}, P_{ij})$

Experiment 8: out-of-the-box

1 Ensemble Learning with Fine-tuning:

Train an ensemble of retrieval models using ensemble learning with fine-tuning. This includes fine-tuned pretrained models like BERT as well as more conventional models like BM25 and language model-based techniques. Utilizing strategies like stacking or averaging, combine their predictions to take advantage of each model's advantages.

2 Query Expansion with Semantic Matching:

Add semantically related terms or phrases to the original query by using semantic matching techniques. Next, apply techniques such as Rocchio feedback or pseudo-relevance feedback to merge the enlarged query with the original query in order to enhance the query representation and retrieval efficiency.

3 Combining Learning to Rank Models:

Try creating a single ranking ensemble by combining various learning to rank models, such as pairwise, listwise, and pointwise methods. To combine the rankings generated by each model and raise the overall ranking quality, apply strategies like rank fusion or rank aggregation.

4 Hybrid Retrieval Systems

Create a hybrid retrieval system that combines vector space models, probabilistic models, and neural models, among other retrieval techniques, in a seamless manner. Utilize a fusion mechanism, such as early or late fusion, to bring the retrieval results from various components together and give the user a single ranking.