Information Retrieval

WS 2016 / 2017

Lecture 8, Tuesday December 13th, 2016
(Vector Space Model)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

Overview of this lecture

- Organizational
  - Your experiences with ES7 — Web app, part 2
  - Demo of some web apps
- Contents
  - Encoding — last part of L7 again
  - Vector Space Model (VSM) — documents as vectors
  - Exercise Sheet 8: re-implement your code from ES2 using the VSM, and re-evaluate benchmark

Experiences with ES7 1/4

- Summary / excerpts
  - Interesting + fun again, but more work than expected

    Not much code, but a lot to understand and a lot that can go wrong + encoding issues can drive you crazy

    Many of you quite busy before Christmas .. as usual
  - Happy to see the end result
  - Jenkins required encoding tag in Java build.xml
  - Add a slide on std::wstring conversion in C++

    Was discussed on the forum + I added a slide now

Experiences with ES7 2/4

- Demos
  - Many of you produced some really nice web apps

    Let's look at a small selection together !
  - Let us also appreciate the easter eggs (or rather: xmas cookies) that were hidden in our new cities2.txt when searching for these lovely places:

    Meteor

    grubierF

    Santas Village

Experiences with ES7 3/4

- Spiritual vs. Solid
  - One of the hallmarks of our (self-)consciousness is that our brain constantly maintains a relative stable view of the world around us (with us in it)

    Note that, in reality, it's the opposite of stable: trillions of particles in a constant flux at extremely high speed, with a constant battle of life and death at all levels

  - This model is extremely selective, conceptual, and biased

    Selective: too much information, our brains ignore most

    Conceptual: we see a "person" and not a mass of cells

    Biased: our brain fills in the gaps for the sake of stability

Experiences with ES7 4/4

- Spiritual vs. Solid
  - What's more important for your brain when seeing another living being in the world:

    See the trillions of cells this person is made of, and all the biomolecular machines and motor proteins at work?

    Have a good idea of the intentions of this person's mind?
  - What's more important for your brain when seeing an inanimate object in the world:

    See the vast amount of space between the electrons and the nuclei of the atoms the objects are made of?

    Have a good idea of what happens when your body collides with it?

Vector Space Model 1/8

- Motivation
  - For this lecture, it will be useful to represent documents as **vectors** … here is our running example for today:

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

  - Each row corresponds to a word, each column to a document
  - Non-zero entries: score for that word in that document

    In the lecture, we use tf scores … for ES8, use BM25 scores

Vector Space Model 2/8

- Terminology
  - Often referred to as the **Vector Space Model (VSM)**
  - In the VSM, words are traditionally referred to as **terms**
  - Putting the vectors from all documents from a given corpus side by side gives us the so-called **term-document matrix**

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

Vector Space Model 3/8

Q = web surfing

- Retrieval
  - A query can also be represented as a vector ... we take 1 for a term used in the query, and 0 for all other terms
  - We measure the relevance of a document to the query by taking the **dot product** of the two vectors

    Note: this is exactly the same score as in Lecture 2

	D_1	D_2	D_3	D_4	D_5	D_6		Q
internet	1	1	0	1	0	0		0
web	1	0	1	1	0	0		1
surfing	1	1	1	2	1	1		1
beach	0	0	0	1	1	1		0
	2	1	2	3	1	1		

Vector Space Model 4/8

- Algebra
  - More formally, let us write A for the term-document matrix and q for the query vector
  - Then the matrix-vector product $q^T \cdot A$ gives us a vector with the relevance scores of all the documents

Let us implement this together now

	D_1	D_2	D_3	D_4	D_5	D_6	Q
internet	1	1	0	1	0	0	0
web	1	0	1	1	0	0	1
surfing	1	1	1	2	1	1	1
beach	0	0	0	1	1	1	0

Vector Space Model 5/8

- Basic linear algebra in Python
  - For standard linear algebra, we can use **numpy**

```
sudo apt-get install python3-numpy
```

```
import numpy
A = numpy.array([[1, 1, 0, 1, 0, 0], …])
q = numpy.array([0, 1, 1, 0])
scores = q.dot(A)
print(scores)
```

Use **numpy.array** and **dot** for multiplication, not *

q is a row vector above = q^T from the previous slide

See the code from the lecture for more example usage

Vector Space Model 6/8

- Sparse matrices
  - Most entries in a term-document matrix are **zero**

    Storing all entries explicitly infeasible for large matrices

  - Sparse-matrix representation: store only the non-zero entries (together with their row and column index)

    (1, 0, 0), (1, 0, 1), (1, 0, 3), …, (2, 2, 3), …

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

Vector Space Model 7/8

- Sparse matrices
  - Two principle ways to store the list of non-zero values

    row-major: store row by row (sort by row index first)

    column-major: store col by col (sort by col index first)

  - Note: the sparse row-major representation of a term-document matrix is equivalent to an inverted index

    (1, 0, 0), (1, 0, 1), (1, 0, 3) inverted list for term 0
    (1, 1, 0), (1, 1, 2), (1, 1, 3) inverted list for term 1
    (1, 2, 0), (1, 2, 1), (1, 2, 2), … inverted list for term 2
    (1, 3, 3), (1, 3, 4), (1, 3, 5) inverted list for term 3

    (non-zero score, row index = term id, col index = doc id)

Vector Space Model 8/8

CSR = compressed sparse row

- Sparse matrices in Python
  - Not included in numpy, we have to use **scipy**

```
sudo apt-get install python3-scipy
```

```
import scipy.sparse
nz_vals  = [1, 1, 1, 1, 1, 1, …]
row_inds = [0, 0, 0, 1, 1, 1, …]
col_inds = [0, 1, 3, 0, 2, 3, …]
A = scipy.sparse.csr_matrix((nz_vals, (row_inds, col_inds)))
q = scipy.sparse.csr_matrix([0, 1, 1, 0])
scores = q.dot(A)
print(scores)
```

See the code from the lecture for more example usage

References

- Textbook

  Section 6.3: The vector space model for scoring

- Linear algebra in Python
  - http://www.numpy.org
  - http://www.scipy.org