I. Information Retrieval

(1) In vector space model, terms are weighted according to their relevance to the document. tf/idf is one of the classic weighting schemes. Explain what is tf and what is idf. Explain the basic idea of the tf/idf weighting scheme. (10 points)

**Answer:**
Tf stands for term frequency. It is the frequency of a term occurring in one document. Idf stands for inverse term frequency. It measures the number of documents in which a term occurs. In tf/idf weighting, tf implies that the more a term occurs in a document, the more important that term is for that document. Idf suggests that meaningful terms should occur only in a few documents so that these terms are good discriminators, which distinguish these documents from the rest.

(2) For web retrieval, what is more important, precision or recall? Explain why? (10 points)

**Answer:**
Precision is usually more important since for most typical queries the web is too large that the exact amount of relevant documents is unknown. Most users just want one or two relevant documents in the top 10 documents that contain an answer to their question.

(3) Assuming simple term frequency weights (no IDF factor), compute the cosine similarity of the following two simple documents (Leave the square root as it is) (10 points):

(a) “computer game retrieval database database”
(b) “computer game game database”

Answer:

<table>
<thead>
<tr>
<th></th>
<th>Computer</th>
<th>Game</th>
<th>Retrieval</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc (a)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Doc (b)</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \text{Cos sim.} = \frac{1*1 + 1*2 + 1*0 + 2*1}{\sqrt{(1^2 + 1^2 + 1^2 + 2^2)} \cdot \sqrt{(1^2 + 2^2 + 0 + 1^2)}} = \frac{5}{\sqrt{7} \cdot \sqrt{6}} \]

(4) Describe how inverted files and suffix arrays are used in text indices. Give illustrative examples of indices using the sample text below (the numbers indicate word position in the text). What is the common searching algorithm using these indices? (15 points)

For illustrative examples use following sample text (the numbers indicate word position in the text.)

> We live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.

**ANSWERS**

**Inverted files**

The inverted file stucture is composed of two elements:

- the vocabulary – the set of all different words in the text.
- the occurrences. For each word a list of all the text positions where the word appears is stored. The set of all those list is called the occurrences. Look example 1.

To reduce space requirements, a technique called block addressing is used. (Look example 2.)

**Suffix trees and suffix arrays**

Suffix Trees and Suffix Arrays indexes see the text as one long string. Each position in the text is considered as a text suffix. Each suffix is thus uniquely identified by its position. (Look example 3.)

Suffix arrays provide essentially the same functionality as suffix trees with much less space requirements. A suffix array is simply an array containing all the pointers to the text suffixes listed in lexicographical order. (Look example 4.)

If suffix array is large, the use of supra-indices over the suffix array has been proposed. (Look example 5.)

**General searching procedure:**

1. Vocabulary search

   The words and patterns present in the query are isolated and searched in the vocabulary.
2. Retrieval of occurrences.
The list of the occurrences of the words found are retrieved.

3. Manipulation of occurrences.
The occurrences are processed to solve phrases, proximity or Boolean operations.
If block addressing (in inverted files) (example 2.) or supra-indices (in suffix arrays)
(example 5.) is used it may be necessary to directly search the text to find the information
missing from the occurrences.

<table>
<thead>
<tr>
<th>Text</th>
<th>Vocabulary</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>We live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.</td>
<td>live</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Trondheim</td>
<td>12, 23</td>
</tr>
<tr>
<td></td>
<td>first</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>capital</td>
<td>47, 74</td>
</tr>
<tr>
<td></td>
<td>Norway</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Now</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Oslo</td>
<td>85</td>
</tr>
</tbody>
</table>

Example 1 A sample text and an inverted index built on it.

<table>
<thead>
<tr>
<th>Text</th>
<th>Vocabulary</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.</td>
<td>live</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Trondheim</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>first</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>capital</td>
<td>3, 5</td>
</tr>
<tr>
<td></td>
<td>Norway</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Now</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Oslo</td>
<td>5</td>
</tr>
</tbody>
</table>

Example 2 The sample text split into five blocks and an inverted index using block addressing built on it.

live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.
Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.
Trondheim was the first capital of Norway. Now the capital is Oslo.
first capital of Norway. Now the capital is Oslo.
capital of Norway. Now the capital is Oslo.
Now the capital is Oslo.
Oslo.

Example 3 Suffixes of sample text.

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>9</th>
<th>12</th>
<th>23</th>
<th>33</th>
<th>37</th>
<th>41</th>
<th>47</th>
<th>55</th>
<th>58</th>
<th>66</th>
<th>70</th>
<th>74</th>
<th>82</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>We live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suffix Array</td>
<td>74 47 41 4 58 66 85 12 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 4 The suffix array for the sample text

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>9</th>
<th>12</th>
<th>23</th>
<th>33</th>
<th>37</th>
<th>41</th>
<th>47</th>
<th>55</th>
<th>58</th>
<th>66</th>
<th>70</th>
<th>74</th>
<th>82</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>We live in Trondheim. Trondheim was the first capital of Norway. Now the capital is Oslo.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supra-Index</td>
<td>capital, live, Oslo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suffix Array</td>
<td>74 47 41 4 58 66 85 12 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 5 A supra-index over suffix array

II. Document analysis

(5) What is text categorization? In which application area can text categorization be used, give one example. Rocchio and K-nearest neighbors are the two text categorization algorithms we introduced at the course; explain the basic principle of the two algorithms.
(20 points)
Answer: Text categorization is process of assigning one or more categories to a document. Text categorization can be applied in many domain. Examples are: Web page recommendation, Email message routing or language identification. The basic idea of K-nearest neighbor is: given a set of training documents D and a new document d, identify the most similar document d’ in D and assign to d the same category as d’ has. If there are more than 1 neighbors to be considered, a voting scheme (majority voting or weighted-sum voting) has to be used. In Rocchio, for each category, a prototype vector is computed by summing up the vectors of the training documents in that category. Then assign the new document to the category with the nearest prototype vector based on cosine similarity.

(6) Pattern match allowing errors is one of the query types, which retrieves all words, that are similar to the given word. Levenstein distance is usually used to measure similarity. What is the Levenstein distance (edit distance) between the following pairs of strings? Use only ‘insert’ and ‘delete’ edit operations. (5 points)
(a) “filosofy” and “philosophy”
(b) “flow” and “follow”
Answer:
(b) The Levenstein distance is 2. Edit operations, as example: insert ‘o’, insert ‘l’.

(7) What is stemming? What is its purpose? (5 points)

Answer:
Stemming is a technique for reducing words to their grammatical roots. Stemming is a process of grouping morphological variants of a term into index term classes. The purpose of stemming is to map variations of a word into a single common representation (concept), to reduce vocabulary mismatch between queries and documents.

(8) Use the mean and variable technique to answer if words “live” and “life” form a collocation in a following text set (10 points):

a. They have their own way to live life.
b. In this story he finds out that he did, indeed, live a wonderful life.
c. How to live the life you love?
d. It seems to me like I live somebody's life.
e. Live your dog's life!

Some calculations and equations, which might be needed:

Equations:
\[
\text{Mean } \bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i
\]

Calculations:
\[
\sqrt{2} = 1.4
\]
\[
\sqrt{3} = 1.7
\]

The variance:
\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (d_i - \bar{d})^2
\]

\[
\sqrt{5} = 2.2
\]
\[
\sqrt{7} = 2.6
\]

Sample deviation:
\[
s = \sqrt{s^2}
\]

Answers

Word life looking from the word live stands in different positions in each sentence:

a. 1
b. 3
c. 2
d. 4
e. 5

In sentence d. somebody's is considered as 3 words. The same is with word dog's in sentence e.
The mean is:
\[
\overline{d} = \frac{1}{n} \sum_{i=1}^{n} d_i = \frac{1 + 3 + 2 + 4 + 5}{5} = 3
\]
and variance:
\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (d_i - \overline{d})^2 = \frac{(1-3)^2 + (3-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2}{5-1} = \frac{10}{4} = \frac{5}{2}
\]
sample deviation:
\[
s = \sqrt{s^2} = \sqrt{\frac{\sqrt{2} \sqrt{5}}{2}} \approx \frac{1.41 \times 2.24}{2} \approx 1.5792
\]
The mean and deviation characterize the distribution of distances between two words in a corpus. We can use this information to discover collocations by looking for pairs with low deviation. A low deviation means that the two words usually occur at about the same distance. Zero deviation means that two words always occur at exactly the same distance. If the mean is much greater than 1.0, then a low deviation indicates phrases like: previous/games - in the previous 10 games, minus/points -minus 2 percentage points, hundreds/ dollars - hundreds of billions of dollars. (our case !) High deviation indicates that the two words of the pair stand in no interesting relationship (editorial/Atlanta, ring/New, point/hundredth, subscribers/by).

Conclusion: these two words (live and life) could be considered as collocation.

III. Ontology

(9) Concept extraction is the process of exacting important (representative) terms from text. Given a set of html files, describe the steps that are necessary to take in order to extract the top-100 ranked concepts for these files (15 points).

Answer:
The process consists of :
Remove HTML tags.
Tokenize document.
Remove stop words
Perform stemming (optional)
Attach part-of-speech tagging to the terms in documents.
Single out nouns and verbs (which are normally good candidates for prominent concept).
Calculate weight for each term. Weight can be calculated by using normal tf/idf weighting or other weighting schemes. Rank the terms by their weights and select the top-100 ranked terms.
---------End--------