CS 469 Information Retrieval Assignment 1

Maheswar Bora
2021A7TS1505H

Aayush Kumar Singh
2021A7PS0430H

Dhananjay Shalley
2021A7PS0401H

Atharva Chikhale
2021A7PS2752H

February 29, 2024

1 Introduction

The code was implemented in Python 3.10 as per the instructions provided. The `main.py` script contains the indexing and querying processes. The `utility.py` contains the initial helper functions that were used for indexing. Also, NLTK was used for stemming, lemmatization and stop word removal. Finally we used memprofiler and cprofiler for profiling and comparing the various methods.

2 Methodology

2.1 Experiment 1: Comparing grep with Inverted Index-based Boolean Retrieval

2.1.1 Implementation details

Approach
• Grep Command: Query terms need to appear consecutively for matching
• Boolean Retrieval (Inverted Index): Non-consecutive occurrences of query terms within documents

Time Complexity
• Grep Command: Indexing doesn't happpen, resulting in a complexity of O(N×m), where N is the number of documents, and m is the estimated length of the search term(s)/sliding window.
• Boolean Retrieval (Inverted Index): Index creation incurs O(nlogn+m) complexity, with n is number of unique tokens and m is the document corpus size. O(nlogn) dominates in most cases due to a large number of unique tokens.
• Queryworst case time complexity is O(ΣP) when all lists are disjoint.

Memory Complexity
• Grep Command: Requires constant memory. O(k), where k is sliding window size.
• Boolean Retrieval (Inverted Index): Worst case O(t*d) memory, with t representing unique terms and d is number of documents in corpus. Average case is much better

provided documents are in same language.

2.1.2 Findings

Using grep:
Average execution time: **26969.270164** μs
Minimum execution time: **24952.010019** μs
Maximum execution time: **37849.588902** μs

Using Inverted Index Boolean Retrieval:
Average time per query: **83.298152** μs.
Minimum time for a query: **6.755930** μs.
Maximum time for a query: **1039.450988** μs.

2.1.3 Results Table

To further elucidate the comparative performance of the grep utility versus the inverted index-based retrieval, refer to the following table of results:

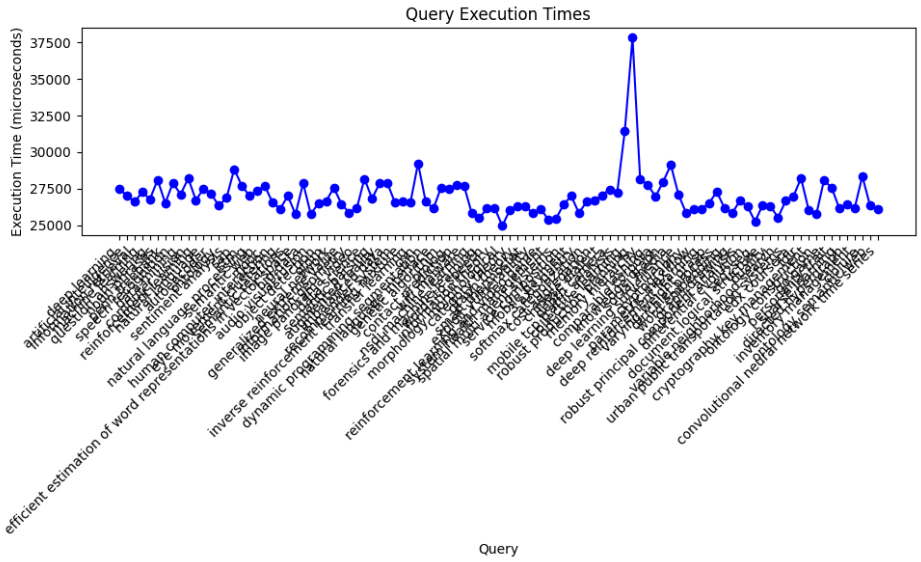

Figure 1: grep execution time per query.

Query	Document Matches	Total Matches	Execution Time (μs)
deep learning	114	204	27459.660079330206
artificial intelligence	130	176	27034.088969230652
information retrieval	194	300	26596.31590358913
machine learning	327	496	27256.510918959975
question answering	108	231	26781.70904982835
noun phrases	50	131	28102.174983359873
penn treebank	56	91	26508.250972256064

speech recognition	191	362	27893.019025214016
data mining	153	298	27102.183899842203
computer vision	189	263	28204.004978761077
reinforcement learning	149	366	26707.483106292784
natural language	351	579	27501.8090615049
autoencoder	89	248	27127.329958602786
ontology	259	870	26332.900975830853
sentiment analysis	86	188	26869.558030739427
sap	96	328	28832.1670377627
lstm	101	414	27692.60189961642
natural language processing	156	193	27003.85299976915
semantic web	147	310	27351.984987035394
mooc	155	817	27679.109945893288
human computer interaction	21	26	26554.384035989642
eye movement clustering	0	0	26118.094101548195
semantic relations	60	122	27004.809002391994
efficient estimation of word representations in vector space	1	1	25742.032914422452
big data	96	258	27869.277982972562
audio visual fusion	0	0	25748.28395154327
object detection	103	189	26510.667987167835
gfdm	60	175	26632.308959960938
neural network	472	873	27564.316987991333
generalized extreme value	27	31	26454.843929968774
information geometry	38	51	25857.455912046134
image panorama video	0	0	26141.22093655169
data science	14	27	28173.587983474135
semantic parsing	42	82	26820.75893972069
augmented reality	107	235	27882.874943315983
imbalanced data	86	183	27870.981954038143
recommender system	112	325	26532.772928476334
inverse reinforcement learning mixture	0	0	26623.503072187304
transfer learning	62	154	26531.868963502347
cnn	137	503	29226.271086372435
dynamic programming segmentation	1	1	26609.471067786217
natural language interface	49	68	26133.911916986108
genetic algorithm	106	191	27566.697914153337
prolog	103	430	27470.628963783383
contact prediction	45	116	27780.976030044258
wifi malware	0	0	27707.339031621814
nsdi machine learning	0	0	25838.515954092145
forensics and machine learning	0	0	25477.71204262972
words to speech	0	0	26137.955952435732
information theory	51	76	26184.769929386675

morphology morphological	0	0	24952.010018751025
category theory	47	65	26056.92599900067
graph theory	52	80	26311.579975299537
smart thermostat	10	18	26307.66201764345
exploit vulnerability	0	0	25802.682037465274
reinforcement learning and video game	0	0	26127.42001656443
system health management	7	7	25387.194007635117
spatial multi agent systems	0	0	25432.22694657743
service composition	83	202	26410.68201046437
mobile payment	90	220	27002.095011994243
3 axis gantry	0	0	25823.91502801329
softmax categorization	0	0	26605.769991874695
cost aggregation	37	72	26671.669096685946
chinese dialect	21	29	27053.134981542826
depth camera	33	57	27444.445993751287
mobile tcp traffic analysis	0	0	27218.464063480496
collective learning	30	39	31475.61103105545
robust production planning	2	2	37849.58890173584
memory hierarchy	46	88	28121.19305599481
hashing	111	376	27769.08001396805
comparable corpora	80	163	26951.500913128257
knowledge graph	16	37	27973.659918643534
social media	125	375	29125.96706300974
deep learning surveillance	0	0	27096.157078631222
cryptography	133	249	25834.7790222615
parametric max flow	3	3	26116.174994967878
deep reinforcement learning	8	11	26104.49492931366
varying weight grasp	0	0	26502.98399385065
dirichlet process	65	150	27297.20203205943
word embedding	57	130	26173.432008363307
graph drawing	81	119	25814.1279919073
robust principal component analysis	45	65	26660.07890366018
differential evolution	94	153	26293.513015843928
seq2seq	9	26	25245.13006210327
document logical structure	9	14	26393.032050691545
duality	98	229	26297.945994883776
variable neighborhood search	86	106	25525.806937366724
urban public transportation systems	1	1	26670.566061511636
edx coursera	0	0	26967.148995026946
fdir	25	52	28218.723949976265
cryptography key management	1	1	26041.948003694415
ontology construction	47	56	25748.934014700353
go game	7	7	28079.175972379744

personality trait	72	163	27556.015993468463
sparse learning	5	7	26155.473082326353
directed hypergraph	39	71	26450.678007677197
inventory management	82	98	26163.266971707344
clojure	42	96	28350.569074973464
ontology semantic web	0	0	26372.986030764878
convolutional neural network time series	0	0	26117.180939763784

Query Times for Boolean Retrieval

Query	Document Count	Processing Time
deep learning	234	148.64502009004354
artificial intelligence	140	38.59598655253649
information retrieval	256	153.75204384326935
machine learning	412	123.54296632111073
question answering	127	43.5439869761467
noun phrases	55	24.625915102660656
penn treebank	81	24.568988010287285
speech recognition	235	82.50190876424313
data mining	206	165.76610505580902
computer vision	210	76.39697287231684
reinforcement learning	158	76.31198968738317
natural language	391	116.70100502669811
autoencoder	74	13.70999962091446
ontology	258	17.535057850182056
sentiment analysis	94	116.27294588834047
sap	69	9.730923920869827
lstm	97	10.711024515330791
natural language processing	183	134.21603944152594
semantic web	203	98.51192589849234
mooc	133	13.279961422085762
human computer interaction	119	109.33005250990391
eye movement clustering	8	45.684981159865856
semantic relations	85	41.70008469372988
efficient estimation of word representations in vector space	1	1039.4509881734848
big data	114	95.63704952597618
audio visual fusion	56	58.8749535381794
object detection	151	78.52900307625532
gfdm	60	9.746057912707329
neural network	333	114.19993825256824
generalized extreme value	33	57.669938541948795
information geometry	75	81.13193325698376
image panorama video	10	74.27402306348085
data science	97	94.72807869315147

semantic parsing	79	42.13897045701742
augmented reality	111	31.4878998324275
imbalanced data	92	83.3999365568161
recommender system	55	141.81190636008978
inverse reinforcement learning mixture	1	82.82193448394537
transfer learning	96	74.88601841032505
cnn	116	12.16307282447815
dynamic programming segmentation	32	76.73597428947687
natural language interface	53	72.06399459391832
genetic algorithm	76	109.46998372673988
prolog	101	10.724063031375408
contact prediction	64	41.78809467703104
wifi malware	10	16.486970707774162
nsdi machine learning	0	39.99401815235615
forensics and machine learning	11	556.2800215557218
words to speech	67	477.7879221364856
information theory	175	114.01204392313957
morphology morphological	38	21.710991859436035
category theory	65	39.004022255539894
graph theory	77	63.0220165476203
smart thermostat	15	20.712031982839108
exploit vulnerability	28	27.57494803518057
reinforcement learning and video game	1	447.53006659448147
system health management	37	109.62598025798798
spatial multi agent systems	18	164.76097516715527
service composition	98	41.98099486529827
mobile payment	86	42.809988372027874
3 axis gantry	2	55.44302985072136
softmax categorization	1	16.318983398377895
cost aggregation	53	45.11093720793724
chinese dialect	27	23.85093830525875
depth camera	70	35.85300873965025
mobile tcp traffic analysis	4	97.28095028549433
collective learning	56	72.24094588309526
robust production planning	18	62.63295654207468
memory hierarchy	58	44.726068153977394
hashing	111	11.495081707835197
comparable corpora	88	37.33194898813963
knowledge graph	66	76.60698611289263
social media	138	50.286995247006416
deep learning surveillance	15	84.94302164763212
cryptography	133	11.913012713193893
parametric max flow	7	34.608994610607624
deep reinforcement learning	10	97.61203546077013
varying weight grasp	1	30.477996915578842
dirichlet process	61	57.774013839662075

word embedding	42	32.57195930927992
graph drawing	84	34.195021726191044
robust principal component analysis	60	122.3050057888031
differential evolution	100	37.36407961696386
seq2seq	9	6.755930371582508
document logical structure	47	80.27302101254463
duality	96	11.083087883889675
variable neighborhood search	90	70.64198143780231
urban public transportation systems	14	95.04903573542833
edx coursera	19	13.779033906757832
fdir	25	7.329974323511124
cryptography key management	18	64.62307646870613
ontology construction	53	31.689065508544445
go game	39	24.622073397040367
personality trait	39	20.520063117146492
sparse learning	111	80.04903793334961
directed hypergraph	21	23.550004698336124
inventory management	87	33.57906825840473
clojure	42	8.652103133499622
ontology semantic web	84	64.31504152715206
convolutional neural network time series	3	167.69696958363056

2.2 Experiment 2: Linguistic Post-processing

2.2.1 Techniques Applied

Stemming, lemmatization, and stopword removal were applied to normalize the text. We get a total of **20522** indexes in the postings list after performing this operation.

2.2.2 Results

Average time per query: **189.837152** μs.
Minimum time for a query: **63.942978** μs.
Maximum time for a query: **442.081946** μs.

Boolean Retrieval Query time after stemming, lemmatization and stop word removal

Query	Document Count	Processing Time
deep learn	254	315.93802850693464
artifici intellig	143	166.88299365341663
inform retriev	289	270.48494666814804
machin learn	467	246.23889476060867
question answer	214	180.49997743219137
noun phrase	70	137.70698569715023
penn treebank	82	124.6799947693944

speech recognit	236	191.99401140213013
data mine	218	281.2549937516451
comput vision	247	256.21603708714247
reinforc learn	164	195.35096362233162
natur languag	432	249.00108110159636
autoencod	89	106.45308066159487
ontolog	292	107.56298433989286
sentiment analysi	94	226.97006352245808
sap	70	111.1950259655714
lstm	99	90.88893420994282
natur languag process	214	328.77991907298565
semant web	219	208.80799274891615
mooc	153	95.52401024848223
human comput interact	153	308.84891748428345
eye movement cluster	10	193.48296336829662
semant relat	177	204.68700677156448
effici estim word represent vector space	1	442.08194594830275
big data	114	197.581946849823
audio visual fusion	56	199.0519231185317
object detect	210	218.33996288478374
gfdm	60	98.31192437559366
neural network	510	260.8589129522443
gener extrem valu	42	364.85702730715275
inform geometri	77	194.97191533446312
imag panorama video	19	212.21500355750322
data scienc	110	207.88901019841433
semant pars	95	158.40097330510616
augment realiti	112	148.06201215833426
imbalanc data	92	197.26203754544258
recommend system	162	273.86704459786415
invers reinforc learn mixtur	1	241.91499687731266
transfer learn	108	191.38690549880266
cnn	127	92.93400216847658
dynam program segment	33	249.45405311882496
natur languag interfac	74	228.31803653389215
genet algorithm	124	243.53410117328167
prolog	101	93.93005166202784
contact predict	80	175.31507182866335
wifi malwar	10	121.82001955807209
nsdi machin learn	0	157.28897415101528
forens machin learn	13	226.48496087640524
word speech	131	156.51003923267126
inform theori	198	231.95194080471992
morpholog morpholog	135	128.2669836655259
categori theori	80	151.0159345343709
graph theori	87	174.7979549691081

smart thermostat	23	122.33993038535118
exploit vulner	68	159.5430076122284
reinforc learn video game	1	255.76294865459204
system health manag	51	250.22495537996292
spatial multi agent system	20	289.871939457953
servic composit	105	167.4590166658163
mobil payment	96	157.26499259471893
axi gantri	9	120.88497169315815
softmax categor	2	131.39809016138315
cost aggreg	66	161.80402599275112
chines dialect	33	129.6279951930046
depth camera	86	139.28103726357222
mobil tcp traffic analysi	4	241.3969486951828
collect learn	193	231.24006111174822
robust product plan	27	219.3100517615676
memori hierarchi	68	152.6799751445651
hash	141	103.46993803977966
compar corpu	144	234.19491481035948
knowledg graph	73	188.41703422367573
social medium	148	158.051960170269
deep learn surveil	17	220.11995315551758
cryptographi	133	97.289914265275
parametr max flow	7	158.11098273843527
deep reinforc learn	11	214.65809550136328
vari weight grasp	3	179.68309111893177
dirichlet process	70	179.86400052905083
word embed	80	152.5740372017026
graph draw	96	149.81592539697886
robust princip compon analysi	61	277.6680048555136
differenti evolut	101	148.68902508169413
	0	63.94297815859318
document logic structur	58	274.0250201895833
dualiti	96	98.48398622125387
variabl neighborhood search	90	212.8009218722582
urban public transport system	19	274.5509846135974
edx coursera	19	112.42600157856941
fdir	25	86.22801396995783
cryptographi key manag	19	198.50104581564665
ontolog construct	68	186.67103722691536
go game	42	121.294055134058
person trait	88	151.9819488748908
spars learn	135	198.03002942353487
direct hypergraph	50	158.35592057555914
inventori manag	87	141.65998436510563
clojur	42	91.98498446494341
ontolog semant web	103	194.36900038272142

convolut neural network time seri	8	387.58700247853994

2.3 Experiment 3: Hash-based vs. Tree-based Dictionaries

2.3.1 Implementation Details

Approach

- Dictionary used for storing postings can have hash maps or tries in backend. Hash Based

Time Complexity

- Trie-based Retrieval: Lookup time is $O(m)$ where m is the length of the token. Therefore, longer terms take longer to search. Worst case $O(\sum P)$ for `and_query` when all doc_id sets are disjoint.
- Hash-based Retrieval: $O(1)$ lookup time for retrieving a query term's posting list. Worst case $O(\sum P)$ for `and_query` when all doc_id sets are disjoint.
- Trie takes longer than hash-table based implementation, and this difference is more pronounced when the token is longer.

Memory Complexity

- Memory taken by hash table based dictionary is $O(T \times (L_{avg} + D_{avg}))$ where T is the number of unique tokens, L_{avg} is the average length of a token, and D_{avg} is the average length of a postings list for a token.
- Memory taken by tries depends on the amount of common prefixes. $O(N+T\times D_{avg})$ where T is the number of unique tokens and N is the number of nodes in the Trie.
- Tries are more space-efficient in storing words with common prefixes, but the overhead of node pointers and dynamic containers for doc_ids at each node leads to higher memory usage compared to a flat hash table here.

2.3.2 Results

For Trie Based Implementation:
Average time per query: **93.841022** μs.
Minimum time for a query: **8.367002** μs.
Maximum time for a query: **1338.128932** μs.
Maximum memory used by Trie based implementation: **288.250 MiB**.
Maximum memory used by Hash based implementation: **264.656 MiB**.

Query-wise time for Tree based implementation:

Query	Document Count	Processing Time
deep learning	234	164.56993762403727
artificial intelligence	140	49.30200520902872

information retrieval	256	211.65399812161922
machine learning	412	131.32498133927584
question answering	127	47.92097024619579
noun phrases	55	31.632022000849247
penn treebank	81	36.45394463092089
speech recognition	235	123.82003478705883
data mining	206	190.4219388961792
computer vision	210	86.32300887256861
reinforcement learning	158	87.16003503650427
natural language	391	123.94099030643702
autoencoder	74	17.2819709405303
ontology	258	20.47303132712841
sentiment analysis	94	126.98699720203876
sap	69	11.633033864200115
lstm	97	12.491014786064625
natural language processing	183	124.38895646482706
semantic web	203	98.40703569352627
mooc	133	15.88999293744564
human computer interaction	119	122.69092258065939
eye movement clustering	8	51.712035201489925
semantic relations	85	48.90502896159887
efficient estimation of word representations in vector space	1	1277.3259077221155
big data	114	124.14890807121992
audio visual fusion	56	69.77503653615713
object detection	151	85.85606701672077
gfdm	60	11.365045793354511
neural network	333	120.66005729138851
generalized extreme value	33	67.70610343664885
information geometry	75	92.85495616495609
image panorama video	10	97.8340394794941
data science	97	112.24497575312853
semantic parsing	79	49.48803689330816
augmented reality	111	39.30903039872646
imbalanced data	92	97.38898370414972
recommender system	55	129.0569780394435
inverse reinforcement learning mixture	1	95.86603846400976
transfer learning	96	79.92703467607498
cnn	116	14.137942343950272
dynamic programming segmentation	32	85.93709208071232
natural language interface	53	95.76207958161831
genetic algorithm	76	118.26399713754654
prolog	101	20.129024051129818
contact prediction	64	48.681977204978466
wifi malware	10	21.335086785256863
nsdi machine learning	0	32.84809645265341

forensics and machine learning	11	573.1550045311451
words to speech	67	524.0869941189885
information theory	175	123.59197717159986
morphology morphological	38	26.977970264852047
category theory	65	42.430940084159374
graph theory	77	67.33497139066458
smart thermostat	15	24.43697303533554
exploit vulnerability	28	32.40606747567654
reinforcement learning and video game	1	376.6709705814719
system health management	37	119.83898002654314
spatial multi agent systems	18	180.4659841582179
service composition	98	49.9130692332983
mobile payment	86	64.32703230530024
3 axis gantry	2	72.22301792353392
softmax categorization	1	22.23497722297907
cost aggregation	53	49.474998377263546
chinese dialect	27	28.586946427822113
depth camera	70	39.616948924958706
mobile tcp traffic analysis	4	111.14694643765688
collective learning	56	82.89096876978874
robust production planning	18	69.8600197210908
memory hierarchy	58	61.99802737683058
hashing	111	13.95493745803833
comparable corpora	88	43.669017031788826
knowledge graph	66	83.78399070352316
social media	138	56.58098962157965
deep learning surveillance	15	99.77992158383131
cryptography	133	14.90896102041006
parametric max flow	7	42.17505920678377
deep reinforcement learning	10	90.47798812389374
varying weight grasp	1	36.65708936750889
dirichlet process	61	64.13902156054974
word embedding	42	39.89890683442354
graph drawing	84	41.7840201407671
robust principal component analysis	60	147.17597514390945
differential evolution	100	43.8059214502573
seq2seq	9	9.78098250925541
document logical structure	47	87.725929915905
duality	96	12.174947187304497
variable neighborhood search	90	77.88697257637978
urban public transportation systems	14	107.65099432319403
edx coursera	19	17.542042769491673
fdir	25	9.292969480156898
cryptography key management	18	73.6479414626956
ontology construction	53	38.92707172781229
go game	39	26.97098534554243

personality trait	39	24.61206167936325
sparse learning	111	90.58404248207808
directed hypergraph	21	28.33595499396324
inventory management	87	39.68097735196352
clojure	42	10.629999451339245
ontology semantic web	84	73.22302553802729
convolutional neural network time series	3	178.5058993846178

2.4 Experiment 4: Wildcard Querying

2.4.1 Implementation Details

Approach

- Permuterm Index: Utilizes token rotations to transform wildcard queries into prefix searches.
- Forward and Backward (FW_BW) Index: Employs two tries to index words in both original and reversed orders. Wildcard queries are separated into prefix and postfix searches. Then each of these searches is performed on Forward Trie and Backward Trie respectively. The results from both tries are intersected to find tokens matching the wildcard.

Time Complexity

- Creation of permuterm takes: $O(T \times (L+1) \times 2)$ where T is the number of unique tokens and L is the average length of token which will have $L+1$ rotations after being appended with '$'.
- Rotation of query wildcard to make it a prefix matching problem is $O(Q+1)$ where L is the length of the Query. It is also appended with '$'.
- Lookup in permuterm index is $O(P \times N)$, where P is the length of the rotated wildcard query and N is the number of entries in permuterm index.
- Creation of each of the tries takes: $O(T \times \text{avg len of token})$ where T is the number of unique tokens.
- Searching in each trie involves lookup in $O(P)$ where P is prefix or suffix length.
- Intersection of both lists takes $O(m+n)$ where m and n are matches of prefix and suffix respectively.
- Forward Backward index using trie is faster than using permuterm index because in the second case we need to search the whole permuterm index while in the first case we skip the non matching subtree.

Memory Complexity

- $O(T \times (L+1) \times 2)$ for permuterm as we store each token with each of its rotations.

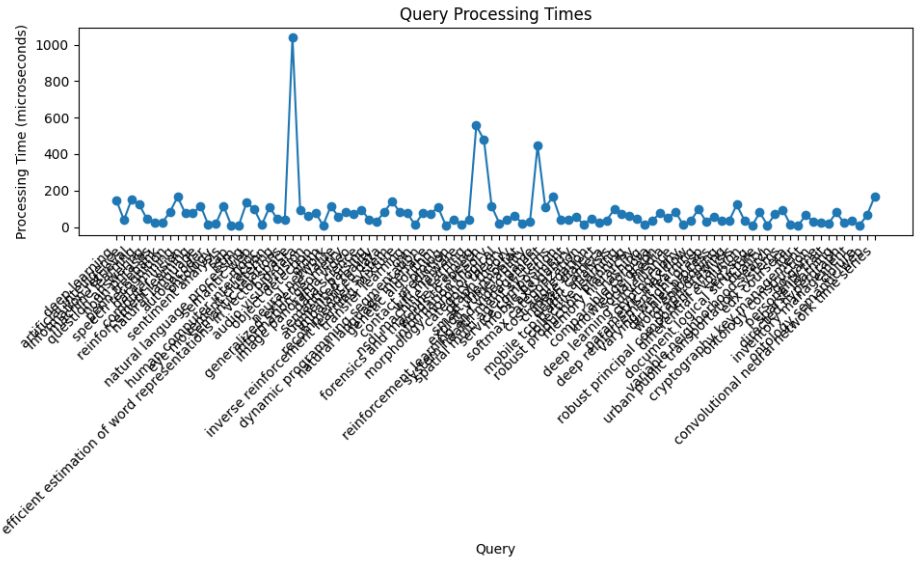

Figure 2: Boolean Retrieval execution time per query.

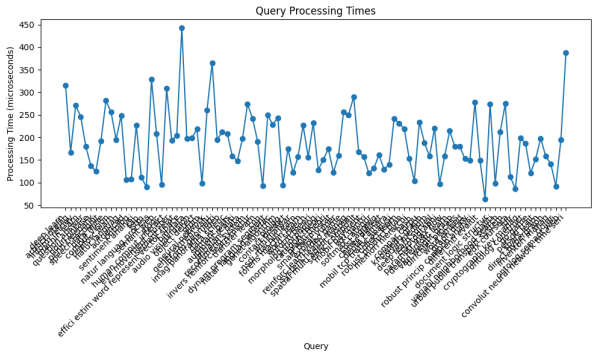

Figure 3: Query processing time of Boolearn retrieval after lemmatization

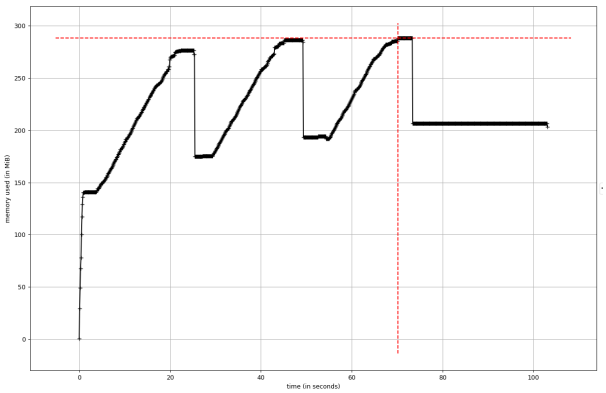

Figure 4: Trie based dictionary memory graph over 3 consecutive runs.

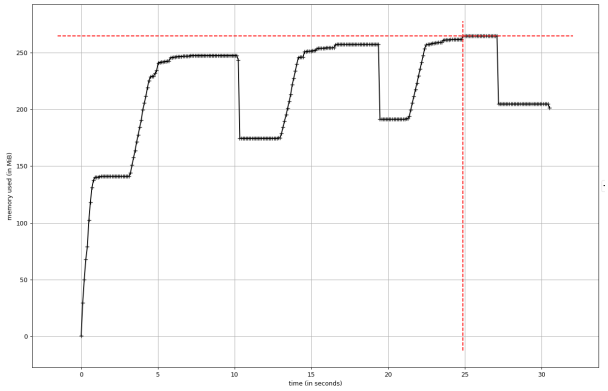

Figure 5: Hash based memory graph over 3 consecutive runs.

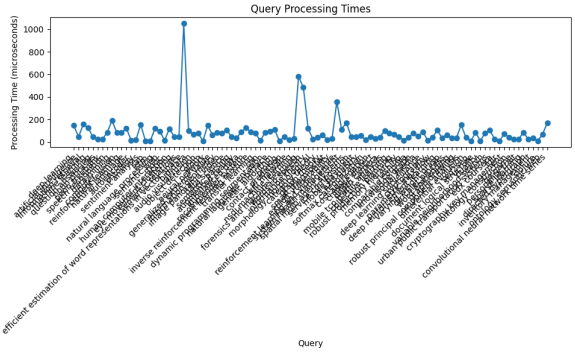

Figure 6: Trie based dictionary memory graph over 3 consecutive runs.

- It's $O(2 \times N)$ where N is the number of nodes in tries where N depends on how many words have common prefixes.
- We need to store all rotations of tokens in permuterm index, hence it takes more memory than forward and backward index tries.

2.4.2 Results

Average time per query: **37664.628293** μs.
Minimum time for a query: **36748.284008** μs.
Maximum time for a query: **39471.737924** μs.

Query	**Document Count**	**Processing Time** μs
l*ng	56	38010.974996723235
in*e	171	37977.0539002493
ret*val	1	38434.740039519966
mac*	38	37514.720926992595
*nswering	1	37905.868026427925
*ses	172	37920.11097539216
tree*	8	37708.0449834466
sp*h	3	37030.02806287259
m*g	111	37230.99594935775

v*n	51	36748.28400835395
rein*	17	37880.15991449356
lan*ge	2	37774.38902761787
auto*	51	37597.93902281672
on*y	8	37755.63708040863
senti*	11	37999.22298640013
s*p	63	37174.435099586844
l*m	22	37234.41401962191
nat*l	3	37164.860987104475
we*	119	37536.12493164837
mo*c	22	37122.99489416182
com*er	8	39471.73792403191
cl*g	18	37556.38492293656
s*ic	69	37514.86202236265
re*s	243	37637.49997597188
*ata	19	37349.41699076444
f*sion	3	37801.61205213517
o*ject	1	38294.30800396949
g*dm	2	37232.18000959605
n*ural	2	37459.15403123945
e*treme	1	37900.69301612675

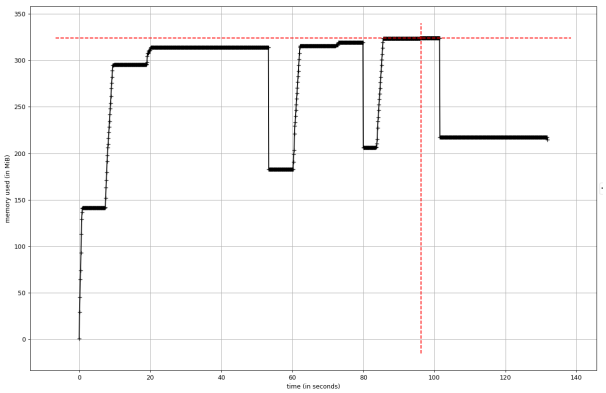

Figure 7: Wild card querying using permuterm index memory graph over 3 consecutive runs.

2.4.3 Tree-based Indexes

Forward and backward trie indexes supported complex wildcard queries.

Average time per query: **13226.690636** μs.
Maximum time for a query: **41552.380077** μs.
Minimum time for a query: **160.977012** μs.

Query	**Document Count**	**Processing Time**

l*ng	2415	3698.7579660490155
in*e	3357	6129.181943833828
ret*val	375	160.97701154649258
mac*	878	41552.38007660955
*nswering	150	30288.905021734536
*ses	2924	31201.98892429471
tree*	381	41092.0400172472
sp*h	406	862.9889925941825
m*g	2417	5314.378999173641
v*n	1021	3664.563992060721
rein*	188	40606.740047223866
lan*ge	962	276.63190849125385
auto*	1244	40900.22703167051
on*y	1338	3604.1870480403304
senti*	123	40696.83502893895
s*p	586	3394.0280554816127
l*m	137	1640.9169184044003
nat*l	774	2009.061980061233
we*	6138	41014.57307115197
mo*c	179	1703.9489466696978
com*er	694	1466.7380601167679
cl*g	345	2870.1419942080975
s*ic	1992	3851.1409657076
re*s	4408	10948.839015327394
*ata	2552	32139.530056156218
f*sion	122	1173.0390833690763
o*ject	471	846.7120351269841
g*dm	61	907.0070227608085
n*ural	1243	1086.9778925552964
e*treme	93	1697.27997854352

2.5 Experiment 5: Tolerant Retrieval

2.5.1 Implemenation

Approach

- Use forward and backward index tries to reduce wildcards to query terms. Then use edit distance (threshold increases as query term becomes longer) calculated by Levenshtein Distance to manage spelling mistakes in query terms. Then intersection for overall wildcard matches. Additionally, employs Levenshtein distance to identify closely spelled words.

This comprehensive analysis highlights the trade-offs between different retrieval methods in terms of efficiency, complexity, and applicability to various search requirements.

Time Complexity

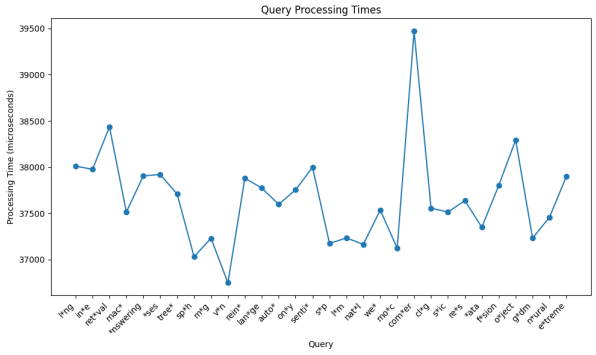

Figure 8: Wild card querying using permuterm index query processing times

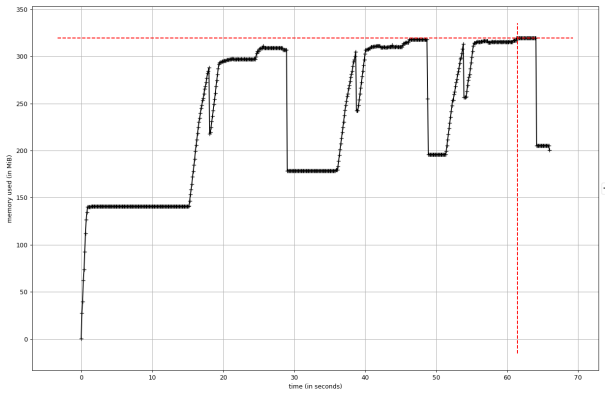

Figure 9: Memory usage of Forward and Backward Tree indexing over 3 consecutive runs

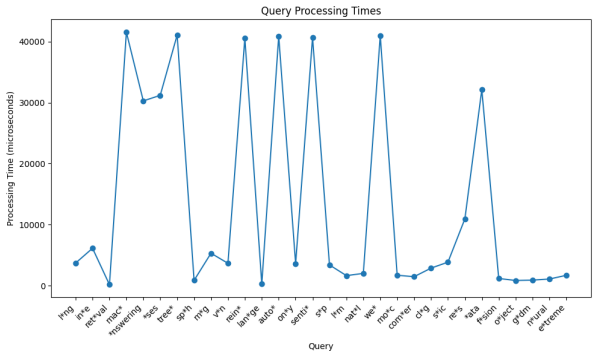

Figure 10: Processing time per query for Forward and Backward Tree indexing.

- Levenshtein distance calculation takes $O(n \times m)$ where n, m are sizes of the terms.
- Searching prefix matching terms and postfix matching terms takes $O(m + k)$ where m is the length of prefix or postfix of query and k is matching words (nodes) of trie.
- Union of postings corresponding takes $O(\sum P)$ in worst case where P is the size of postings.

Memory Complexity

- $O(2 \times N)$ for forward and backward tries where N is the number of nodes in tries, where N depends on how many words have common prefixes.

2.5.2 Observations

Forward-backward indices using tries generally perform better but perform noticeably worse when the '' is in either end of the wild card query as seen in the spikes in the graph. This is because one of the forward or backward trie is not being used.

2.5.3 Results

Average time per query: **380.455298** ms.
Minimum time for a query: **0.935313** ms.
Maximum time for a query: **1075.509554** ms.

Query	**Document Count**	**Processing Time**
Query	Results Count	Processing Time
deep l*ng	262	300.37545796949416
artificial in*e	185	667.4139079404995
*tion ret*val	369	35.927033983170986
mac* learning	478	591.2097569089383
question *nswering	133	571.8973459443077
noun *ses	64	328.50218191742897
n tree	377	88.05875794496387
sp*h recognition	239	734.0690040728077
data m*g	974	298.45780099276453
computer v*n	315	560.0412540370598
rein* learning	163	572.887085028924
natural lan*ge	391	475.01332603860646
auto*	1244	42.675414006225765
on*y	1338	3.69984598364681
senti* analysis	96	597.5445590447634
s*p	586	3.8466069381684065
l*m	137	1.7478680238127708
nat*l language *ing	375	588.9658259693533
semantic we*	666	682.5603580800816
mo*c	179	2.0749319810420275
hu* com*er interaction	147	812.0284540345892

e* movement cl*g	15	591.370249981992
s*ic relations	194	616.7521979659796
e*t est* of word re*s in v*r space	4	1075.509554008022
big *ata	115	261.34333200752735
a*o v*l f*sion	57	6.630899966694415
o*ject *tion	465	33.47160096745938
g*dm	61	0.9353130590170622
n*ural network	558	471.92694002296776
genera* e*treme value	37	396.7220619088039

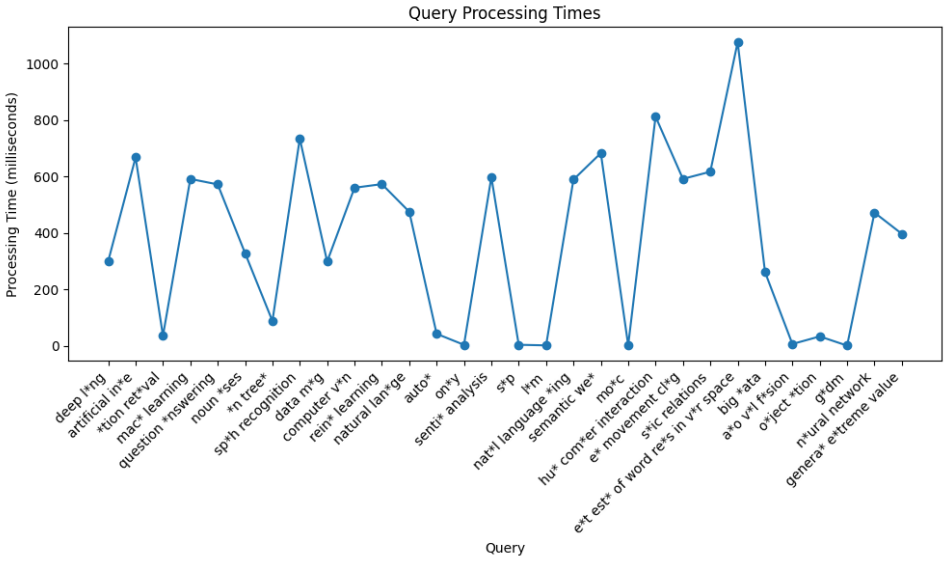

Figure 11: Tolerant Retrieval Processing time per query.

3 Conclusion

We have explored and implemented several types of retrieval methods that we have learnt in the class. This included Boolean retrieval using inverted index, wild card query retrieval and tolerant retrieal. During this we learnt about tradeoffs between query time, indexing time and memory.