

Seminar 12

Exercise 12/1

Pick a topic of your interest and describe it by 5-10 words.

Open Sketch Engine <https://ske.fi.muni.cz/>. Go to WebBootCaT. Create a corpus using the description words as seed. Wait until data are downloaded. Search the word corpus for collocations.

Solutions can vary.

Definition 1 (Index Relations)

Suppose that we could pick a random page from the index of E_1 and test whether it is in E_2 's index and symmetrically, test whether a random page from E_2 is in E_1 . These experiments give us fractions x and y such that our estimate is that a fraction x of the pages in E_1 are in E_2 , while a fraction y of the pages in E_2 are in E_1 . Then, letting $|E_i|$ denote the size of the index of search engine E_i , we have

$$x|E_1| \approx y|E_2|,$$

from which we have the form we will use

$$\frac{|E_1|}{|E_2|} \approx \frac{y}{x}.$$

Exercise 12/2

Two web search engines A and B each generate a large number of pages uniformly at random from their indexes. 30% of A 's pages are present in B 's index, while 50% of B 's pages are present in A 's index. What is the number of pages in A 's index relative to B 's?

Substituting to the Definition 1 we get the fractions

- $\frac{3}{10}$ of A is in B
- $\frac{5}{10}$ of B is in A

and we get the equation

$$\begin{aligned} 0.3|A| &\approx 0.5|B| \\ \frac{|A|}{|B|} &\approx \frac{0.5}{0.3} \\ \frac{|A|}{|B|} &\approx \frac{5}{3} \end{aligned}$$

Definition 2 (Path Similarity)

Similarity between a query XPath c_q and a document path c_d is calculated as

$$CR(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ can be expanded to } c_d \text{ by adding nodes to the path} \\ 0 & \text{otherwise} \end{cases}$$

Definition 3 (Structural Term)

Structural term is defined as an XML-context/term pair denoted by $\langle c, t \rangle$ of existing path to a value and the value itself, where the value itself is also a node in the XML document. For example, an XML document containing only a root element with `test`.

```
<root>
  test
</root>
```

contains two structural terms $\langle \text{/root/}, \text{test} \rangle$ and $\langle \text{/}, \text{test} \rangle$.

Exercise 12/3

Consider the following the XML document:

```
<Course_Catalog>
  <Department Code="CS">
    <Title>Computer Science</Title>
    <Chair>
      <Professor>
        <First_Name>Jennifer</First_Name>
        <Last_Name>Widom</Last_Name>
      </Professor>
    </Chair>
    <Course Number="CS106A" Enrollment="1070">
      <Title>Programming Methodology</Title>
      <Description>Introduction to the engineering of computer applications
        emphasizing modern software engineering principles.</Description>
      <Instructors>
        <Lecturer>
          <First_Name>Jerry</First_Name>
          <Middle_Initial>R.</Middle_Initial>
          <Last_Name>Cain</Last_Name>
        </Lecturer>
        <Professor>
          <First_Name>Eric</First_Name>
          <Last_Name>Roberts</Last_Name>
        </Professor>
        <Professor>
          <First_Name>Mehran</First_Name>
          <Last_Name>Sahami</Last_Name>
        </Professor>
      </Instructors>
    </Course>
    <Course Number="CS106B" Enrollment="620">
      <Title>Programming Abstractions</Title>
      <Description>Abstraction and its relation to programming.</Description>
      <Instructors>
        <Professor>
          <First_Name>Eric</First_Name>
          <Last_Name>Roberts</Last_Name>
        </Professor>
        <Lecturer>
```

```

    <First_Name>Jerry</First_Name>
    <Middle_Initial>R.</Middle_Initial>
    <Last_Name>Cain</Last_Name>
  </Lecturer>
</Instructors>
<Prerequisites>
  <Prereq>CS106A</Prereq>
</Prerequisites>
</Course>
</Department>
</Course_Catalog>

```

1. Write the following expressions:
 - a) Return all titles (including both departments and courses).
 - b) Return all course titles that contain the word *programming*.
 - c) Return the surnames of all instructors teaching at least one course that contains the word *software* in its description.
 - d) Return the surnames of all professors teaching at least one course that contains the word *software* in its description.
2. Calculate the similarity between the queries and the corresponding document paths.
 - a) `//Instructors//Last_Name#Cain`
 - b) `//Course/Instructors/Lecturer/Last_Name#Cain`

1. a) `//Title`
- b) `//Course//Title[contains(current(), 'programming')]`
- c) `//Course[contains(Description, 'software')]/Instructors//Last_Name`
- d) `//Course[contains(Description, 'software')]/Instructors/Professor/Last_Name`

2. By Definition 2, a query c_q corresponds to document c_d if and only if it can be expanded. Original query for a) can be expanded to `Course_Catalog/Department/Course/Instructors/Lecturer/Last_Name`. Since c_q is expandable to c_d , use the equation from the definition. Substituting for the query length $c_q = 2$ and the document length $c_d = 6$ to the formula we get

$$CR(c_q, c_d) = \frac{1 + |c_q|}{1 + |c_d|} = \frac{1 + 2}{1 + 6} = \frac{3}{7}.$$

For b) we only change the query length $c_q = 4$ and obtain

$$CR(c_q, c_d) = \frac{1 + |c_q|}{1 + |c_d|} = \frac{1 + 4}{1 + 6} = \frac{5}{7}.$$

Exercise 12/4

Count how many structural terms are present in the XML tree:

```
<Course>
  <Title>Programming Abstractions</Title>
  <Description>Abstraction and its relation to programming</Description>
  <Instructors>
    <Professor>
      <First_Name>Eric</First_Name>
      <Last_Name>Roberts</Last_Name>
    </Professor>
  </Instructors>
</Course>
```

To save space, mark each element with its first letter only (D stands for Description, P for Professor, ...). By Definition 3, we count all combinations and write them into the table:

C T Programming Abstractions	C I P F Eric	C I P L Roberts
T Programming Abstractions	I P F Eric	I P L Roberts
Programming Abstractions	P F Eric	P L Roberts
C D Abstraction ...	F Eric	L Roberts
D Abstraction ...	Eric	Roberts
Abstraction ...		

There are 16 structural terms in total.