Lab 2: Document Preprocessing, Indexing, and Relevance Feedback

Modules Covered

- Module 3: Document Preprocessing, Indexing & Searching
- Module 4: Evaluation of IR Systems
- Module 5: Relevance Feedback and Query Expansion

Objective

Implement and evaluate core IR techniques including tokenization, normalization, inverted indexing, relevance feedback, and query expansion. Build a mini IR pipeline and test its performance using precision, recall, and feedback mechanisms.

Submission: Submit both .ipynb file and .ipynb converted to PDF

Submissions with following cases will get a zero

- Code or commented text truncated from the pdf version of the notebook
- Any compilation error in the notebook
- Missing output for any of the programming cells. There should be an output for every code cell

✓ Submission Checklist

- Preprocessing code and inverted index output
- Evaluation metrics for at least 3 queries
- Expanded queries and feedback results
- Answers to reflection questions

□ Part A: Document Preprocessing and Indexing

Task 1: Tokenization and Normalization

- Load a small corpus of 5–10 sample documents (e.g., news articles or Wikipedia snippets).
- Apply tokenization using NLTK or spaCy.
- Normalize tokens:
  - Lowercasing
  - Removing punctuation
  - Handling Unicode and spelling variants

In [1]:
```
#code/program Task 1

# Using wikipedia articles
import wikipedia

titles = ["Python (programming language)", "Badminton", "Soccer", "Computer Science", "Data science"]
documents = {}
for title in titles:
    page = wikipedia.page(title)
    documents[title] = page.summary

# Made documents a dictionary where key is title and value is the summary
documents
```

Out[1]:
```
{'Python (programming language)': 'Python is a high-level, general-purpose programming language. Its design phi
losophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked
and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedura
l), object-oriented and functional programming.\nGuido van Rossum began working on Python in the late 1980s as
a successor to the ABC programming language. Python 3.0, released in 2008, was a major revision and not complet
ely backward-compatible with earlier versions. Beginning with Python 3.5, capabilities and keywords for typing
were added to the language, allowing optional static typing. Currently only versions in the 3.x series are supp
orted. \nPython has gained widespread use in the machine learning community. It is widely taught as an introduc
tory programming language. Since 2003, Python has consistently ranked in the top ten of the most popular progra
```

```
mming languages in the TIOBE Programming Community Index, which ranks based on searches in 24 platforms.\n\n',
 'Badminton': 'Badminton is a racquet sport played using racquets to hit a shuttlecock across a net. Although i
t may be played with larger teams, the most common forms of the game are "singles" (with one player per side) a
nd "doubles" (with two players per side). Badminton is often played as a casual outdoor activity in a yard or o
n a beach; professional games are played on a rectangular indoor court. Points are scored by striking the shutt
lecock with the racquet and landing it within the other team\'s half of the court, within the set boundaries.\n
Each side may only strike the shuttlecock once before it passes over the net. Play ends once the shuttlecock ha
s struck the floor or ground, or if a fault has been called by the umpire, service judge, or (in their absence)
the opposing side.\nThe shuttlecock is a feathered or (in informal matches) plastic projectile that flies diffe
rently from the balls used in many other sports. In particular, the feathers create much higher drag, causing t
he shuttlecock to decelerate more rapidly. Shuttlecocks also have a high top speed compared to the balls in oth
er racquet sports, making badminton the fastest racquet sport in the world. The flight of the shuttlecock gives
the sport its distinctive nature, and in certain languages the sport is named by reference to this feature (e.g
., German Federball, literally feather-ball).\nThe game developed in British India from the earlier game of bat
tledore and shuttlecock. European play came to be dominated by Denmark but the game has become very popular in
Asia. In 1992, badminton debuted as a Summer Olympic sport with four events: men\'s singles, women\'s singles,
men\'s doubles, and women\'s doubles; mixed doubles was added four years later. At high levels of play, the spo
rt demands excellent fitness: players require aerobic stamina, agility, strength, speed, and precision. It is a
lso a technical sport, requiring good motor coordination and the development of sophisticated racquet movements
involving much greater flexibility in the wrist than some other racquet sports.',
 'Soccer': "Association football, more commonly known as football or soccer, is a team sport played between two
teams of 11 players who almost exclusively use their feet to propel a ball around a rectangular field called a
pitch. \nThe objective of the game is to score more goals than the opposing team by moving the ball beyond the
goal line into a rectangular-framed goal defended by the opponent. Traditionally, the game has been played over
two 45-minute halves, for a total match time of 90 minutes. With an estimated 250 million players active in ove
r 200 countries and territories, it is the world's most popular sport.\nAssociation football is played in accor
dance with the Laws of the Game, a set of rules that has been in effect since 1863 and maintained by the IFAB s
ince 1886. The game is played with a football that is 68–70 cm (27–28 in) in circumference. The two teams compe
te to score goals by getting the ball into the other team's goal (between the posts, under the bar, and fully a
cross the goal line). When the ball is in play, the players mainly use their feet, but may also use any other p
art of their body, except for their hands or arms, to control, strike, or pass the ball; the head, chest, and t
highs are commonly used. Only the goalkeepers may use their hands and arms, but only within their own penalty a
rea. The team that has scored more goals at the end of the game is the winner. Depending on the format of the c
ompetition, an equal number of goals scored may result in a draw being declared with 1 point awarded to each te
am, or the game may go into extra time or a penalty shoot-out.\nInternationally, association football is govern
ed by FIFA. Under FIFA, there are six continental confederations: AFC, CAF, CONCACAF, CONMEBOL, OFC, and UEFA.
National associations (e.g. the FA in England, U.S. Soccer in the United States, etc.) are responsible for mana
ging the game in their own countries both professionally and at an amateur level, and coordinating competitions
in accordance with the Laws of the Game. The most prestigious senior international competition is the FIFA Worl
d Cup. The men's World Cup is the most-viewed sporting event in the world, surpassing the Olympic Games. The mo
st prestigious competition in European club football is the UEFA Champions League, which attracts an extensive
television audience worldwide. The final of the men's Champions League is the most-watched annual sporting even
t in the world.",
 'Computer Science': 'Computer science is the study of computation, information, and automation. Included broad
ly in the sciences, computer science spans theoretical disciplines (such as algorithms, theory of computation,
and information theory) to applied disciplines (including the design and implementation of hardware and softwar
e). An expert in the field is known as a computer scientist. \nAlgorithms and data structures are central to co
mputer science.\nThe theory of computation concerns abstract models of computation and general classes of probl
ems that can be solved using them. The fields of cryptography and computer security involve studying the means
for secure communication and preventing security vulnerabilities. Computer graphics and computational geometry
address the generation of images. Programming language theory considers different ways to describe computationa
l processes, and database theory concerns the management of repositories of data. Human–computer interaction in
vestigates the interfaces through which humans and computers interact, and software engineering focuses on the
design and principles behind developing software. Areas such as operating systems, networks and embedded system
s investigate the principles and design behind complex systems. Computer architecture describes the constructio
n of computer components and computer-operated equipment. Artificial intelligence and machine learning aim to s
ynthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, plannin
g and learning found in humans and animals. Within artificial intelligence, computer vision aims to understand
and process image and video data, while natural language processing aims to understand and process textual and
linguistic data.\nThe fundamental concern of computer science is determining what can and cannot be automated.
The Turing Award is generally recognized as the highest distinction in computer science.\n\n',
 'Data science': 'Data science is an interdisciplinary academic field that uses statistics, scientific computin
g, scientific methods, processing, scientific visualization, algorithms and systems to extract or extrapolate k
nowledge from potentially noisy, structured, or unstructured data. \nData science also integrates domain knowle
dge from the underlying application domain (e.g., natural sciences, information technology, and medicine). Data
science is multifaceted and can be described as a science, a research paradigm, a research method, a discipline
, a workflow, and a profession.\nData science is "a concept to unify statistics, data analysis, informatics, an
d their related methods" to "understand and analyze actual phenomena" with data. It uses techniques and theorie
s drawn from many fields within the context of mathematics, statistics, computer science, information science,
and domain knowledge. However, data science is different from computer science and information science. Turing
Award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computat
ional, and now data-driven) and asserted that "everything about science is changing because of the impact of in
formation technology" and the data deluge.\nA data scientist is a professional who creates programming code and
combines it with statistical knowledge to summarize data.\n\n'}
```

In [2]:
```
# Tokenizing Text using NLTK
import nltk
tokenized_documents = {}
for title, summary in documents.items():
    tokenized_documents[title] = nltk.word_tokenize(summary)
```

In [3]:
```
# This is a dictionary of lists, where the key is the title,
# and the value is a list of tokenized terms per document
```

```
print(tokenized_documents["Badminton"])
```

```
['Badminton', 'is', 'a', 'racquet', 'sport', 'played', 'using', 'racquets', 'to', 'hit', 'a', 'shuttlecock', 'ac
ross', 'a', 'net', '.', 'Although', 'it', 'may', 'be', 'played', 'with', 'larger', 'teams', ',', 'the', 'most',
'common', 'forms', 'of', 'the', 'game', 'are', '``', 'singles', "''", '(', 'with', 'one', 'player', 'per', 'side
', ')', 'and', '``', 'doubles', "''", '(', 'with', 'two', 'players', 'per', 'side', ')', '.', 'Badminton', 'is',
'often', 'played', 'as', 'a', 'casual', 'outdoor', 'activity', 'in', 'a', 'yard', 'or', 'on', 'a', 'beach', ';',
'professional', 'games', 'are', 'played', 'on', 'a', 'rectangular', 'indoor', 'court', '.', 'Points', 'are', 'sc
ored', 'by', 'striking', 'the', 'shuttlecock', 'with', 'the', 'racquet', 'and', 'landing', 'it', 'within', 'the'
, 'other', 'team', "'s", 'half', 'of', 'the', 'court', ',', 'within', 'the', 'set', 'boundaries', '.', 'Each', '
side', 'may', 'only', 'strike', 'the', 'shuttlecock', 'once', 'before', 'it', 'passes', 'over', 'the', 'net', '.
', 'Play', 'ends', 'once', 'the', 'shuttlecock', 'has', 'struck', 'the', 'floor', 'or', 'ground', ',', 'or', 'if
', 'a', 'fault', 'has', 'been', 'called', 'by', 'the', 'umpire', ',', 'service', 'judge', ',', 'or', '(', 'in',
'their', 'absence', ')', 'the', 'opposing', 'side', '.', 'The', 'shuttlecock', 'is', 'a', 'feathered', 'or', '('
, 'in', 'informal', 'matches', ')', 'plastic', 'projectile', 'that', 'flies', 'differently', 'from', 'the', 'bal
ls', 'used', 'in', 'many', 'other', 'sports', '.', 'In', 'particular', ',', 'the', 'feathers', 'create', 'much',
'higher', 'drag', ',', 'causing', 'the', 'shuttlecock', 'to', 'decelerate', 'more', 'rapidly', '.', 'Shuttlecock
s', 'also', 'have', 'a', 'high', 'top', 'speed', 'compared', 'to', 'the', 'balls', 'in', 'other', 'racquet', 'sp
orts', ',', 'making', 'badminton', 'the', 'fastest', 'racquet', 'sport', 'in', 'the', 'world', '.', 'The', 'flig
ht', 'of', 'the', 'shuttlecock', 'gives', 'the', 'sport', 'its', 'distinctive', 'nature', ',', 'and', 'in', 'cer
tain', 'languages', 'the', 'sport', 'is', 'named', 'by', 'reference', 'to', 'this', 'feature', '(', 'e.g.', ',',
'German', 'Federball', ',', 'literally', 'feather-ball', ')', '.', 'The', 'game', 'developed', 'in', 'British',
'India', 'from', 'the', 'earlier', 'game', 'of', 'battledore', 'and', 'shuttlecock', '.', 'European', 'play', 'c
ame', 'to', 'be', 'dominated', 'by', 'Denmark', 'but', 'the', 'game', 'has', 'become', 'very', 'popular', 'in',
'Asia', '.', 'In', '1992', ',', 'badminton', 'debuted', 'as', 'a', 'Summer', 'Olympic', 'sport', 'with', 'four',
'events', ':', 'men', "'s", 'singles', ',', 'women', "'s", 'singles', ',', 'men', "'s", 'doubles', ',', 'and', '
women', "'s", 'doubles', ';', 'mixed', 'doubles', 'was', 'added', 'four', 'years', 'later', '.', 'At', 'high', '
levels', 'of', 'play', ',', 'the', 'sport', 'demands', 'excellent', 'fitness', ':', 'players', 'require', 'aerob
ic', 'stamina', ',', 'agility', ',', 'strength', ',', 'speed', ',', 'and', 'precision', '.', 'It', 'is', 'also',
'a', 'technical', 'sport', ',', 'requiring', 'good', 'motor', 'coordination', 'and', 'the', 'development', 'of',
'sophisticated', 'racquet', 'movements', 'involving', 'much', 'greater', 'flexibility', 'in', 'the', 'wrist', 't
han', 'some', 'other', 'racquet', 'sports', '.']
```

In [4]:
```
# Normalizing tokens (FIRST, LOWERCASING)
lowercased_tokenized_documents = {}
for title, tokenized_doc in tokenized_documents.items():
    lowercased_tokenized_documents[title.lower()] = [token.lower() for token in tokenized_doc]

# Using badminton document as the reference document to show in this report
print(lowercased_tokenized_documents["badminton"])
```

```
['badminton', 'is', 'a', 'racquet', 'sport', 'played', 'using', 'racquets', 'to', 'hit', 'a', 'shuttlecock', 'ac
ross', 'a', 'net', '.', 'although', 'it', 'may', 'be', 'played', 'with', 'larger', 'teams', ',', 'the', 'most',
'common', 'forms', 'of', 'the', 'game', 'are', '``', 'singles', "''", '(', 'with', 'one', 'player', 'per', 'side
', ')', 'and', '``', 'doubles', "''", '(', 'with', 'two', 'players', 'per', 'side', ')', '.', 'badminton', 'is',
'often', 'played', 'as', 'a', 'casual', 'outdoor', 'activity', 'in', 'a', 'yard', 'or', 'on', 'a', 'beach', ';',
'professional', 'games', 'are', 'played', 'on', 'a', 'rectangular', 'indoor', 'court', '.', 'points', 'are', 'sc
ored', 'by', 'striking', 'the', 'shuttlecock', 'with', 'the', 'racquet', 'and', 'landing', 'it', 'within', 'the'
, 'other', 'team', "'s", 'half', 'of', 'the', 'court', ',', 'within', 'the', 'set', 'boundaries', '.', 'each', '
side', 'may', 'only', 'strike', 'the', 'shuttlecock', 'once', 'before', 'it', 'passes', 'over', 'the', 'net', '.
', 'play', 'ends', 'once', 'the', 'shuttlecock', 'has', 'struck', 'the', 'floor', 'or', 'ground', ',', 'or', 'if
', 'a', 'fault', 'has', 'been', 'called', 'by', 'the', 'umpire', ',', 'service', 'judge', ',', 'or', '(', 'in',
'their', 'absence', ')', 'the', 'opposing', 'side', '.', 'the', 'shuttlecock', 'is', 'a', 'feathered', 'or', '('
, 'in', 'informal', 'matches', ')', 'plastic', 'projectile', 'that', 'flies', 'differently', 'from', 'the', 'bal
ls', 'used', 'in', 'many', 'other', 'sports', '.', 'in', 'particular', ',', 'the', 'feathers', 'create', 'much',
'higher', 'drag', ',', 'causing', 'the', 'shuttlecock', 'to', 'decelerate', 'more', 'rapidly', '.', 'shuttlecock
s', 'also', 'have', 'a', 'high', 'top', 'speed', 'compared', 'to', 'the', 'balls', 'in', 'other', 'racquet', 'sp
orts', ',', 'making', 'badminton', 'the', 'fastest', 'racquet', 'sport', 'in', 'the', 'world', '.', 'the', 'flig
ht', 'of', 'the', 'shuttlecock', 'gives', 'the', 'sport', 'its', 'distinctive', 'nature', ',', 'and', 'in', 'cer
tain', 'languages', 'the', 'sport', 'is', 'named', 'by', 'reference', 'to', 'this', 'feature', '(', 'e.g.', ',',
'german', 'federball', ',', 'literally', 'feather-ball', ')', '.', 'the', 'game', 'developed', 'in', 'british',
'india', 'from', 'the', 'earlier', 'game', 'of', 'battledore', 'and', 'shuttlecock', '.', 'european', 'play', 'c
ame', 'to', 'be', 'dominated', 'by', 'denmark', 'but', 'the', 'game', 'has', 'become', 'very', 'popular', 'in',
'asia', '.', 'in', '1992', ',', 'badminton', 'debuted', 'as', 'a', 'summer', 'olympic', 'sport', 'with', 'four',
'events', ':', 'men', "'s", 'singles', ',', 'women', "'s", 'singles', ',', 'men', "'s", 'doubles', ',', 'and', '
women', "'s", 'doubles', ';', 'mixed', 'doubles', 'was', 'added', 'four', 'years', 'later', '.', 'at', 'high', '
levels', 'of', 'play', ',', 'the', 'sport', 'demands', 'excellent', 'fitness', ':', 'players', 'require', 'aerob
ic', 'stamina', ',', 'agility', ',', 'strength', ',', 'speed', ',', 'and', 'precision', '.', 'it', 'is', 'also',
'a', 'technical', 'sport', ',', 'requiring', 'good', 'motor', 'coordination', 'and', 'the', 'development', 'of',
'sophisticated', 'racquet', 'movements', 'involving', 'much', 'greater', 'flexibility', 'in', 'the', 'wrist', 't
han', 'some', 'other', 'racquet', 'sports', '.']
```

In [5]:
```
# REMOVING PUNCTUATION AND HANDLING UNICODE USING isascii()
normalized_tokenized_documents = {}

for title, lowercased_doc in lowercased_tokenized_documents.items():
    normalized_tokenized_documents[title] = [
        token for token in lowercased_doc if token.isalpha() and token.isascii()
    ]

print(normalized_tokenized_documents["badminton"])
```

```
['badminton', 'is', 'a', 'racquet', 'sport', 'played', 'using', 'racquets', 'to', 'hit', 'a', 'shuttlecock', 'ac
ross', 'a', 'net', 'although', 'it', 'may', 'be', 'played', 'with', 'larger', 'teams', 'the', 'most', 'common',
'forms', 'of', 'the', 'game', 'are', 'singles', 'with', 'one', 'player', 'per', 'side', 'and', 'doubles', 'with'
, 'two', 'players', 'per', 'side', 'badminton', 'is', 'often', 'played', 'as', 'a', 'casual', 'outdoor', 'activi
ty', 'in', 'a', 'yard', 'or', 'on', 'a', 'beach', 'professional', 'games', 'are', 'played', 'on', 'a', 'rectangu
lar', 'indoor', 'court', 'points', 'are', 'scored', 'by', 'striking', 'the', 'shuttlecock', 'with', 'the', 'racq
uet', 'and', 'landing', 'it', 'within', 'the', 'other', 'team', 'half', 'of', 'the', 'court', 'within', 'the', '
set', 'boundaries', 'each', 'side', 'may', 'only', 'strike', 'the', 'shuttlecock', 'once', 'before', 'it', 'pass
es', 'over', 'the', 'net', 'play', 'ends', 'once', 'the', 'shuttlecock', 'has', 'struck', 'the', 'floor', 'or',
'ground', 'or', 'if', 'a', 'fault', 'has', 'been', 'called', 'by', 'the', 'umpire', 'service', 'judge', 'or', 'i
n', 'their', 'absence', 'the', 'opposing', 'side', 'the', 'shuttlecock', 'is', 'a', 'feathered', 'or', 'in', 'in
formal', 'matches', 'plastic', 'projectile', 'that', 'flies', 'differently', 'from', 'the', 'balls', 'used', 'in
', 'many', 'other', 'sports', 'in', 'particular', 'the', 'feathers', 'create', 'much', 'higher', 'drag', 'causin
g', 'the', 'shuttlecock', 'to', 'decelerate', 'more', 'rapidly', 'shuttlecocks', 'also', 'have', 'a', 'high', 't
op', 'speed', 'compared', 'to', 'the', 'balls', 'in', 'other', 'racquet', 'sports', 'making', 'badminton', 'the'
, 'fastest', 'racquet', 'sport', 'in', 'the', 'world', 'the', 'flight', 'of', 'the', 'shuttlecock', 'gives', 'th
e', 'sport', 'its', 'distinctive', 'nature', 'and', 'in', 'certain', 'languages', 'the', 'sport', 'is', 'named',
'by', 'reference', 'to', 'this', 'feature', 'german', 'federball', 'literally', 'the', 'game', 'developed', 'in'
, 'british', 'india', 'from', 'the', 'earlier', 'game', 'of', 'battledore', 'and', 'shuttlecock', 'european', 'p
lay', 'came', 'to', 'be', 'dominated', 'by', 'denmark', 'but', 'the', 'game', 'has', 'become', 'very', 'popular'
, 'in', 'asia', 'in', 'badminton', 'debuted', 'as', 'a', 'summer', 'olympic', 'sport', 'with', 'four', 'events',
'men', 'singles', 'women', 'singles', 'men', 'doubles', 'and', 'women', 'doubles', 'mixed', 'doubles', 'was', 'a
dded', 'four', 'years', 'later', 'at', 'high', 'levels', 'of', 'play', 'the', 'sport', 'demands', 'excellent', '
fitness', 'players', 'require', 'aerobic', 'stamina', 'agility', 'strength', 'speed', 'and', 'precision', 'it',
'is', 'also', 'a', 'technical', 'sport', 'requiring', 'good', 'motor', 'coordination', 'and', 'the', 'developmen
t', 'of', 'sophisticated', 'racquet', 'movements', 'involving', 'much', 'greater', 'flexibility', 'in', 'the', '
wrist', 'than', 'some', 'other', 'racquet', 'sports']
```

Task 2: Stop Word Removal and Stemming

- Remove stop words using a standard list (e.g., NLTK's English stopwords).
- Apply stemming (Porter or Snowball) and compare with lemmatization.

In [6]:
```python
#code/program Task 2
# Stop word removal
from nltk.corpus import stopwords
stopwords = stopwords.words('english')

removed_stopwords_tokenized_documents = {}
for title, normalized_doc in normalized_tokenized_documents.items():
    removed_stopwords_tokenized_documents[title] = [
        token for token in normalized_doc if token not in stopwords
    ]

print(removed_stopwords_tokenized_documents["badminton"])
```

```
['badminton', 'racquet', 'sport', 'played', 'using', 'racquets', 'hit', 'shuttlecock', 'across', 'net', 'althoug
h', 'may', 'played', 'larger', 'teams', 'common', 'forms', 'game', 'singles', 'one', 'player', 'per', 'side', 'd
oubles', 'two', 'players', 'per', 'side', 'badminton', 'often', 'played', 'casual', 'outdoor', 'activity', 'yard
', 'beach', 'professional', 'games', 'played', 'rectangular', 'indoor', 'court', 'points', 'scored', 'striking',
'shuttlecock', 'racquet', 'landing', 'within', 'team', 'half', 'court', 'within', 'set', 'boundaries', 'side', '
may', 'strike', 'shuttlecock', 'passes', 'net', 'play', 'ends', 'shuttlecock', 'struck', 'floor', 'ground', 'fau
lt', 'called', 'umpire', 'service', 'judge', 'absence', 'opposing', 'side', 'shuttlecock', 'feathered', 'informa
l', 'matches', 'plastic', 'projectile', 'flies', 'differently', 'balls', 'used', 'many', 'sports', 'particular',
'feathers', 'create', 'much', 'higher', 'drag', 'causing', 'shuttlecock', 'decelerate', 'rapidly', 'shuttlecocks
', 'also', 'high', 'top', 'speed', 'compared', 'balls', 'racquet', 'sports', 'making', 'badminton', 'fastest', '
racquet', 'sport', 'world', 'flight', 'shuttlecock', 'gives', 'sport', 'distinctive', 'nature', 'certain', 'lang
uages', 'sport', 'named', 'reference', 'feature', 'german', 'federball', 'literally', 'game', 'developed', 'brit
ish', 'india', 'earlier', 'game', 'battledore', 'shuttlecock', 'european', 'play', 'came', 'dominated', 'denmark
', 'game', 'become', 'popular', 'asia', 'badminton', 'debuted', 'summer', 'olympic', 'sport', 'four', 'events',
'men', 'singles', 'women', 'singles', 'men', 'doubles', 'women', 'doubles', 'mixed', 'doubles', 'added', 'four',
'years', 'later', 'high', 'levels', 'play', 'sport', 'demands', 'excellent', 'fitness', 'players', 'require', 'a
erobic', 'stamina', 'agility', 'strength', 'speed', 'precision', 'also', 'technical', 'sport', 'requiring', 'goo
d', 'motor', 'coordination', 'development', 'sophisticated', 'racquet', 'movements', 'involving', 'much', 'great
er', 'flexibility', 'wrist', 'racquet', 'sports']
```

In [7]:
```python
# Porter Stemmer
from nltk.stem import PorterStemmer
porter = PorterStemmer()

porter_tokenized_documents = {}
for title, no_stopword_doc in removed_stopwords_tokenized_documents.items():
    porter_tokenized_documents[title] = [porter.stem(token) for token in no_stopword_doc]

print(porter_tokenized_documents["badminton"])
```

```
['badminton', 'racquet', 'sport', 'play', 'use', 'racquet', 'hit', 'shuttlecock', 'across', 'net', 'although', '
may', 'play', 'larger', 'team', 'common', 'form', 'game', 'singl', 'one', 'player', 'per', 'side', 'doubl', 'two
', 'player', 'per', 'side', 'badminton', 'often', 'play', 'casual', 'outdoor', 'activ', 'yard', 'beach', 'profes
sion', 'game', 'play', 'rectangular', 'indoor', 'court', 'point', 'score', 'strike', 'shuttlecock', 'racquet', '
land', 'within', 'team', 'half', 'court', 'within', 'set', 'boundari', 'side', 'may', 'strike', 'shuttlecock', '
pass', 'net', 'play', 'end', 'shuttlecock', 'struck', 'floor', 'ground', 'fault', 'call', 'umpir', 'servic', 'ju
dg', 'absenc', 'oppos', 'side', 'shuttlecock', 'feather', 'inform', 'match', 'plastic', 'projectil', 'fli', 'dif
fer', 'ball', 'use', 'mani', 'sport', 'particular', 'feather', 'creat', 'much', 'higher', 'drag', 'caus', 'shutt
lecock', 'deceler', 'rapidli', 'shuttlecock', 'also', 'high', 'top', 'speed', 'compar', 'ball', 'racquet', 'spor
t', 'make', 'badminton', 'fastest', 'racquet', 'sport', 'world', 'flight', 'shuttlecock', 'give', 'sport', 'dist
inct', 'natur', 'certain', 'languag', 'sport', 'name', 'refer', 'featur', 'german', 'federbal', 'liter', 'game',
'develop', 'british', 'india', 'earlier', 'game', 'battledor', 'shuttlecock', 'european', 'play', 'came', 'domin
', 'denmark', 'game', 'becom', 'popular', 'asia', 'badminton', 'debut', 'summer', 'olymp', 'sport', 'four', 'eve
nt', 'men', 'singl', 'women', 'singl', 'men', 'doubl', 'women', 'doubl', 'mix', 'doubl', 'ad', 'four', 'year', '
later', 'high', 'level', 'play', 'sport', 'demand', 'excel', 'fit', 'player', 'requir', 'aerob', 'stamina', 'agi
l', 'strength', 'speed', 'precis', 'also', 'technic', 'sport', 'requir', 'good', 'motor', 'coordin', 'develop',
'sophist', 'racquet', 'movement', 'involv', 'much', 'greater', 'flexibl', 'wrist', 'racquet', 'sport']
```

In [8]:
```python
# Lemmatization
from nltk.stem import WordNetLemmatizer
lemma = WordNetLemmatizer()

lemma_tokenized_documents = {}
for title, no_stopword_doc in removed_stopwords_tokenized_documents.items():
    lemma_tokenized_documents[title] = [lemma.lemmatize(token) for token in no_stopword_doc]

print(lemma_tokenized_documents["badminton"])
```

```
['badminton', 'racquet', 'sport', 'played', 'using', 'racquet', 'hit', 'shuttlecock', 'across', 'net', 'although
', 'may', 'played', 'larger', 'team', 'common', 'form', 'game', 'single', 'one', 'player', 'per', 'side', 'doubl
e', 'two', 'player', 'per', 'side', 'badminton', 'often', 'played', 'casual', 'outdoor', 'activity', 'yard', 'be
ach', 'professional', 'game', 'played', 'rectangular', 'indoor', 'court', 'point', 'scored', 'striking', 'shuttl
ecock', 'racquet', 'landing', 'within', 'team', 'half', 'court', 'within', 'set', 'boundary', 'side', 'may', 'st
rike', 'shuttlecock', 'pass', 'net', 'play', 'end', 'shuttlecock', 'struck', 'floor', 'ground', 'fault', 'called
', 'umpire', 'service', 'judge', 'absence', 'opposing', 'side', 'shuttlecock', 'feathered', 'informal', 'match',
'plastic', 'projectile', 'fly', 'differently', 'ball', 'used', 'many', 'sport', 'particular', 'feather', 'create
', 'much', 'higher', 'drag', 'causing', 'shuttlecock', 'decelerate', 'rapidly', 'shuttlecock', 'also', 'high', '
top', 'speed', 'compared', 'ball', 'racquet', 'sport', 'making', 'badminton', 'fastest', 'racquet', 'sport', 'wo
rld', 'flight', 'shuttlecock', 'give', 'sport', 'distinctive', 'nature', 'certain', 'language', 'sport', 'named'
, 'reference', 'feature', 'german', 'federball', 'literally', 'game', 'developed', 'british', 'india', 'earlier'
, 'game', 'battledore', 'shuttlecock', 'european', 'play', 'came', 'dominated', 'denmark', 'game', 'become', 'po
pular', 'asia', 'badminton', 'debuted', 'summer', 'olympic', 'sport', 'four', 'event', 'men', 'single', 'woman',
'single', 'men', 'double', 'woman', 'double', 'mixed', 'double', 'added', 'four', 'year', 'later', 'high', 'leve
l', 'play', 'sport', 'demand', 'excellent', 'fitness', 'player', 'require', 'aerobic', 'stamen', 'agility', 'str
ength', 'speed', 'precision', 'also', 'technical', 'sport', 'requiring', 'good', 'motor', 'coordination', 'devel
opment', 'sophisticated', 'racquet', 'movement', 'involving', 'much', 'greater', 'flexibility', 'wrist', 'racque
t', 'sport']
```

When comparing Lemmatization and Porter stemmer, we notice that the porter stemmer may be less accurate due some some words being stemmed incorrectly. For example, we notice in the porter stemmer dictionary, the word "doubles" got stemmed into "doubl", which clearly is not right. It was also noticeable that the lemmatization cell took slightly longer to run.

Task 3: Build an Inverted Index

- Create a dictionary mapping each term to a list of document IDs.
- Store term positions for phrase search support.

In [9]:
```python
#code/program Task 3
# This inverted index would need to have the term as keys,
# and another dictionary as values with the document title as keys,
# and a list of word-level indexes as values
inverted_index = {}

for title, tokens in lemma_tokenized_documents.items():
    for position, token in enumerate(tokens):
        if token not in inverted_index:
            inverted_index[token] = {}
        if title not in inverted_index[token]:
            inverted_index[token][title] = []
        inverted_index[token][title].append(position)

inverted_index
```

Out[9]:
```
{'python': {'python (programming language)': [0, 11, 28, 34, 42, 56, 69]},
 'programming': {'python (programming language)': [1, 15, 22, 32, 66, 75, 78],
  'computer science': [67],
  'data science': [116]},
 'language': {'python (programming language)': [2, 33, 47, 67, 76],
  'badminton': [119],
  'computer science': [68, 141]},
 'design': {'python (programming language)': [3],
  'computer science': [22, 91, 104]},
```

```
'philosophy': {'python (programming language)': [4]},
'emphasizes': {'python (programming language)': [5]},
'code': {'python (programming language)': [6], 'data science': [117]},
'readability': {'python (programming language)': [7]},
'use': {'python (programming language)': [8, 59],
 'soccer': [14, 92, 96, 113]},
'significant': {'python (programming language)': [9]},
'indentation': {'python (programming language)': [10]},
'dynamically': {'python (programming language)': [12]},
'support': {'python (programming language)': [13]},
'multiple': {'python (programming language)': [14]},
'paradigm': {'python (programming language)': [16], 'data science': [44, 98]},
'including': {'python (programming language)': [17],
 'computer science': [21]},
'structured': {'python (programming language)': [18], 'data science': [21]},
'particularly': {'python (programming language)': [19]},
'procedural': {'python (programming language)': [20]},
'functional': {'python (programming language)': [21]},
'guido': {'python (programming language)': [23]},
'van': {'python (programming language)': [24]},
'rossum': {'python (programming language)': [25]},
'began': {'python (programming language)': [26]},
'working': {'python (programming language)': [27]},
'late': {'python (programming language)': [29]},
'successor': {'python (programming language)': [30]},
'abc': {'python (programming language)': [31]},
'released': {'python (programming language)': [35]},
'major': {'python (programming language)': [36]},
'revision': {'python (programming language)': [37]},
'completely': {'python (programming language)': [38]},
'earlier': {'python (programming language)': [39], 'badminton': [131]},
'version': {'python (programming language)': [40, 53]},
'beginning': {'python (programming language)': [41]},
'capability': {'python (programming language)': [43]},
'keywords': {'python (programming language)': [44]},
'typing': {'python (programming language)': [45, 51]},
'added': {'python (programming language)': [46], 'badminton': [161]},
'allowing': {'python (programming language)': [48]},
'optional': {'python (programming language)': [49]},
'static': {'python (programming language)': [50]},
'currently': {'python (programming language)': [52]},
'series': {'python (programming language)': [54]},
'supported': {'python (programming language)': [55]},
'gained': {'python (programming language)': [57]},
'widespread': {'python (programming language)': [58]},
'machine': {'python (programming language)': [60], 'computer science': [117]},
'learning': {'python (programming language)': [61],
 'computer science': [118, 125]},
'community': {'python (programming language)': [62, 79]},
'widely': {'python (programming language)': [63]},
'taught': {'python (programming language)': [64]},
'introductory': {'python (programming language)': [65]},
'since': {'python (programming language)': [68], 'soccer': [64, 67]},
'consistently': {'python (programming language)': [70]},
'ranked': {'python (programming language)': [71]},
'top': {'python (programming language)': [72], 'badminton': [100]},
'ten': {'python (programming language)': [73]},
'popular': {'python (programming language)': [74],
 'badminton': [142],
 'soccer': [53]},
'tiobe': {'python (programming language)': [77]},
'index': {'python (programming language)': [80]},
'rank': {'python (programming language)': [81]},
'based': {'python (programming language)': [82]},
'search': {'python (programming language)': [83]},
'platform': {'python (programming language)': [84]},
'badminton': {'badminton': [0, 28, 107, 144]},
'racquet': {'badminton': [1, 5, 46, 104, 109, 189, 196]},
'sport': {'badminton': [2, 86, 105, 110, 115, 120, 148, 168, 182, 197],
 'soccer': [7, 54]},
'played': {'badminton': [3, 12, 30, 38], 'soccer': [8, 39, 57, 69]},
'using': {'badminton': [4], 'computer science': [47]},
'hit': {'badminton': [6]},
'shuttlecock': {'badminton': [7, 45, 58, 63, 75, 94, 97, 113, 134]},
'across': {'badminton': [8], 'soccer': [85]},
'net': {'badminton': [9, 60]},
'although': {'badminton': [10]},
'may': {'badminton': [11, 56], 'soccer': [94, 112, 132, 140]},
'larger': {'badminton': [13]},
'team': {'badminton': [14, 49], 'soccer': [6, 10, 28, 74, 80, 119, 138]},
'common': {'badminton': [15]},
'form': {'badminton': [16]},
'game': {'badminton': [17, 37, 127, 132, 140],
```

```
 'soccer': [24, 38, 60, 68, 123, 139, 170, 179, 195]},
'single': {'badminton': [18, 152, 154]},
'one': {'badminton': [19]},
'player': {'badminton': [20, 25, 172], 'soccer': [11, 48, 90]},
'per': {'badminton': [21, 26]},
'side': {'badminton': [22, 27, 55, 74]},
'double': {'badminton': [23, 156, 158, 160]},
'two': {'badminton': [24], 'soccer': [9, 40, 73]},
'often': {'badminton': [29]},
'casual': {'badminton': [31]},
'outdoor': {'badminton': [32]},
'activity': {'badminton': [33]},
'yard': {'badminton': [34]},
'beach': {'badminton': [35]},
'professional': {'badminton': [36], 'data science': [114]},
'rectangular': {'badminton': [39], 'soccer': [19]},
'indoor': {'badminton': [40]},
'court': {'badminton': [41, 51]},
'point': {'badminton': [42], 'soccer': [136]},
'scored': {'badminton': [43], 'soccer': [120, 131]},
'striking': {'badminton': [44]},
'landing': {'badminton': [47]},
'within': {'badminton': [48, 52],
 'soccer': [116],
 'computer science': [129],
 'data science': [71]},
'half': {'badminton': [50], 'soccer': [41]},
'set': {'badminton': [53], 'soccer': [61]},
'boundary': {'badminton': [54]},
'strike': {'badminton': [57], 'soccer': [103]},
'pass': {'badminton': [59]},
'play': {'badminton': [61, 136, 167], 'soccer': [89]},
'end': {'badminton': [62], 'soccer': [122]},
'struck': {'badminton': [64]},
'floor': {'badminton': [65]},
'ground': {'badminton': [66]},
'fault': {'badminton': [67]},
'called': {'badminton': [68], 'soccer': [21]},
'umpire': {'badminton': [69]},
'service': {'badminton': [70]},
'judge': {'badminton': [71]},
'absence': {'badminton': [72]},
'opposing': {'badminton': [73], 'soccer': [27]},
'feathered': {'badminton': [76]},
'informal': {'badminton': [77]},
'match': {'badminton': [78], 'soccer': [43]},
'plastic': {'badminton': [79]},
'projectile': {'badminton': [80]},
'fly': {'badminton': [81]},
'differently': {'badminton': [82]},
'ball': {'badminton': [83, 103], 'soccer': [17, 30, 79, 88, 105]},
'used': {'badminton': [84], 'soccer': [110]},
'many': {'badminton': [85], 'data science': [69]},
'particular': {'badminton': [87]},
'feather': {'badminton': [88]},
'create': {'badminton': [89]},
'much': {'badminton': [90, 192]},
'higher': {'badminton': [91]},
'drag': {'badminton': [92]},
'causing': {'badminton': [93]},
'decelerate': {'badminton': [95]},
'rapidly': {'badminton': [96]},
'also': {'badminton': [98, 180], 'soccer': [95], 'data science': [26]},
'high': {'badminton': [99, 165]},
'speed': {'badminton': [101, 178]},
'compared': {'badminton': [102]},
'making': {'badminton': [106]},
'fastest': {'badminton': [108]},
'world': {'badminton': [111], 'soccer': [52, 185, 188, 192, 216]},
'flight': {'badminton': [112]},
'give': {'badminton': [114]},
'distinctive': {'badminton': [116]},
'nature': {'badminton': [117]},
'certain': {'badminton': [118]},
'named': {'badminton': [121]},
'reference': {'badminton': [122]},
'feature': {'badminton': [123]},
'german': {'badminton': [124]},
'federball': {'badminton': [125]},
'literally': {'badminton': [126]},
'developed': {'badminton': [128]},
'british': {'badminton': [129]},
'india': {'badminton': [130]},
```

```
'battledore': {'badminton': [133]},
'european': {'badminton': [135], 'soccer': [198]},
'came': {'badminton': [137]},
'dominated': {'badminton': [138]},
'denmark': {'badminton': [139]},
'become': {'badminton': [141]},
'asia': {'badminton': [143]},
'debuted': {'badminton': [145]},
'summer': {'badminton': [146]},
'olympic': {'badminton': [147], 'soccer': [194]},
'four': {'badminton': [149, 162]},
'event': {'badminton': [150], 'soccer': [191, 215]},
'men': {'badminton': [151, 155], 'soccer': [187, 210]},
'woman': {'badminton': [153, 157]},
'mixed': {'badminton': [159]},
'year': {'badminton': [163]},
'later': {'badminton': [164]},
'level': {'badminton': [166], 'soccer': [174]},
'demand': {'badminton': [169]},
'excellent': {'badminton': [170]},
'fitness': {'badminton': [171]},
'require': {'badminton': [173]},
'aerobic': {'badminton': [174]},
'stamen': {'badminton': [175]},
'agility': {'badminton': [176]},
'strength': {'badminton': [177]},
'precision': {'badminton': [179]},
'technical': {'badminton': [181]},
'requiring': {'badminton': [183]},
'good': {'badminton': [184]},
'motor': {'badminton': [185]},
'coordination': {'badminton': [186]},
'development': {'badminton': [187]},
'sophisticated': {'badminton': [188]},
'movement': {'badminton': [190]},
'involving': {'badminton': [191]},
'greater': {'badminton': [193]},
'flexibility': {'badminton': [194]},
'wrist': {'badminton': [195]},
'association': {'soccer': [0, 55, 146, 161]},
'football': {'soccer': [1, 4, 56, 70, 147, 200]},
'commonly': {'soccer': [2, 109]},
'known': {'soccer': [3], 'computer science': [28]},
'soccer': {'soccer': [5, 164]},
'almost': {'soccer': [12]},
'exclusively': {'soccer': [13]},
'foot': {'soccer': [15, 93]},
'propel': {'soccer': [16]},
'around': {'soccer': [18]},
'field': {'soccer': [20],
 'computer science': [27, 48],
 'data science': [4, 70]},
'pitch': {'soccer': [22]},
'objective': {'soccer': [23]},
'score': {'soccer': [25, 76]},
'goal': {'soccer': [26, 32, 34, 77, 81, 86, 121, 130]},
'moving': {'soccer': [29]},
'beyond': {'soccer': [31]},
'line': {'soccer': [33, 87]},
'defended': {'soccer': [35]},
'opponent': {'soccer': [36]},
'traditionally': {'soccer': [37]},
'total': {'soccer': [42]},
'time': {'soccer': [44, 143]},
'minute': {'soccer': [45]},
'estimated': {'soccer': [46]},
'million': {'soccer': [47]},
'active': {'soccer': [49]},
'country': {'soccer': [50, 171]},
'territory': {'soccer': [51]},
'accordance': {'soccer': [58, 177]},
'law': {'soccer': [59, 178]},
'rule': {'soccer': [62]},
'effect': {'soccer': [63]},
'maintained': {'soccer': [65]},
'ifab': {'soccer': [66]},
'cm': {'soccer': [71]},
'circumference': {'soccer': [72]},
'compete': {'soccer': [75]},
'getting': {'soccer': [78]},
'post': {'soccer': [82]},
'bar': {'soccer': [83]},
'fully': {'soccer': [84]},
```

```
'mainly': {'soccer': [91]},
'part': {'soccer': [97]},
'body': {'soccer': [98]},
'except': {'soccer': [99]},
'hand': {'soccer': [100, 114]},
'arm': {'soccer': [101, 115]},
'control': {'soccer': [102]},
'pas': {'soccer': [104]},
'head': {'soccer': [106]},
'chest': {'soccer': [107]},
'thigh': {'soccer': [108]},
'goalkeeper': {'soccer': [111]},
'penalty': {'soccer': [117, 144]},
'area': {'soccer': [118], 'computer science': [96]},
'winner': {'soccer': [124], 'data science': [91]},
'depending': {'soccer': [125]},
'format': {'soccer': [126]},
'competition': {'soccer': [127, 176, 183, 197]},
'equal': {'soccer': [128]},
'number': {'soccer': [129]},
'result': {'soccer': [133]},
'draw': {'soccer': [134]},
'declared': {'soccer': [135]},
'awarded': {'soccer': [137]},
'go': {'soccer': [141]},
'extra': {'soccer': [142]},
'internationally': {'soccer': [145]},
'governed': {'soccer': [148]},
'fifa': {'soccer': [149, 150, 184]},
'six': {'soccer': [151]},
'continental': {'soccer': [152]},
'confederation': {'soccer': [153]},
'afc': {'soccer': [154]},
'caf': {'soccer': [155]},
'concacaf': {'soccer': [156]},
'conmebol': {'soccer': [157]},
'ofc': {'soccer': [158]},
'uefa': {'soccer': [159, 201]},
'national': {'soccer': [160]},
'fa': {'soccer': [162]},
'england': {'soccer': [163]},
'united': {'soccer': [165]},
'state': {'soccer': [166]},
'etc': {'soccer': [167]},
'responsible': {'soccer': [168]},
'managing': {'soccer': [169]},
'professionally': {'soccer': [172]},
'amateur': {'soccer': [173]},
'coordinating': {'soccer': [175]},
'prestigious': {'soccer': [180, 196]},
'senior': {'soccer': [181]},
'international': {'soccer': [182]},
'cup': {'soccer': [186, 189]},
'sporting': {'soccer': [190, 214]},
'surpassing': {'soccer': [193]},
'club': {'soccer': [199]},
'champion': {'soccer': [202, 211]},
'league': {'soccer': [203, 212]},
'attracts': {'soccer': [204]},
'extensive': {'soccer': [205]},
'television': {'soccer': [206]},
'audience': {'soccer': [207]},
'worldwide': {'soccer': [208]},
'final': {'soccer': [209]},
'annual': {'soccer': [213]},
'computer': {'computer science': [0,
  9,
  29,
  35,
  50,
  60,
  86,
  108,
  112,
  132,
  151,
  161],
 'data science': [75, 85]},
'science': {'computer science': [1, 8, 10, 36, 152, 162],
 'data science': [1, 25, 34, 39, 42, 51, 76, 78, 83, 86, 88, 96, 99, 105]},
'study': {'computer science': [2]},
'computation': {'computer science': [3, 16, 38, 42]},
'information': {'computer science': [4, 17],
```

```
 'data science': [35, 77, 87, 108]},
'automation': {'computer science': [5]},
'included': {'computer science': [6]},
'broadly': {'computer science': [7]},
'span': {'computer science': [11]},
'theoretical': {'computer science': [12], 'data science': [101]},
'discipline': {'computer science': [13, 20], 'data science': [47]},
'algorithm': {'computer science': [14, 31], 'data science': [14]},
'theory': {'computer science': [15, 18, 37, 69, 77], 'data science': [67]},
'applied': {'computer science': [19]},
'implementation': {'computer science': [23]},
'hardware': {'computer science': [24]},
'software': {'computer science': [25, 88, 95]},
'expert': {'computer science': [26]},
'scientist': {'computer science': [30], 'data science': [113]},
'data': {'computer science': [32, 81, 139, 148],
 'data science': [0, 23, 24, 38, 50, 55, 64, 82, 95, 110, 112, 122]},
'structure': {'computer science': [33]},
'central': {'computer science': [34]},
'concern': {'computer science': [39, 78, 150]},
'abstract': {'computer science': [40]},
'model': {'computer science': [41]},
'general': {'computer science': [43]},
'class': {'computer science': [44]},
'problem': {'computer science': [45]},
'solved': {'computer science': [46]},
'cryptography': {'computer science': [49]},
'security': {'computer science': [51, 58]},
'involve': {'computer science': [52]},
'studying': {'computer science': [53]},
'mean': {'computer science': [54]},
'secure': {'computer science': [55]},
'communication': {'computer science': [56]},
'preventing': {'computer science': [57]},
'vulnerability': {'computer science': [59]},
'graphic': {'computer science': [61]},
'computational': {'computer science': [62, 74], 'data science': [102]},
'geometry': {'computer science': [63]},
'address': {'computer science': [64]},
'generation': {'computer science': [65]},
'image': {'computer science': [66, 137]},
'considers': {'computer science': [70]},
'different': {'computer science': [71], 'data science': [84]},
'way': {'computer science': [72]},
'describe': {'computer science': [73]},
'process': {'computer science': [75, 121, 136, 145]},
'database': {'computer science': [76]},
'management': {'computer science': [79]},
'repository': {'computer science': [80]},
'interaction': {'computer science': [82]},
'investigates': {'computer science': [83]},
'interface': {'computer science': [84]},
'human': {'computer science': [85, 127]},
'interact': {'computer science': [87]},
'engineering': {'computer science': [89]},
'focus': {'computer science': [90]},
'principle': {'computer science': [92, 103]},
'behind': {'computer science': [93, 105]},
'developing': {'computer science': [94]},
'operating': {'computer science': [97]},
'system': {'computer science': [98, 101, 107], 'data science': [15]},
'network': {'computer science': [99]},
'embedded': {'computer science': [100]},
'investigate': {'computer science': [102]},
'complex': {'computer science': [106]},
'architecture': {'computer science': [109]},
'describes': {'computer science': [110]},
'construction': {'computer science': [111]},
'component': {'computer science': [113]},
'equipment': {'computer science': [114]},
'artificial': {'computer science': [115, 130]},
'intelligence': {'computer science': [116, 131]},
'aim': {'computer science': [119, 134, 143]},
'synthesize': {'computer science': [120]},
'environmental': {'computer science': [122]},
'adaptation': {'computer science': [123]},
'planning': {'computer science': [124]},
'found': {'computer science': [126]},
'animal': {'computer science': [128]},
'vision': {'computer science': [133]},
'understand': {'computer science': [135, 144], 'data science': [60]},
'video': {'computer science': [138]},
'natural': {'computer science': [140], 'data science': [33]},
```

```
'processing': {'computer science': [142], 'data science': [11]},
'textual': {'computer science': [146]},
'linguistic': {'computer science': [147]},
'fundamental': {'computer science': [149]},
'determining': {'computer science': [153]},
'automated': {'computer science': [154]},
'turing': {'computer science': [155], 'data science': [89]},
'award': {'computer science': [156], 'data science': [90]},
'generally': {'computer science': [157]},
'recognized': {'computer science': [158]},
'highest': {'computer science': [159]},
'distinction': {'computer science': [160]},
'interdisciplinary': {'data science': [2]},
'academic': {'data science': [3]},
'us': {'data science': [5, 65]},
'statistic': {'data science': [6, 54, 74]},
'scientific': {'data science': [7, 9, 12]},
'computing': {'data science': [8]},
'method': {'data science': [10, 46, 59]},
'visualization': {'data science': [13]},
'extract': {'data science': [16]},
'extrapolate': {'data science': [17]},
'knowledge': {'data science': [18, 29, 80, 120]},
'potentially': {'data science': [19]},
'noisy': {'data science': [20]},
'unstructured': {'data science': [22]},
'integrates': {'data science': [27]},
'domain': {'data science': [28, 32, 79]},
'underlying': {'data science': [30]},
'application': {'data science': [31]},
'technology': {'data science': [36, 109]},
'medicine': {'data science': [37]},
'multifaceted': {'data science': [40]},
'described': {'data science': [41]},
'research': {'data science': [43, 45]},
'workflow': {'data science': [48]},
'profession': {'data science': [49]},
'concept': {'data science': [52]},
'unify': {'data science': [53]},
'analysis': {'data science': [56]},
'informatics': {'data science': [57]},
'related': {'data science': [58]},
'analyze': {'data science': [61]},
'actual': {'data science': [62]},
'phenomenon': {'data science': [63]},
'technique': {'data science': [66]},
'drawn': {'data science': [68]},
'context': {'data science': [72]},
'mathematics': {'data science': [73]},
'however': {'data science': [81]},
'jim': {'data science': [92]},
'gray': {'data science': [93]},
'imagined': {'data science': [94]},
'fourth': {'data science': [97]},
'empirical': {'data science': [100]},
'asserted': {'data science': [103]},
'everything': {'data science': [104]},
'changing': {'data science': [106]},
'impact': {'data science': [107]},
'deluge': {'data science': [111]},
'creates': {'data science': [115]},
'combine': {'data science': [118]},
'statistical': {'data science': [119]},
'summarize': {'data science': [121]}}
```

Part B: Evaluation of IR Systems

Task 4: Implement Precision, Recall, and F1

- Define a set of queries and manually label relevant documents.
- Retrieve documents using keyword matching or TF-IDF ranking.
- Compute:
  - Precision
  - Recall
  - F1 Score

In [10]:
```
#code/program Task 4
# This dictionary maps 5 chosen queries to their relevant documents
queries_mapped_to_relevant_documents = {"how to start programming":["python (programming language)",
                                                                    "computer science","data science"],
```

```
                                    "what is a good sport to play":["badminton","soccer"],
                                    "how to code using computer":["python (programming language)",
                                                                  "computer science","data science"],
                                    "how to kick a ball":["soccer"],
                                    "coding project ideas":["python (programming language)",
                                                            "computer science","data science"],
                                    "best world cup goals":["soccer"],
                                    "sports that require cardio":["soccer","badminton"],
                                    "what language should i learn":["python (programming language)",
                                                                    "computer science"],
                                    "what is a fun game i can play with friends":["soccer","badminton"],
                                    "what do professional daily routines look like":["soccer","badminton",
                                                                                     "data science"]}
query_list = list(queries_mapped_to_relevant_documents.keys())
```

In [11]:
```
# Keyword Matching algorithm
def keyword(query, corpus):
    query_words = query.lower().split()
    res = []

    for title, token_list in corpus.items():
        # count the occurences of keyword (Using lemmatized documents for more accurate performance of search)
        keyword_count = 0
        for word in query_words:
            keyword_count += token_list.count(word)

        if keyword_count > 0:
            res.append(title)

    res.sort(key=lambda x:x[1], reverse=True)
    return res
```

In [12]:
```
# RESULTS OF KEYWORD SEARCH PER QUERY
for query in queries_mapped_to_relevant_documents.keys():
    print(f"{query}: {keyword(query, lemma_tokenized_documents)}")
```

```
how to start programming: ['python (programming language)', 'computer science', 'data science']
what is a good sport to play: ['soccer', 'badminton']
how to code using computer: ['python (programming language)', 'computer science', 'badminton', 'data science']
how to kick a ball: ['soccer', 'badminton']
coding project ideas: []
best world cup goals: ['soccer', 'badminton']
sports that require cardio: ['badminton']
what language should i learn: ['python (programming language)', 'computer science', 'badminton']
what is a fun game i can play with friends: ['soccer', 'badminton']
what do professional daily routines look like: ['badminton', 'data science']
```

In [13]:
```
#### PRECISION, RECALL AND F1 SCORE

# "how to start programming"
Q1_precision = 3/3
Q1_recall = 3/3
Q1_F1 = 2*(Q1_precision*Q1_recall) / (Q1_precision+Q1_recall)

# "what is a good sport to play"
Q2_precision = 2/2
Q2_recall = 2/2
Q2_F1 = 2*(Q2_precision*Q2_recall) / (Q2_precision+Q2_recall)

# "how to code using computer"
Q3_precision = 3/4
Q3_recall = 3/3
Q3_F1 = 2*(Q3_precision*Q3_recall) / (Q3_precision+Q3_recall)

# "how to kick a ball"
Q4_precision = 1/2
Q4_recall = 1/1
Q4_F1 = 2*(Q4_precision*Q4_recall) / (Q4_precision+Q4_recall)

# "coding project ideas"
Q5_precision = 0/1
Q5_recall = 0/3
Q5_F1 = 2*(Q5_precision*Q5_recall) / (Q5_precision+Q5_recall+1)

# best world cup goals
Q6_precision = 1/2
Q6_recall = 1/1
Q6_F1 = 2*(Q6_precision*Q6_recall) / (Q6_precision+Q6_recall)

# sports that require cardio
Q7_precision = 1/1
Q7_recall = 1/2
Q7_F1 = 2*(Q7_precision*Q7_recall) / (Q7_precision+Q7_recall)
```

```
# what language should i learn
Q8_precision = 2/3
Q8_recall = 2/2
Q8_F1 = 2*(Q8_precision*Q8_recall) / (Q8_precision+Q8_recall)

# what is a fun game i can play with friends
Q9_precision = 2/2
Q9_recall = 2/2
Q9_F1 = 2*(Q9_precision*Q9_recall) / (Q9_precision+Q9_recall)

# what do professional daily routines look like
Q10_precision = 2/2
Q10_recall = 2/3
Q10_F1 = 2*(Q10_precision*Q10_recall) / (Q10_precision+Q10_recall)

# Average metrics
avg_precision = (Q1_precision + Q2_precision + Q3_precision + Q4_precision + Q5_precision +
                 Q6_precision + Q7_precision + Q8_precision + Q9_precision + Q10_precision) / 10
avg_recall = (Q1_recall + Q2_recall + Q3_recall + Q4_recall + Q5_recall + Q6_recall + Q7_recall +
              Q8_recall + Q9_recall + Q10_recall) / 10
avg_F1 = (Q1_F1 + Q2_F1 + Q3_F1 + Q4_F1 + Q5_F1 + Q6_F1 + Q7_F1 + Q8_F1 + Q9_F1 + Q10_F1) / 10

print(f"Q1-{query_list[0]}: precision = {Q1_precision:.3f}, recall = {Q1_recall:.3f}, f1 = {Q1_F1:.3f}")
print(f"Q2-{query_list[1]}: precision = {Q2_precision:.3f}, recall = {Q2_recall:.3f}, f1 = {Q2_F1:.3f}")
print(f"Q3-{query_list[2]}: precision = {Q3_precision:.3f}, recall = {Q3_recall:.3f}, f1 = {Q3_F1:.3f}")
print(f"Q4-{query_list[3]}: precision = {Q4_precision:.3f}, recall = {Q4_recall:.3f}, f1 = {Q4_F1:.3f}")
print(f"Q5-{query_list[4]}: precision = {Q5_precision:.3f}, recall = {Q5_recall:.3f}, f1 = {Q5_F1:.3f}")
print(f"Q6-{query_list[5]}: precision = {Q6_precision:.3f}, recall = {Q6_recall:.3f}, f1 = {Q6_F1:.3f}")
print(f"Q7-{query_list[6]}: precision = {Q7_precision:.3f}, recall = {Q7_recall:.3f}, f1 = {Q7_F1:.3f}")
print(f"Q8-{query_list[7]}: precision = {Q8_precision:.3f}, recall = {Q8_recall:.3f}, f1 = {Q8_F1:.3f}")
print(f"Q9-{query_list[8]}: precision = {Q9_precision:.3f}, recall = {Q9_recall:.3f}, f1 = {Q9_F1:.3f}")
print(f"Q10-{query_list[9]}: precision = {Q10_precision:.3f}, recall = {Q10_recall:.3f}, f1 = {Q10_F1:.3f}")
print(f"AVERAGES: precision = {avg_precision:.3f}, recall = {avg_recall:.3f}, f1 = {avg_F1:.3f}")
```

```
Q1-how to start programming: precision = 1.000, recall = 1.000, f1 = 1.000
Q2-what is a good sport to play: precision = 1.000, recall = 1.000, f1 = 1.000
Q3-how to code using computer: precision = 0.750, recall = 1.000, f1 = 0.857
Q4-how to kick a ball: precision = 0.500, recall = 1.000, f1 = 0.667
Q5-coding project ideas: precision = 0.000, recall = 0.000, f1 = 0.000
Q6-best world cup goals: precision = 0.500, recall = 1.000, f1 = 0.667
Q7-sports that require cardio: precision = 1.000, recall = 0.500, f1 = 0.667
Q8-what language should i learn: precision = 0.667, recall = 1.000, f1 = 0.800
Q9-what is a fun game i can play with friends: precision = 1.000, recall = 1.000, f1 = 1.000
Q10-what do professional daily routines look like: precision = 1.000, recall = 0.667, f1 = 0.800
AVERAGES: precision = 0.742, recall = 0.817, f1 = 0.746
```

Task 5: Precision@k and MAP

- Rank documents using cosine similarity.
- Evaluate Precision@5 and MAP across multiple queries.

In [14]:

```
#code/program Task 5
import math
# COSINE SIMILARITY
def cosine_similarity(query, corpus):
    # Find Term frequency for each document and term, including query
    freq_dict  = {}
    document_titles = list(corpus.keys())

    for idx, title in enumerate(document_titles):
        list_words = corpus[title]
        for word in list_words:
            if word not in freq_dict:
                freq_dict[word] = [0] * len(corpus)
            freq_dict[word][idx] += 1

    # Cosine Similarity
    # prepare document and query vectors
    vocabulary = freq_dict.keys()

    document_vectors = []
    for title in document_titles:
        words = corpus[title]
        vector = [words.count(term) for term in vocabulary]
        document_vectors.append(vector)

    query_vector = [query.split().count(term) for term in vocabulary]

    # 5b: calculate similarity between query and each document
    cosine_similarities = []
    for idx,doc_vec in enumerate(document_vectors):
```

```
        dot_prod = sum(x*y for x,y in zip(doc_vec, query_vector))
        doc_norm = math.sqrt(sum(x*x for x in doc_vec))
        query_norm = math.sqrt(sum(x*x for x in query_vector))
        # Division by 0 error handling
        if doc_norm == 0 or query_norm == 0:
            cs = 0
        else:
            cs = round(dot_prod / (doc_norm*query_norm),3)
        cosine_similarities.append((document_titles[idx],cs))

    cosine_similarities.sort(key=lambda x: x[1], reverse=True)

    return cosine_similarities
```

In [15]:
```
# Cosine Similarity on first query
cosine_similarity("how to start programming", lemma_tokenized_documents)
```

Out[15]:
```
[('python (programming language)', 0.499),
 ('computer science', 0.048),
 ('data science', 0.045),
 ('badminton', 0.0),
 ('soccer', 0.0)]
```

In [16]:
```
# Precision@5 for all queries
def precision_at_5(relevant_docs_for_query, search_results, k=5):
    topk = search_results[:k]

    topk_docs = [doc for doc, score in topk]

    rel_topk = sum(1 for doc in topk_docs if doc in relevant_docs_for_query)

    precision_k = rel_topk / k

    return precision_k

for query in queries_mapped_to_relevant_documents.keys():
    relevant_docs_for_query = queries_mapped_to_relevant_documents[query]
    print(f"{query} PRECISION@5: {precision_at_5(relevant_docs_for_query,
                                    cosine_similarity(query, lemma_tokenized_documents))}")
```

```
how to start programming PRECISION@5: 0.6
what is a good sport to play PRECISION@5: 0.4
how to code using computer PRECISION@5: 0.6
how to kick a ball PRECISION@5: 0.2
coding project ideas PRECISION@5: 0.6
best world cup goals PRECISION@5: 0.2
sports that require cardio PRECISION@5: 0.4
what language should i learn PRECISION@5: 0.4
what is a fun game i can play with friends PRECISION@5: 0.4
what do professional daily routines look like PRECISION@5: 0.6
```

In [17]:
```
# AVERAGE PRECISION
def avg_precision(relevant_docs_for_query, search_results):
    if len(relevant_docs_for_query) == 0:
        return 0

    precision_list = []
    relevant_doc_count = 0

    for idx, (doc, score) in enumerate(search_results):
        if doc in relevant_docs_for_query:
            relevant_doc_count += 1
            precision_at_k = relevant_doc_count / (idx+1)
            precision_list.append(precision_at_k)

    if len(precision_list) == 0:
        return 0

    avg_precision = sum(precision_list) / len(relevant_docs_for_query)
    return avg_precision

# MAP
APs = []
for query in queries_mapped_to_relevant_documents.keys():
    relevant_docs_for_query = queries_mapped_to_relevant_documents[query]
    search_results = cosine_similarity(query, lemma_tokenized_documents)
    average_precision = avg_precision(relevant_docs_for_query, search_results)
    APs.append(average_precision)

MAP = sum(APs) / len(APs)
print(f"MAP SCORE = {MAP}")
```

```
MAP SCORE = 0.945
```

Part C: Relevance Feedback and Query Expansion

Task 6: Pseudo Relevance Feedback

- Assume top-k retrieved documents are relevant.
- Extract frequent terms and expand the query.
- Re-run retrieval and compare performance metrics.

In [18]:
```
#code/program Task 6
# Using three queries for PRF
queries_used_3 = {"how to kick a ball":["soccer"],
                  "how to code using computer":["python (programming language)",
                                                "computer science","data science"],
                  "what do professional daily routines look like":["soccer","badminton","data science"]}
query_3_list = list(queries_used_3.keys())

for query in queries_used_3.keys():
    relevant_docs = queries_used_3[query]
    # Initial values
    init_res = cosine_similarity(query, lemma_tokenized_documents)

    # Using top 3 docs as relevant
    top3 = init_res[:3]
    terms = []
    for doc, _ in top3:
        terms.extend(lemma_tokenized_documents[doc])

    # get top 5 new terms
    query_tokens = set(query.split())
    term_frequency = {}
    for term in terms:
        if term not in query_tokens:
            term_frequency[term] = term_frequency.get(term, 0) + 1

    # sort by frqeuncy
    sorted_terms = sorted(term_frequency.items(), key=lambda x:x[1], reverse=True)
    new_terms = [term for term, freq in sorted_terms[:5]]
    expanded_query = query + " " +" ".join(new_terms)
    print(expanded_query)
```

```
how to kick a ball game sport team shuttlecock played
how to code using computer science data programming language python
what do professional daily routines look like science data sport shuttlecock programming
```

In [19]:
```
def recall_at_3(relevant_docs_for_query, search_results, k=3):
    topk = search_results[:k]
    topk_docs = [doc for doc, score in topk]
    rel_topk = sum(1 for doc in topk_docs if doc in relevant_docs_for_query)
    if len(relevant_docs_for_query) == 0:
        recall_k = 0
    else:
        recall_k = rel_topk / len(relevant_docs_for_query)
    return recall_k
```

In [20]:
```
queries_used_ref_3 = {"how to kick a ball game sport team shuttlecock played":["soccer"],
        "how to code using computer science data programming language python":["python (programming language)",
                                                                               "computer science","data science"],
        "what do professional daily routines look like science data sport shuttlecock programming":["soccer",
                                                                                "badminton","data science"]}
# USING PRECISION@3 BY MAKING K=3
query_3_list = list(queries_used_3.keys())
query_3_ref_list = list(queries_used_ref_3.keys())

print(f"{query_3_list[0]} P@3:{precision_at_5(queries_used_3[query_3_list[0]],
                                    cosine_similarity(query_3_list[0], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[0]} P@3:{precision_at_5(queries_used_ref_3[query_3_ref_list[0]],
                                    cosine_similarity(query_3_ref_list[0], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_3_list[1]} P@3:{precision_at_5(queries_used_3[query_3_list[1]],
                                    cosine_similarity(query_3_list[1], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[1]} P@3:{precision_at_5(queries_used_ref_3[query_3_ref_list[1]],
                                    cosine_similarity(query_3_ref_list[1], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_3_list[2]} P@3:{precision_at_5(queries_used_3[query_3_list[2]],
                                    cosine_similarity(query_3_list[2], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[2]} P@3:{precision_at_5(queries_used_ref_3[query_3_ref_list[2]],
                                    cosine_similarity(query_3_ref_list[2], lemma_tokenized_documents),k=3)}")
```

```
how to kick a ball P@3:0.3333333333333333
how to kick a ball game sport team shuttlecock played P@3:0.3333333333333333

how to code using computer P@3:1.0
how to code using computer science data programming language python P@3:1.0

what do professional daily routines look like P@3:0.6666666666666666
what do professional daily routines look like science data sport shuttlecock programming P@3:0.6666666666666666
```

In [21]:
```python
# RECALL@3
print(f"{query_3_list[0]} R@3:{recall_at_3(queries_used_3[query_3_list[0]],
                                    cosine_similarity(query_3_list[0], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[0]} R@3:{recall_at_3(queries_used_ref_3[query_3_ref_list[0]],
                                    cosine_similarity(query_3_ref_list[0], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_3_list[1]} R@3:{recall_at_3(queries_used_3[query_3_list[1]],
                                    cosine_similarity(query_3_list[1], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[1]} R@3:{recall_at_3(queries_used_ref_3[query_3_ref_list[1]],
                                    cosine_similarity(query_3_ref_list[1], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_3_list[2]} R@3:{recall_at_3(queries_used_3[query_3_list[2]],
                                    cosine_similarity(query_3_list[2], lemma_tokenized_documents),k=3)}")
print(f"{query_3_ref_list[2]} R@3:{recall_at_3(queries_used_ref_3[query_3_ref_list[2]],
                                    cosine_similarity(query_3_ref_list[2], lemma_tokenized_documents),k=3)}")
```

```
how to kick a ball R@3:1.0
how to kick a ball game sport team shuttlecock played R@3:1.0

how to code using computer R@3:1.0
how to code using computer science data programming language python R@3:1.0

what do professional daily routines look like R@3:0.6666666666666666
what do professional daily routines look like science data sport shuttlecock programming R@3:0.6666666666666666
```

Task 7: Query Expansion Techniques

- Apply synonym expansion using WordNet.
- Use local analysis to extract terms from top-ranked documents.
- Compare original vs. expanded query results.

In [22]:
```python
#code/program Task 7
# SYNONYM EXAPNSION USING WORDNET
from nltk.corpus import wordnet

# We will once again use these three queries
queries_used_3 = {
    "how to kick a ball":["soccer"],
    "how to code using computer":["python (programming language)","computer science","data science"],
    "what do professional daily routines look like":["soccer","badminton","data science"]
}

def query_synonym_expansion(query):
    tokens = query.lower().split()
    # Remove stopwords, and tokenize
    tokens = [token for token in tokens if token not in stopwords]
    expanded_query = []

    for token in tokens:
        expanded_query.append(token)

        synonyms = []
        for set_of_synonyms in wordnet.synsets(token):
            for lemma in set_of_synonyms.lemmas():
                synonym_words = lemma.name().lower()
                if synonym_words != token:
                    synonyms.append(synonym_words)

        expanded_query.extend(list(set(synonyms)))
    return " ".join(expanded_query)
```

In [23]:
```python
synonym_expansion = dict()
for query in queries_used_3.keys():
    synonyms = query_synonym_expansion(query)
    synonym_expansion[synonyms] = queries_used_3[query]
synonym_expansion
```

Out[23]:
```
{'kick kicking flush quetch plain thrill bitch kick_back rush charge bang boot recoil gripe sound_off kvetch gi
ve_up complain beef squawk ball musket_ball lucille_ball orb formal orchis clod bollock glob chunk ballock egg
lump nut testicle globe clump testis': ['soccer'],
 'code encipher codification encrypt cypher inscribe write_in_code computer_code cipher using apply victimizati
on victimisation expend practice utilise habituate use employ exploitation utilize computer computing_machine e
stimator calculator computing_device reckoner electronic_computer figurer information_processing_system data_pr
ocessor': ['python (programming language)',
  'computer science',
  'data science'],
 'professional pro professional_person master daily day-to-day day-by-day casual everyday day_by_day day-after-
day routines modus_operandi act procedure subprogram function subroutine turn routine bit number look take_care
see wait spirit aspect depend face appear feeling reckon flavour calculate bet tone smell count expression look
ing facial_expression feel seem attend looking_at front expect search await flavor like alike comparable wish t
he_like ilk similar the_likes_of care same corresponding': ['soccer',
  'badminton',
  'data science']}
```

In [24]:
```
# LOCAL ANALYSIS
local_analysis = dict()
for query in queries_used_3.keys():
    relevant_docs = queries_used_3[query]
    # Initial values
    init_res = cosine_similarity(query, lemma_tokenized_documents)

    # Using top 3 docs as relevant
    top3 = init_res[:3]
    terms = []
    for doc, _ in top3:
        terms.extend(lemma_tokenized_documents[doc])

    # get top 5 new terms
    query_tokens = set(query.split())
    term_frequency = {}
    for term in terms:
        if term not in query_tokens:
            term_frequency[term] = term_frequency.get(term, 0) + 1

    # sort by frqeuncy
    sorted_terms = sorted(term_frequency.items(), key=lambda x:x[1], reverse=True)
    new_terms = [term for term, freq in sorted_terms[:5]]
    expanded_query = query + " " +" ".join(new_terms)
    local_analysis[expanded_query] = queries_used_3[query]
local_analysis
```

Out[24]:
```
{'how to kick a ball game sport team shuttlecock played': ['soccer'],
 'how to code using computer science data programming language python': ['python (programming language)',
  'computer science',
  'data science'],
 'what do professional daily routines look like science data sport shuttlecock programming': ['soccer',
  'badminton',
  'data science']}
```

In [25]:
```
# USING PRECISION@3 BY MAKING K=3
queries_syn = list(synonym_expansion.keys())
query_local = list(local_analysis.keys())

print("QUERY 1 RESULTS:")
print(f"{queries_syn[0]} PRECISION@3: {precision_at_5(synonym_expansion[queries_syn[0]],
                                        cosine_similarity(queries_syn[0], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[0]} PRECISION@3: {precision_at_5(local_analysis[query_local[0]],
                                        cosine_similarity(query_local[0], lemma_tokenized_documents),k=3)}")
print()
print("QUERY 2 RESULTS:")
print(f"{queries_syn[1]} PRECISION@3: {precision_at_5(synonym_expansion[queries_syn[1]],
                                        cosine_similarity(queries_syn[1], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[1]} PRECISION@3: {precision_at_5(local_analysis[query_local[1]],
                                        cosine_similarity(query_local[1], lemma_tokenized_documents),k=3)}")
print()
print("QUERY 3 RESULTS:")
print(f"{queries_syn[2]} PRECISION@3: {precision_at_5(synonym_expansion[queries_syn[2]],
                                        cosine_similarity(queries_syn[2], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[2]} PRECISION@3: {precision_at_5(local_analysis[query_local[2]],
                                        cosine_similarity(query_local[2], lemma_tokenized_documents),k=3)}")
```

```
QUERY 1 RESULTS:
kick kicking flush quetch plain thrill bitch kick_back rush charge bang boot recoil gripe sound_off kvetch give_
up complain beef squawk ball musket_ball lucille_ball orb formal orchis clod bollock glob chunk ballock egg lump
nut testicle globe clump testis PRECISION@3: 0.3333333333333333

how to kick a ball game sport team shuttlecock played PRECISION@3: 0.3333333333333333

QUERY 2 RESULTS:
code encipher codification encrypt cypher inscribe write_in_code computer_code cipher using apply victimization
victimisation expend practice utilise habituate use employ exploitation utilize computer computing_machine estim
ator calculator computing_device reckoner electronic_computer figurer information_processing_system data_process
or PRECISION@3: 0.6666666666666666

how to code using computer science data programming language python PRECISION@3: 1.0

QUERY 3 RESULTS:
professional pro professional_person master daily day-to-day day-by-day casual everyday day_by_day day-after-day
routines modus_operandi act procedure subprogram function subroutine turn routine bit number look take_care see
wait spirit aspect depend face appear feeling reckon flavour calculate bet tone smell count expression looking f
acial_expression feel seem attend looking_at front expect search await flavor like alike comparable wish the_lik
e ilk similar the_likes_of care same corresponding PRECISION@3: 0.6666666666666666

what do professional daily routines look like science data sport shuttlecock programming PRECISION@3: 0.66666666
66666666
```

In [26]:
```python
# RECALL @ 3
print("QUERY 1 RESULTS:")
print(f"{queries_syn[0]} RECALL@3: {recall_at_3(synonym_expansion[queries_syn[0]],
                                        cosine_similarity(queries_syn[0], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[0]} RECALL@3: {recall_at_3(local_analysis[query_local[0]],
                                        cosine_similarity(query_local[0], lemma_tokenized_documents),k=3)}")
print()
print("QUERY 2 RESULTS:")
print(f"{queries_syn[1]} RECALL@3: {recall_at_3(synonym_expansion[queries_syn[1]],
                                        cosine_similarity(queries_syn[1], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[1]} RECALL@3: {recall_at_3(local_analysis[query_local[1]],
                                        cosine_similarity(query_local[1], lemma_tokenized_documents),k=3)}")
print()
print("QUERY 3 RESULTS:")
print(f"{queries_syn[2]} RECALL@3: {recall_at_3(synonym_expansion[queries_syn[2]],
                                        cosine_similarity(queries_syn[2], lemma_tokenized_documents),k=3)}")
print()
print(f"{query_local[2]} RECALL@3: {recall_at_3(local_analysis[query_local[2]],
                                        cosine_similarity(query_local[2], lemma_tokenized_documents),k=3)}")
```

```
QUERY 1 RESULTS:
kick kicking flush quetch plain thrill bitch kick_back rush charge bang boot recoil gripe sound_off kvetch give_
up complain beef squawk ball musket_ball lucille_ball orb formal orchis clod bollock glob chunk ballock egg lump
nut testicle globe clump testis RECALL@3: 1.0

how to kick a ball game sport team shuttlecock played RECALL@3: 1.0

QUERY 2 RESULTS:
code encipher codification encrypt cypher inscribe write_in_code computer_code cipher using apply victimization
victimisation expend practice utilise habituate use employ exploitation utilize computer computing_machine estim
ator calculator computing_device reckoner electronic_computer figurer information_processing_system data_process
or RECALL@3: 0.6666666666666666

how to code using computer science data programming language python RECALL@3: 1.0

QUERY 3 RESULTS:
professional pro professional_person master daily day-to-day day-by-day casual everyday day_by_day day-after-day
routines modus_operandi act procedure subprogram function subroutine turn routine bit number look take_care see
wait spirit aspect depend face appear feeling reckon flavour calculate bet tone smell count expression looking f
acial_expression feel seem attend looking_at front expect search await flavor like alike comparable wish the_lik
e ilk similar the_likes_of care same corresponding RECALL@3: 0.6666666666666666

what do professional daily routines look like science data sport shuttlecock programming RECALL@3: 0.66666666666
66666
```

Reflection Questions

Please answer the following questions in markdown cells following each question:

Q1: How did normalization affect retrieval performance?

Normalization affects retreival performance heavily. Firstly, we lowercased which merged many words together in our token list (for example "Badminton" and "badminton" weren't seperate tokens after lowercasing, which helps lower our index size. Secondly, there were many useless tokens in our tokenized lists which were punctuations such as "." and ",". Removing these are good since they do not help

in document retrieval so keeping them in our tokens list is redundant, removing them lowers our space and improves retrieval performance. Also we handled unicode which is useful in retrieval performance, because if it is not handled properly we may miss relevant documents when using retrieval algorithms. We prefer to keep a universal representation of our characters so our retrival can be consistent.

Q2: What are the trade-offs between stemming and lemmatization?

While stemming and lemmtizing our documents. It was noticeable that stemming is more imprecise than lemmatization. For example, some words such as "doubles" were stemmed to be "doubl" which is not a correct word. A lot of words such as double which ended with "e" actually got the "e" cut off in stemming, which is not ideal. It was also noticeable that the lemmatization algorithm is more accurate, but the trade-off was that it was slower when ran. This makes sense because it has POS awareness. Since our corpus was small, it was not a big issue, but may be a bigg issue when a large corpus for search retreival is used.

Q3: How did relevance feedback improve recall?

Normally, the recall should improve after relevance feedback. But due to the small corpus set, adding additional terms in the query may not change anything due to the fact that the queries were probably already pretty accurate when calculating the cosine similarity due to the small sample size. Nowrmally however, the recall and precision using relevance feedback should increase (improve).

Q4: Which expansion technique yielded the best precision?

For our scenario and documents, local analysis gave the best results. We can see on query 2 that both precision and recall increased, and did better than our base query and our synonym expansion. This is most likely due to the fact that for synonym expansion, our query and documents aren't very large so not many synonyms of existing words in the document can be found. Local analysis works well in this case since we already assume that the top-k are relevant and gets higher weighted terms from those documents. Overall, both expansion techniques are solid but in this case, local analysis is better.

Evaluation Criteria

Component	Points
Preprocessing implementation	20
Inverted index construction	15
Evaluation metrics	20
Relevance feedback & QE	25
Reflection answers	20
Total	**100**