1 Front page - DAT640-2022 exam info

Exam
DAT640 - Information Retrieval and Text Mining
2022 Autumn

DATE AND TIME

- Exam starts: 21.11.2020, 09:00
- Exam closes: 21.11.2020, 13:00

You can see how much time you have left on the exam on the top of the screen. Exam answers that are submitted after the time has expired will not be accepted.

AIDS
All aids are permitted. This includes both written and printed material as well as files and programs on your own device.

IMPORTANT CONTACTS
If you need help during the exam, you can call one of the phone numbers below. This applies if you need clarifications from the course responsible or administrative support.

- Lecturer: Ivica Kostric, tlf. 90 80 09 02
- Administrative support tlf. 51 83 31 26

WITHDRAW DURING THE EXAM
If you wish to withdraw from the exam, you must do so by choosing “deliver blank” in the top right menu and follow the instructions.

HANDING IN
The exam will automatically close for uploading when the time is up.

Note: In case something goes wrong in Inspera, such that you are unable to submit your exam, you must contact administrative support immediately.

QUESTIONS AND GRADING
The exam contains 27 questions in total.

- There are multiple choice questions or sub-questions, where there is -1 point for each wrong answer (no answer is 0 points). These are explicitly indicated.

Total points: 100

Grading (standard scale)

- 0-39: F
- 40-49: E
- 50-59: D
- 60-79: C
- 80-89: B
- 90-100: A

For all computations, provide numbers rounded to 3 digits (e.g., 0.7, 0.25, 0.333).

GOOD LUCK!

If you have any comments about the exam, write them here

Format

Words: 0

Maximum marks: 0

2 Normalization

Normalize the following vector of values using min-max normalization: <2, 12, 7>.
Write each of the three (normalized) values in the corresponding input cells. *(3 points)*

< □ , □ , □ >

Maximum marks: 3

3 Classification

Assume a multiclass classification problem with 3 categories that is decomposed into a binary classification problem. The one-against-one and one-against-rest strategies in this case require the same number of binary classifiers. *(2 points, -1 if incorrect)*

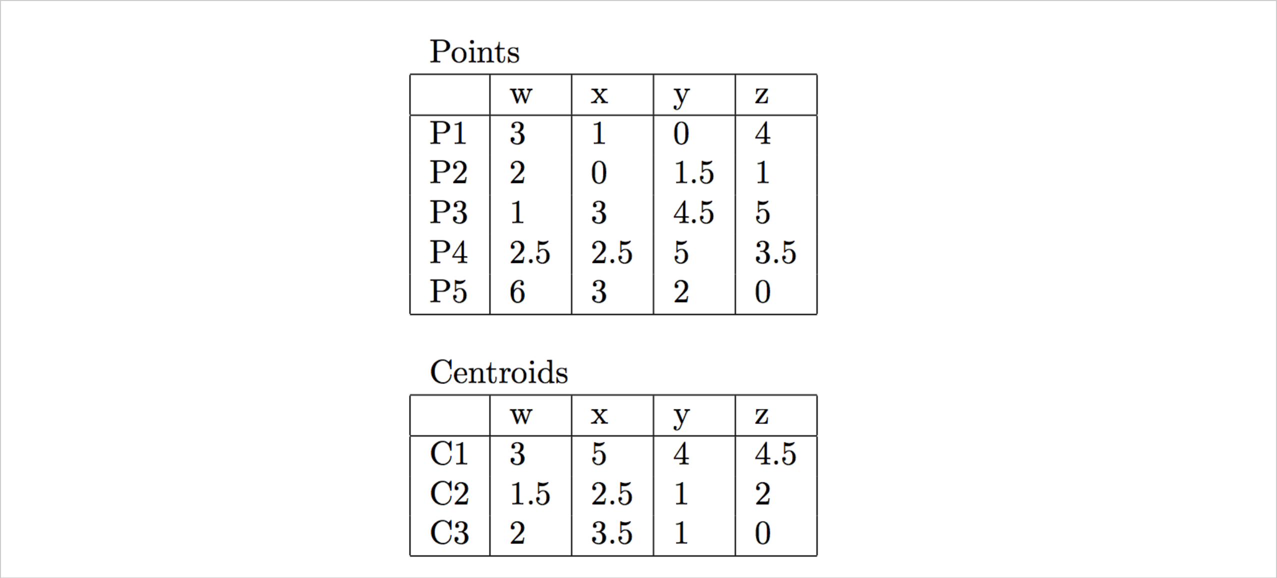
Select one alternative:

○ True

○ False

Maximum marks: 2

4 Clustering



Points

	w	x	y	z
P1	3	1	0	4
P2	2	0	1.5	1
P3	1	3	4.5	5
P4	2.5	2.5	5	3.5
P5	6	3	2	0

Centroids

	w	x	y	z
C1	3	5	4	4.5
C2	1.5	2.5	1	2
C3	2	3.5	1	0

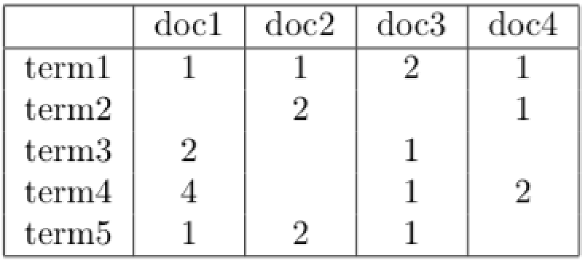
You are given five 4-dimensional data points and three initial cluster centroids.
Perform the first iteration of K-means clustering using the Euclidean distance. *(3 points)*

Which cluster the following data points get assigned to?
(In case a data point is of equal distance to more than one cluster, pick the cluster with the lowest index.)

- P1 is assigned to Select alternative (C1, C2, C3)
- P3 is assigned to Select alternative (C1, C2, C3)
- P5 is assigned to Select alternative (C1, C2, C3)

Maximum marks: 3

5 Retrieval



	doc1	doc2	doc3	doc4
term1	1	1	2	1
term2		2		1
term3	2		1	
term4	4		1	2
term5	1	2	1	

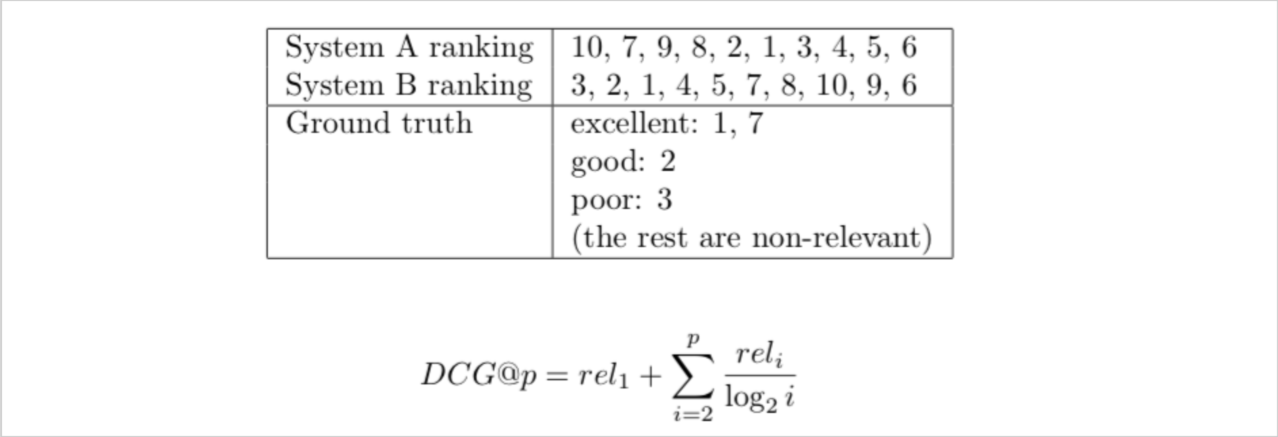
A term-document matrix is given above.
We use a Language Modeling retrieval method with Dirichlet smoothing and the smoothing parameter (mu) set to 6.

Answer the following questions: *(5x2 points)*

- What is the probability of term5 in the empirical language model of doc1?
- What is the probability of term4 in the background language model?
- What is the probability of term2 in the (smoothed) language model of doc3?
- Which term has the lowest probability in the (smoothed) language model of doc2?
  Select alternative (term1, term2, term3, term4, term5)
- Which is the top scoring document for the query ``term1 term3 term5''?
  Select alternative (doc1, doc2, doc3, doc4)

Maximum marks: 10

6 Retrieval evaluation



System A ranking	10, 7, 9, 8, 2, 1, 3, 4, 5, 6
System B ranking	3, 2, 1, 4, 5, 7, 8, 10, 9, 6
Ground truth	excellent: 1, 7 good: 2 poor: 3 (the rest are non-relevant)

$$DCG@p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

Evaluate two retrieval systems in terms of DCG@5 and NDCG@10 on a given search query.
(4x2 points)

The table above contains the rankings generated by the two systems as well as the ground truth. Documents are judged on a 4-point scale: non-relevant (0), poor (1), good (2), excellent (3). The DCG formula to be used is also shown for your reference.

- What is DCG@5 for System A? □
- What is DCG@5 for System B? □
- What is NDCG@10 for System A? □
- What is NDCG@10 for System B? □

Maximum marks: 8

7 Entity retrieval

Which of the following statements about predicate folding is *false*? *(3 points, -1 if incorrect)*

Select one alternative:

○ It helps to preserve the semantics of RDF data.

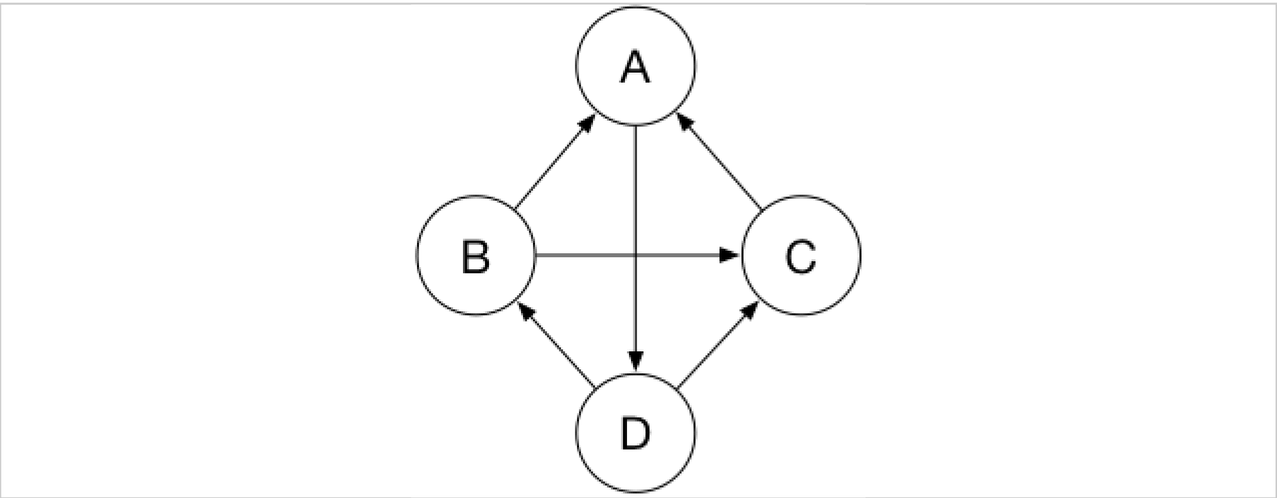
○ A single RDF triple may be mapped to multiple fields.

○ It helps to deal with data sparsity.

○ It is not needed when there are only a handful of different predicates.

Maximum marks: 3

8 PageRank



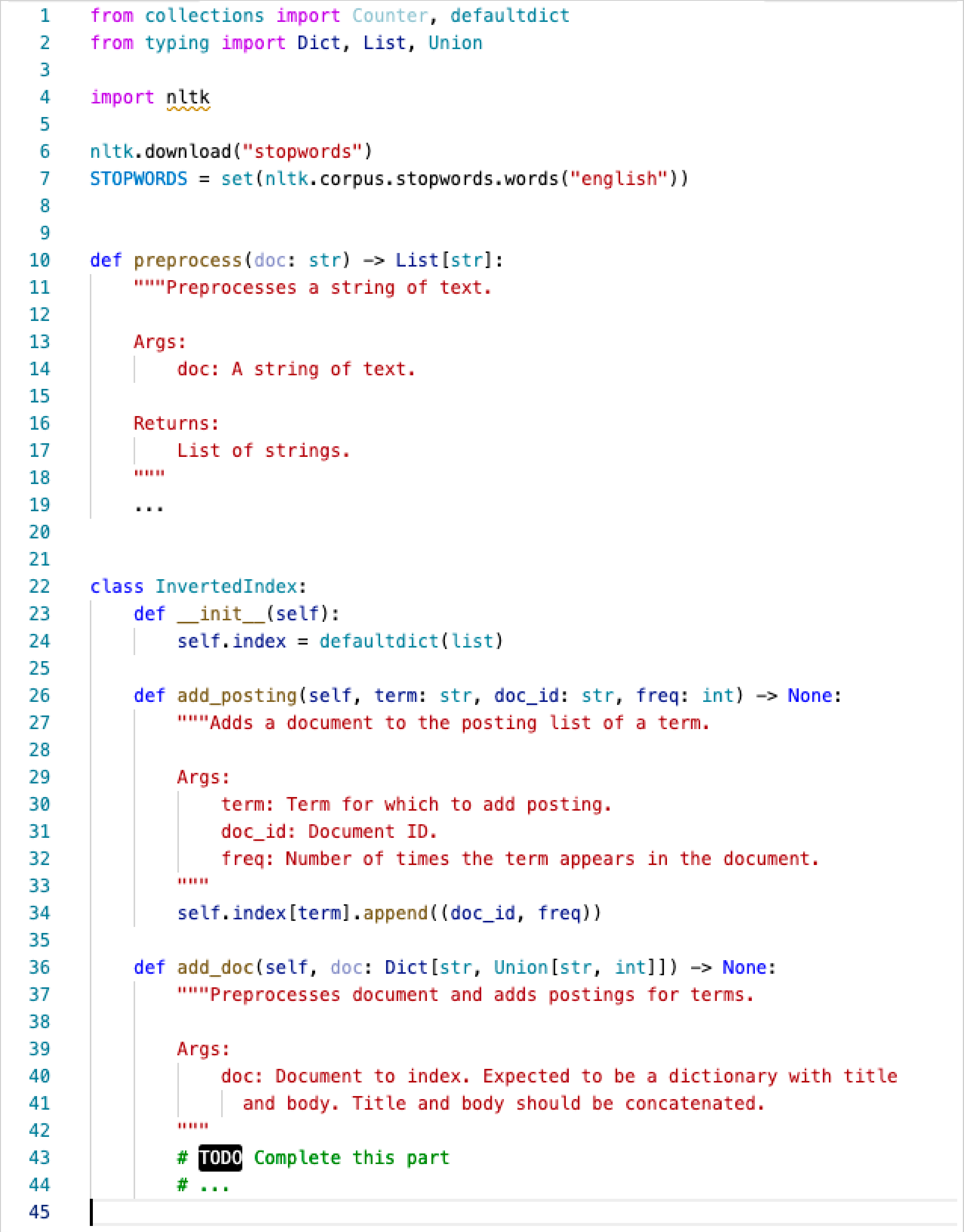
Compute the PageRank values for the following graph for two iterations. *(8x1 point)*

The probability of a random jump (i.e., the parameter q) is 0.2.

	Iteration 0	Iteration 1	Iteration 2
A	0.25		
B	0.25		
C	0.25		
D	0.25		

Maximum marks: 8

9 Indexing



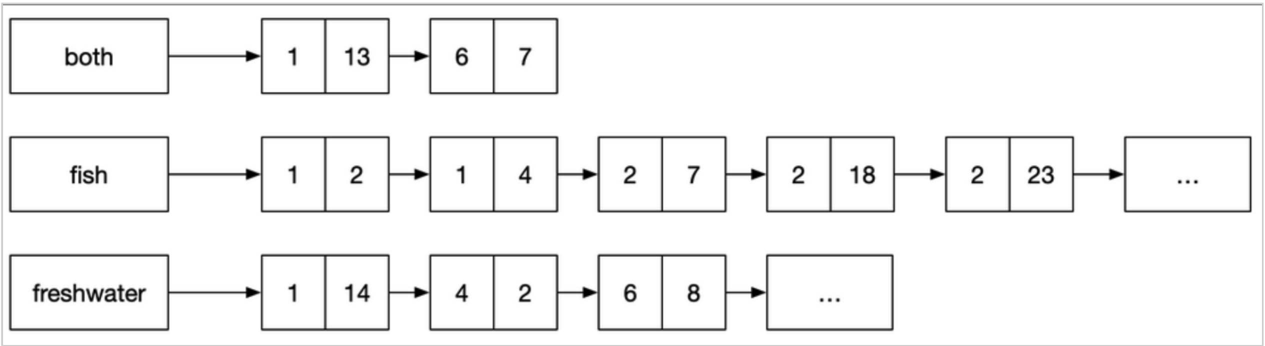
```python
from collections import Counter, defaultdict
from typing import Dict, List, Union

import nltk

nltk.download("stopwords")
STOPWORDS = set(nltk.corpus.stopwords.words("english"))


def preprocess(doc: str) -> List[str]:
    """Preprocesses a string of text.

    Args:
        doc: A string of text.

    Returns:
        List of strings.
    """
    ...


class InvertedIndex:
    def __init__(self):
        self.index = defaultdict(list)

    def add_posting(self, term: str, doc_id: str, freq: int) -> None:
        """Adds a document to the posting list of a term.

        Args:
            term: Term for which to add posting.
            doc_id: Document ID.
            freq: Number of times the term appears in the document.
        """
        self.index[term].append((doc_id, freq))

    def add_doc(self, doc: Dict[str, Union[str, int]]) -> None:
        """Preprocesses document and adds postings for terms.

        Args:
            doc: Document to index. Expected to be a dictionary with title
              and body. Title and body should be concatenated.
        """
        # TODO Complete this part
        # ...

```

Write the body of the **add_doc()** method that adds a document to an inverted index. You can use the method **preprocess()** if necessary. *(4 points)*

Fill in your answer here

Maximum marks: 4

10 Indexing



You are given an excerpt from an inverted index. Select all statements that apply for this kind of index. *(3 points)*

Select one or more alternatives:

□ The index is suitable for SDM retrieval

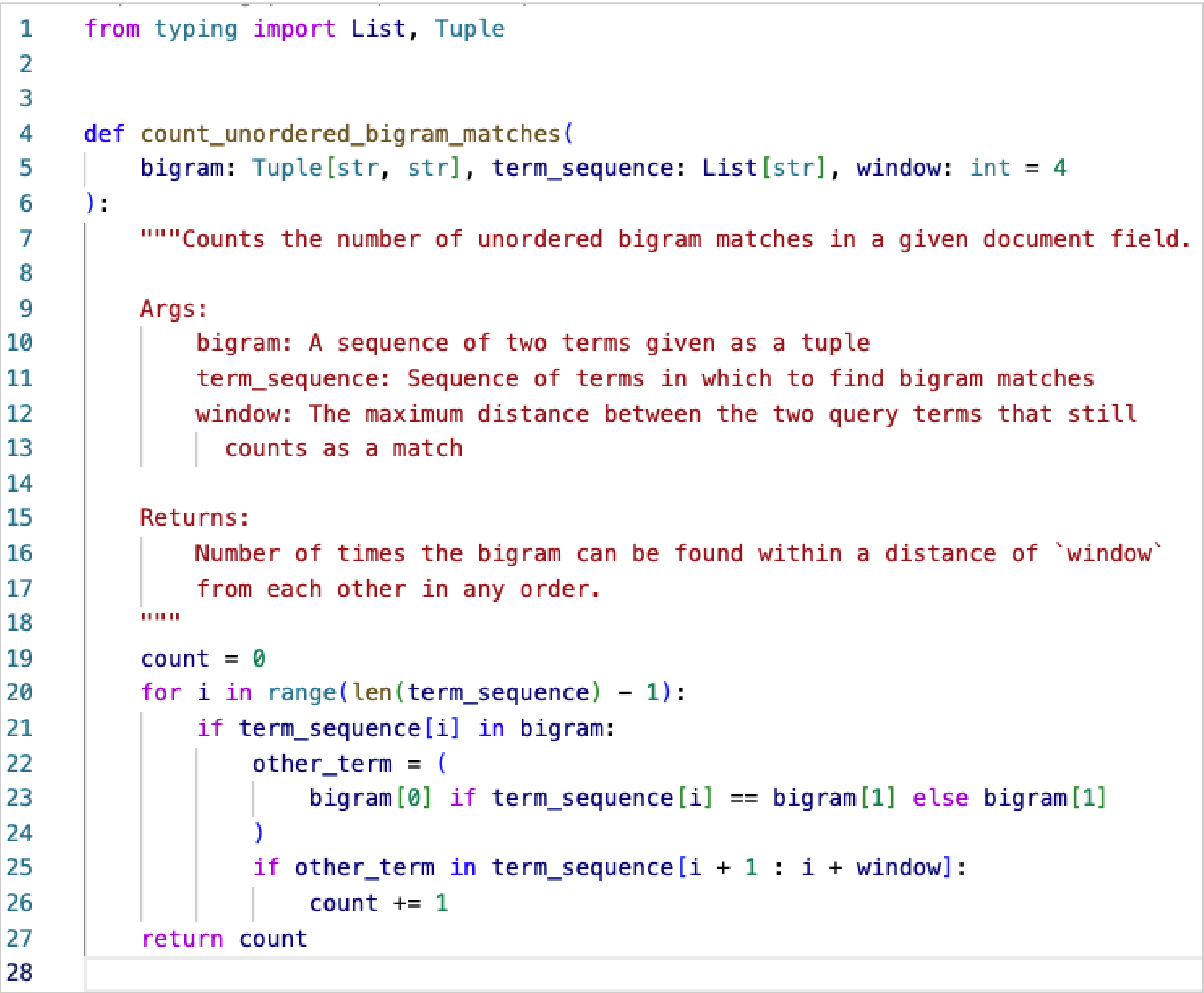
□ The posting lists contain term frequencies and positions

□ The length of a posting list will increase with the frequency of a term in any document

□ Document IDs are stored in the payload

Maximum marks: 3

11 Coding



```
1   from typing import List, Tuple
2
3
4   def count_unordered_bigram_matches(
5       bigram: Tuple[str, str], term_sequence: List[str], window: int = 4
6   ):
7       """Counts the number of unordered bigram matches in a given document field.
8
9       Args:
10          bigram: A sequence of two terms given as a tuple
11          term_sequence: Sequence of terms in which to find bigram matches
12          window: The maximum distance between the two query terms that still
13            counts as a match
14
15      Returns:
16          Number of times the bigram can be found within a distance of `window`
17          from each other in any order.
18      """
19      count = 0
20      for i in range(len(term_sequence) - 1):
21          if term_sequence[i] in bigram:
22              other_term = (
23                  bigram[0] if term_sequence[i] == bigram[1] else bigram[1]
24              )
25              if other_term in term_sequence[i + 1 : i + window]:
26                  count += 1
27      return count
28
```

Is the above implementation correct? If not, specify the problem line and give an input on which it fails (bigram and text sequence). *(4 points)*

Fill in your answer here

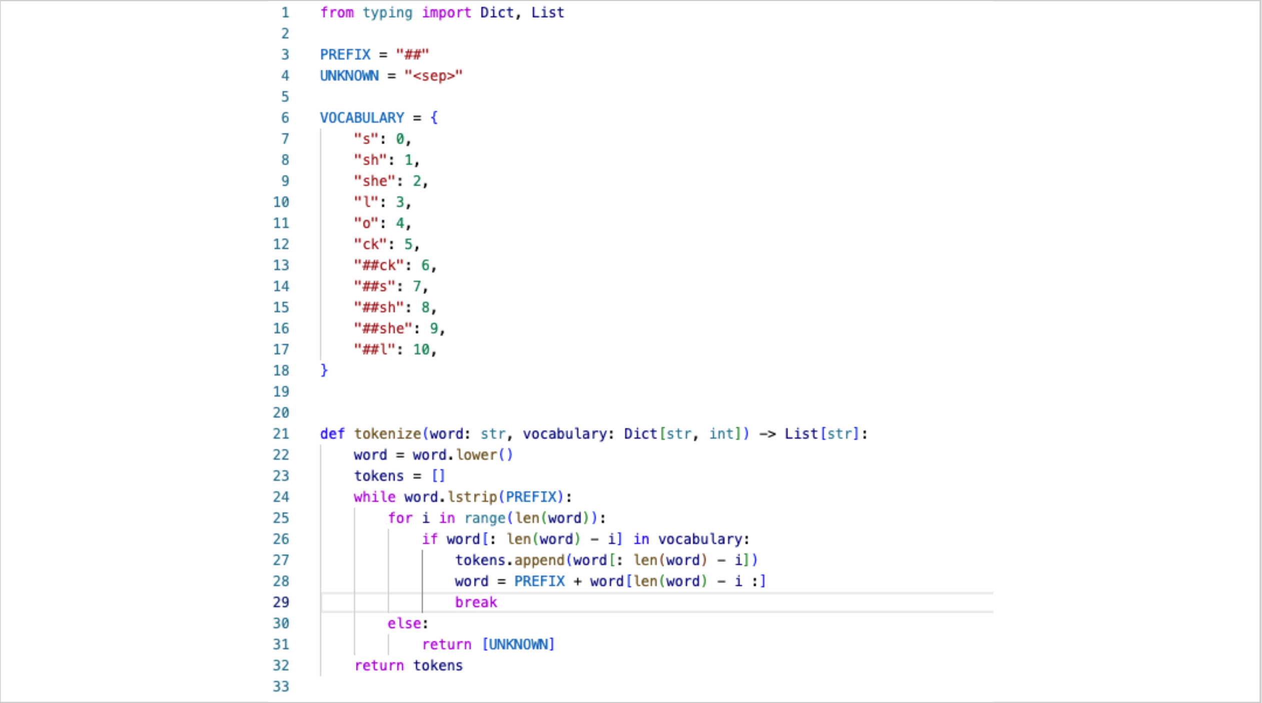
Format

Σ

Words: 0

Maximum marks: 4

12 Coding



```
from typing import Dict, List

PREFIX = "##"
UNKNOWN = "<sep>"

VOCABULARY = {
    "s": 0,
    "sh": 1,
    "she": 2,
    "l": 3,
    "o": 4,
    "ck": 5,
    "##ck": 6,
    "##s": 7,
    "##sh": 8,
    "##she": 9,
    "##l": 10,
}


def tokenize(word: str, vocabulary: Dict[str, int]) -> List[str]:
    word = word.lower()
    tokens = []
    while word.lstrip(PREFIX):
        for i in range(len(word)):
            if word[: len(word) - i] in vocabulary:
                tokens.append(word[: len(word) - i])
                word = PREFIX + word[len(word) - i :]
                break
        else:
            return [UNKNOWN]
    return tokens

```

Describe in short what the above function does. What will be the output for *tokenize("shells", VOCABULARY)*? What will be the output for *tokenize("shellshock", VOCABULARY)*? *(4 points)*

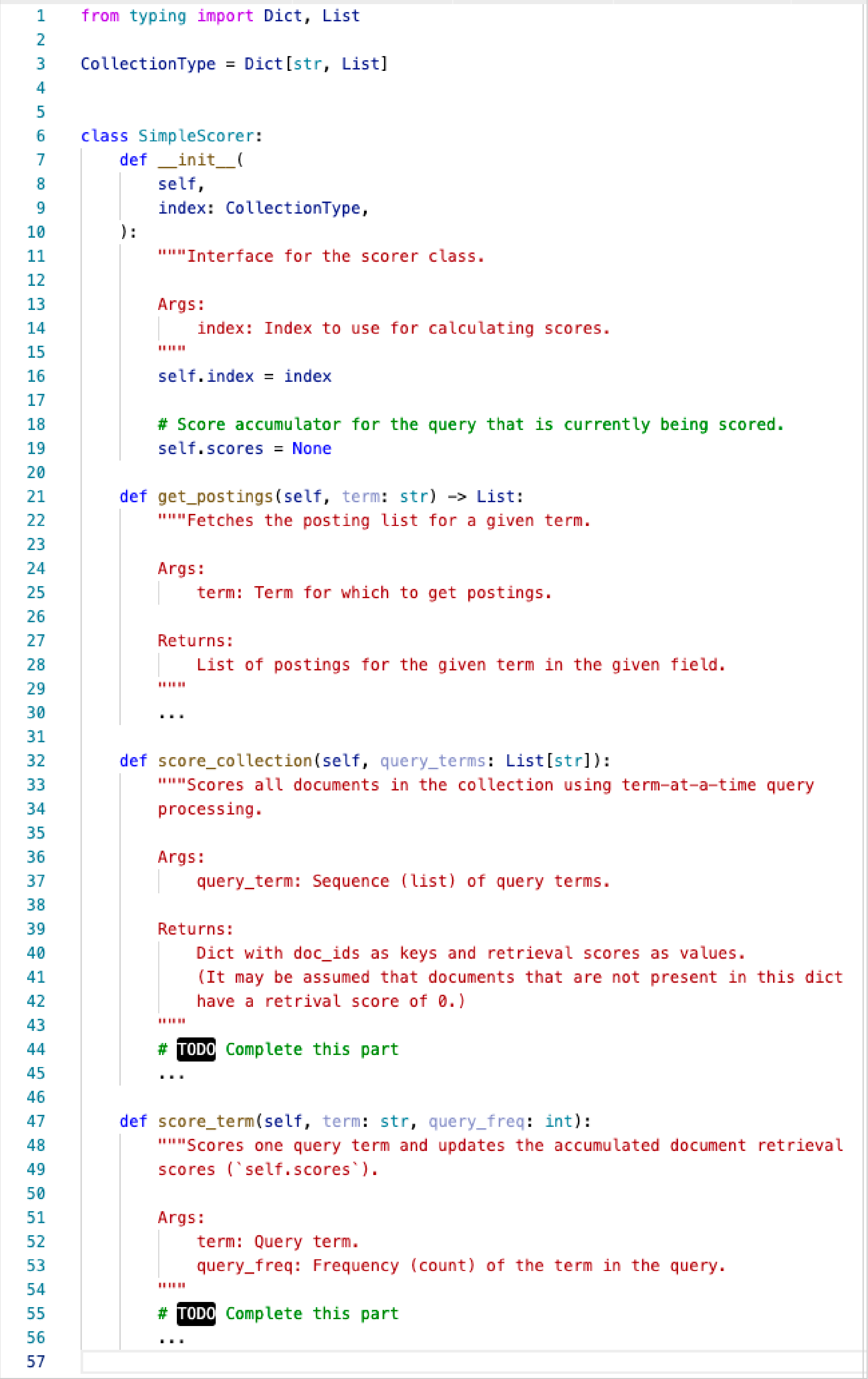
Fill in your answer here

Format

Words: 0

Maximum marks: 4

13 Coding



```python
from typing import Dict, List

CollectionType = Dict[str, List]


class SimpleScorer:
    def __init__(
        self,
        index: CollectionType,
    ):
        """Interface for the scorer class.

        Args:
            index: Index to use for calculating scores.
        """
        self.index = index

        # Score accumulator for the query that is currently being scored.
        self.scores = None

    def get_postings(self, term: str) -> List:
        """Fetches the posting list for a given term.

        Args:
            term: Term for which to get postings.

        Returns:
            List of postings for the given term in the given field.
        """
        ...

    def score_collection(self, query_terms: List[str]):
        """Scores all documents in the collection using term-at-a-time query
        processing.

        Args:
            query_term: Sequence (list) of query terms.

        Returns:
            Dict with doc_ids as keys and retrieval scores as values.
            (It may be assumed that documents that are not present in this dict
            have a retrival score of 0.)
        """
        # TODO Complete this part
        ...

    def score_term(self, term: str, query_freq: int):
        """Scores one query term and updates the accumulated document retrieval
        scores (`self.scores`).

        Args:
            term: Query term.
            query_freq: Frequency (count) of the term in the query.
        """
        # TODO Complete this part
        ...

```

Implement the *score_collection()* and *score_term()* methods to perform term-at-a-time scoring using a simple retrieval function. You can use the *get_postings()* method if necessary. *(2x2 points)*

Fill in your answer here

Maximum marks: 4

14 Conversational search

Which of the following metrics take into consideration the turn depth and focus on deeper rounds to capture the ability of the system to understand the context of the whole conversation? *(3 points; -1 if incorrect)*

Select one alternative:

○ NDCG@3 averaged for all turns in the given topic

○ Mean Average Precision across all topics in the test dataset

○ Average NDCG@3 at each turn depth for all turns in all topics

○ Recall@1000 across all topics in the test dataset

Maximum marks: 3

15 Fairness

Select all statements that are correct about fairness in IR. *(2 points)*

Select one or more alternatives:

□ Fairness does not have a unique definition in IR

□ Fairness can be studied at different levels

□ Fairness has a unique definition in IR

Maximum marks: 2

16 Conversational information access evaluation

Given the space of possible system responses and the conversation history up to that point, is it possible to create a reusable offline text collection to automatically evaluate system responses at a given conversation turn? *(2 points; -1 if incorrect)*

Select one alternative:

○ True

○ False

Maximum marks: 2

17 Conversational information access

Connect the given statements with the dialogue system components. You are expected to know what the acronyms stand for. *(5x1 point)*

Please match the values:

	ST	NLG	DP	NLU
Responsible for sentence realization and in some cases content planning	○	○	○	○
Does slot filling for the current utterance	○	○	○	○
Responsible for determining domain and intent	○	○	○	○
Decides what dialogue act to generate	○	○	○	○
Determines both the current state of the frame and the user’s most recent dialogue act	○	○	○	○

Maximum marks: 5

18 Conversational information access

Which of the following statements are true for conversational AI? *(3 points)*

Select one or more alternatives:

□ A frame in frame-based dialogue systems represents the kinds of intentions the system can extract from user sentences

□ Chatbots are task-based dialogue systems whose task is to entertain people

□ A frame in frame-based dialogue systems consists of a collection of slots, each of which can take a set of possible slot values

□ Chatbot architectures fall into two classes: rule-based systems and template-based systems

Maximum marks: 3

19 Retrieval models

Which of the following statements about negative sampling used in dense retrieval are *false*? *(2 points)*

Select one or more alternatives:

□ Random and in-batch negatives are very informative for a model because they are less relevant than the top match for a given query.

□ Negative samples are used to fine-tune a neural model for sparse retrieval task.

□ The main goal of the negative sampling is helping the model with placing query and the relevant documents close to each other in the embedding space and far away from non-relevant documents.

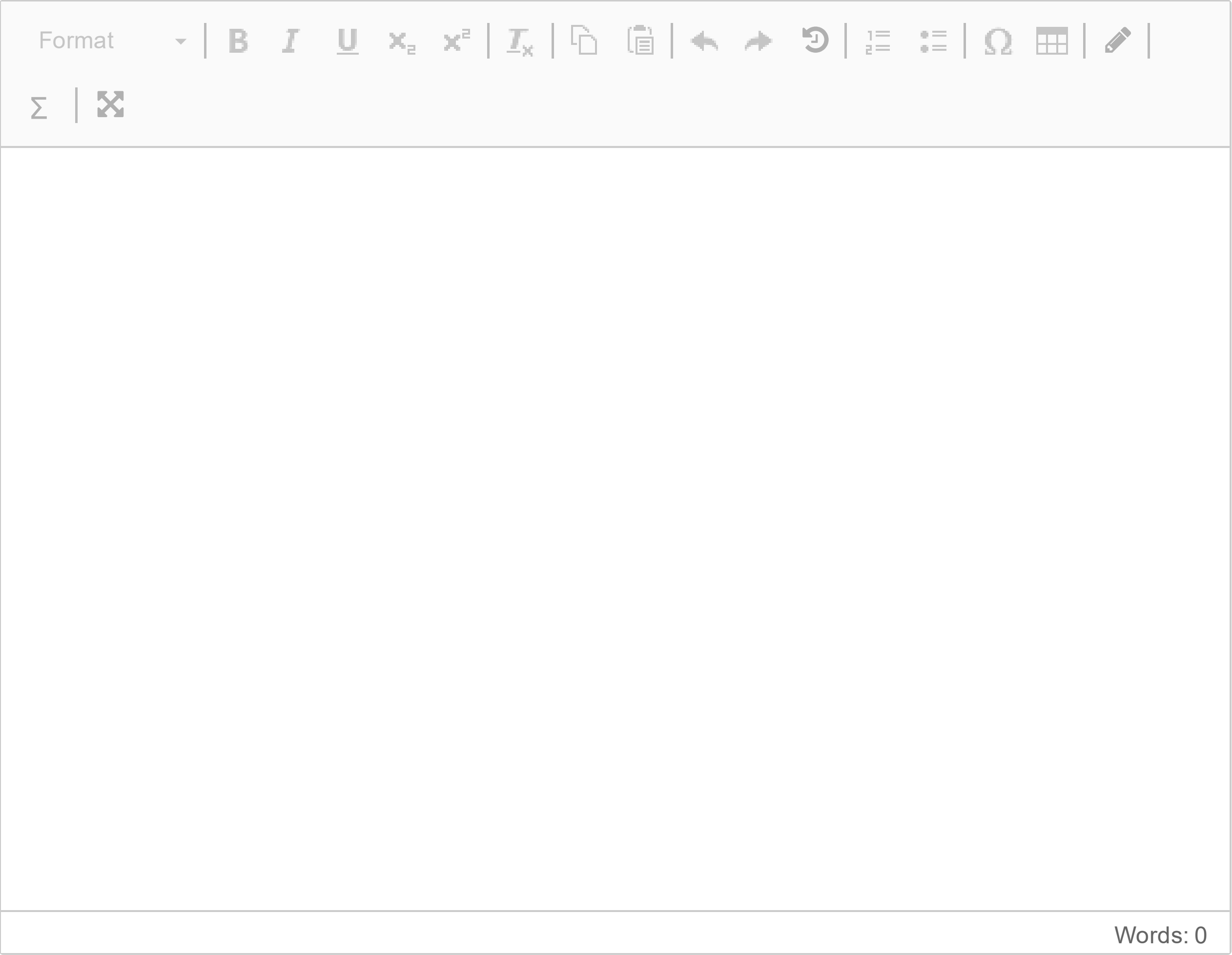
□ The top non-relevant documents retrieved by a retrieval system given a query are good negative examples for fine-tuning a model.

Maximum marks: 2

20 Retrieval models

What are the main characteristics of an interaction-focused ranking system? Focus on the architecture, main components, and relevance estimation in your answer. *(3 points)*

Fill in your answer here



Words: 0

Maximum marks: 3

21 Retrieval models

What are the main advantages of using a contextualized distributional text representation in retrieval compared with a discrete text representation? *(2 points)*

Fill in your answer here

Format

Σ

Words: 0

Maximum marks: 2

22 Entity linking

Which of the following statements about entity disambiguation is *false*? *(*3 points, -1 if incorrect*)*

Select one alternative:

○ Commonness in itself can be used as a disambiguation approach

○ Incorporating context and dependence between entities increases complexity

○ Individual local disambiguation is NP-hard

○ Disambiguation approaches make simplifying assumptions for computational tractability

Maximum marks: 3

23 Entity linking

```
This is some text where [[entityA|A]] is first mentioned.
[[entityA|A]] often co-occurs with [[entityB|B]], who is known as [[entityB|The King]].

[[entityB|B]] also named his son [[entityD|B]], aka [[entityD|King Jr.]].

Others known as "The King" include [[entityC|C]] and [[entityD|D]].

The programming language [[entityCPP|C++]] is the successor of [[entityC_prog|C]].
```

You are given the above entity-annotated document collection with Wikipedia link syntax, i.e., [[Wikipedia page ID | link text]].
Assuming that all Wikipedia pages are to be treated as entities, **compute the following *commonness scores***: *(4x1 point)*

P(e=entityA | m=``A") :

P(e=entityB | m=``B") :

P(e=entityC | m=``C") :

P(e=entityD | m=``The King") :

Maximum marks: 4

24 Retrieval evaluation

In a randomization test, the null hypothesis H_0 is that two systems are identical with regards to a test statistic. *(2 points, -1 if incorrect)*

Select one alternative:

○ True

○ False

Maximum marks: 2

25 Retrieval evaluation

You are given ranked lists of n documents from m different retrieval system. You wish to obtain the relevance judgements for the retrieved documents but your budget only allows for annotating a fraction of documents (i.e., number of documents to annotate is $<< n \times m$). **Describe how to choose the subset of documents to judge? What about the remaining documents?** *(4 points)*

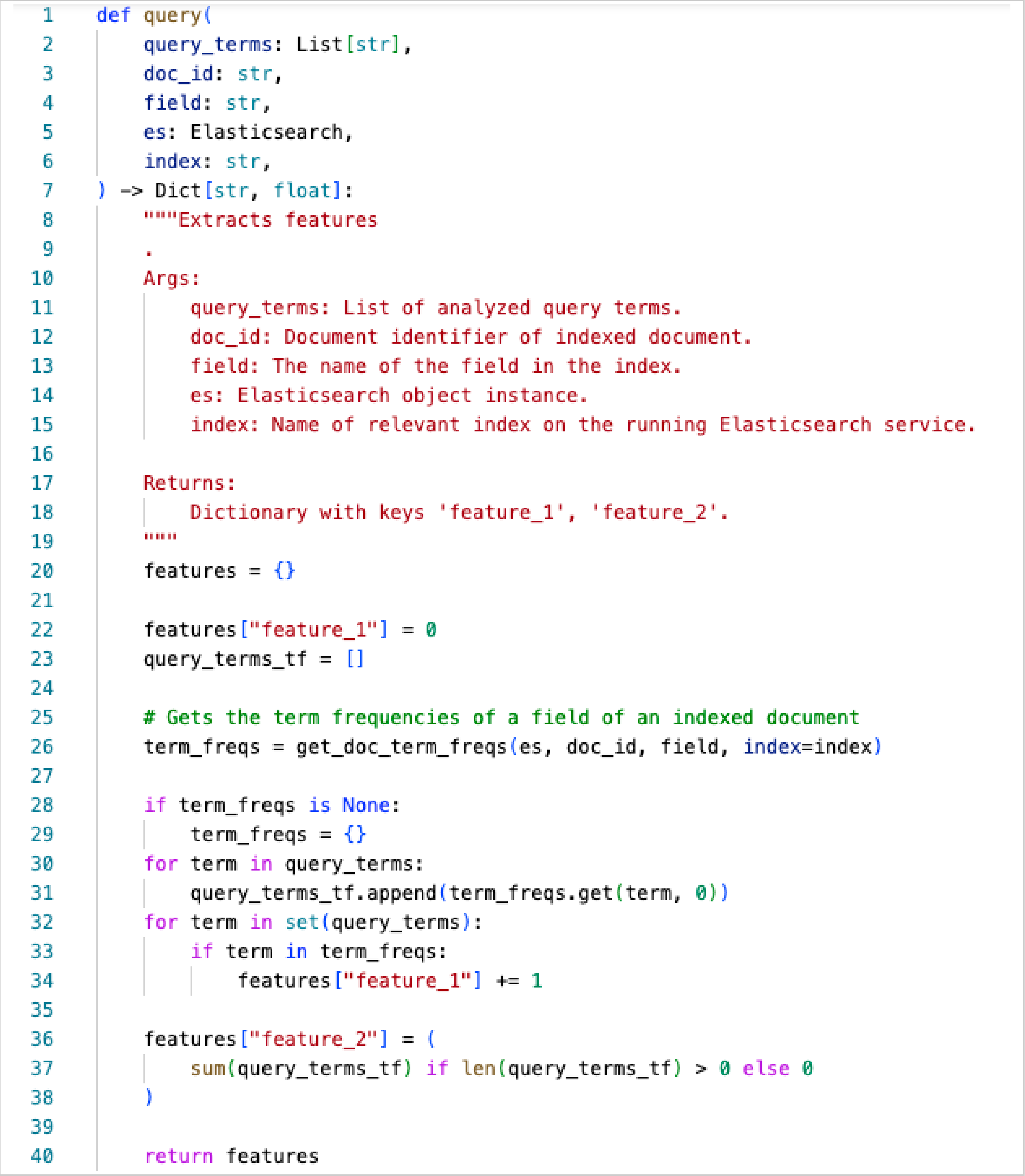
Fill in your answer here

Format

Words: 0

Maximum marks: 4

26 Coding



```
def query(
    query_terms: List[str],
    doc_id: str,
    field: str,
    es: Elasticsearch,
    index: str,
) -> Dict[str, float]:
    """Extracts features
    .
    Args:
        query_terms: List of analyzed query terms.
        doc_id: Document identifier of indexed document.
        field: The name of the field in the index.
        es: Elasticsearch object instance.
        index: Name of relevant index on the running Elasticsearch service.

    Returns:
        Dictionary with keys 'feature_1', 'feature_2'.
    """
    features = {}

    features["feature_1"] = 0
    query_terms_tf = []

    # Gets the term frequencies of a field of an indexed document
    term_freqs = get_doc_term_freqs(es, doc_id, field, index=index)

    if term_freqs is None:
        term_freqs = {}
    for term in query_terms:
        query_terms_tf.append(term_freqs.get(term, 0))
    for term in set(query_terms):
        if term in term_freqs:
            features["feature_1"] += 1

    features["feature_2"] = (
        sum(query_terms_tf) if len(query_terms_tf) > 0 else 0
    )

    return features
```

Which features are extracted in this piece of code? *(3 points)*

Select one or more alternatives:

□ Sum over the term frequencies of each query term

□ Count of tokens in the document field

□ Sum of TF scores of query terms in the document field

□ Average term frequency in of each query term

□ Count of uniques terms present in the document field

Maximum marks: 3

27 Relevance feedback

Which of the following statements about relevance feedback are *true*? *(2 points)*

Select one or more alternatives:

□ Pseudo releance feedback adjusts a query relative to the documents that initially appear to match the query

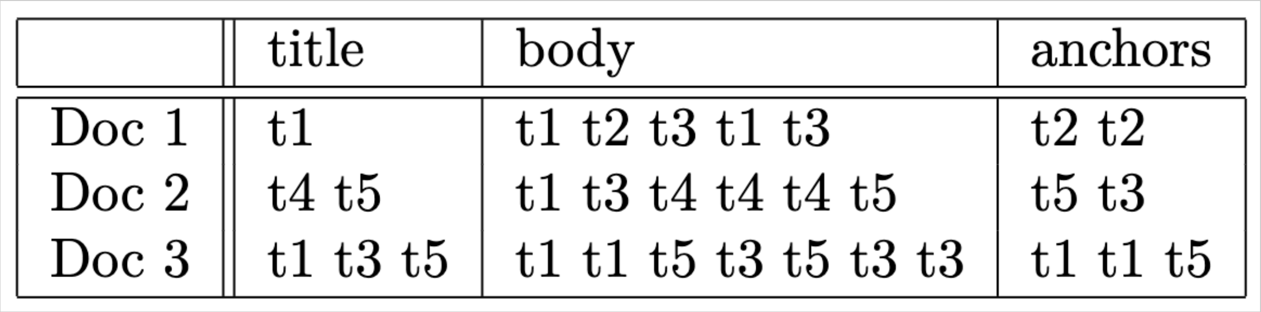
□ The weights in the expanded query are adjusted in such a way that they are always higher for the original query terms than for expansion terms

□ Relevance feedback usually extends the original query with additional terms

□ In case of blind feedback the user is not directly involved in selecting the relevant documents

Maximum marks: 2

28 Retrieval models



	title	body	anchors
Doc 1	t1	t1 t2 t3 t1 t3	t2 t2
Doc 2	t4 t5	t1 t3 t4 t4 t4 t5	t5 t3
Doc 3	t1 t3 t5	t1 t1 t5 t3 t5 t3 t3	t1 t1 t5

Fielded BM25 is defined as:

$$BM25F(d,q) = \sum_{t \in q} \frac{\tilde{c}_{t,d}}{k_1 + \tilde{c}_{t,d}} \times idf_t$$

$$\tilde{c}_{t,d} = \sum_i w_i \times \frac{c_{t,d_i}}{B_i}$$

where

- i corresponds to the field index
- w_i is the field weight (assume $[w_1 = 0.1, w_2 = 0.7, w_3 = 0.2]$)
- B_i is soft normalization for field i: $B_i = (1 - b_i + b_i \frac{|d_i|}{avgdl_i})$ (assume $b_1 = b_2 = b_3 = 0.75$)

IDF values should be computed based on the body field using natural-base logarithm.

Given the document-term matrix corresponding to the document collection, and formulae shown above, answer the following: *(2x2 points)*

What is the value of BM25F scoring function for query "t3" and document 1? □

What is the value of BM25F scoring function for query "t2 t4" and document 2? □

Maximum marks: 4