Measuring the Structural Similarity of Semistructured Documents Using Entropy

Sven Helmer

University of London, Birkbeck

London, UK

Introduction

- XML is everywhere . . .
- In traditional IR detecting similarities used widely:
  - for querying
  - for clustering
- Consequently, lots of similarity measures for text documents

Introduction(2)

- New challenges with semistructured documents:
  - measuring structural similarity
  - semistructured documents show great structural diversity
- Measuring structural similarity used for:
  - entity resolution in data cleaning
  - clustering documents before extracting DTD or schema information
  - integrating heterogeneous data sources
  - as a query tool for inexperienced users (query-by-example)

Measuring Entropy

- Bennet et al. introduced concept of universal information metric
- Based on Kolmogorov complexity:
  - given data object x, Kolmogorov complexity $K(x)$ is the length of shortest program that outputs x
- Generalized form is conditional Kolmogorov complexity $K(x|y)$:
  - length of the shortest program with input y that outputs x

Information Distance

- Similarity of two data objects can be measured by normalized information distance:

$$NID(x,y) \quad = \quad \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))}$$

- Has some nice properties: it's "almost" a metric, lower bound for admissible distances
- So what's the catch?

Information Distance(2)

- Unfortunately, Kolmogorov complexity is not computable in general
- However, can be approximated by compression (Cilibrasi and Vitányi):

$$NCD(x,y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))}$$

Measuring Structural Similarity

- Just compressing XML files does not get the job done
- Extract structural information first:
  - Tags: list element/attribute names in document order
  - Pairwise: like tags, but with names of parents
  - Path: like tags, but with full path to root
  - Family order: family-order traversal of document
- Except Path, all extractions can be done in linear time

Measuring Structural Similarity(2)

- After extracting structural information, we use
  - NCD with gzip
  - Ziv-Merhav crossparsing

  to come up with similarity measure
- Can be done in linear time (with suffix trees)

Competitors

- Tree-editing distance (Nierman and Jagadish):
  - measuring the minimum editing distance
  - five different edit operations: relabel, insert & delete node, insert & delete (sub-)tree
  - Quadratic runtime

Competitors(2)

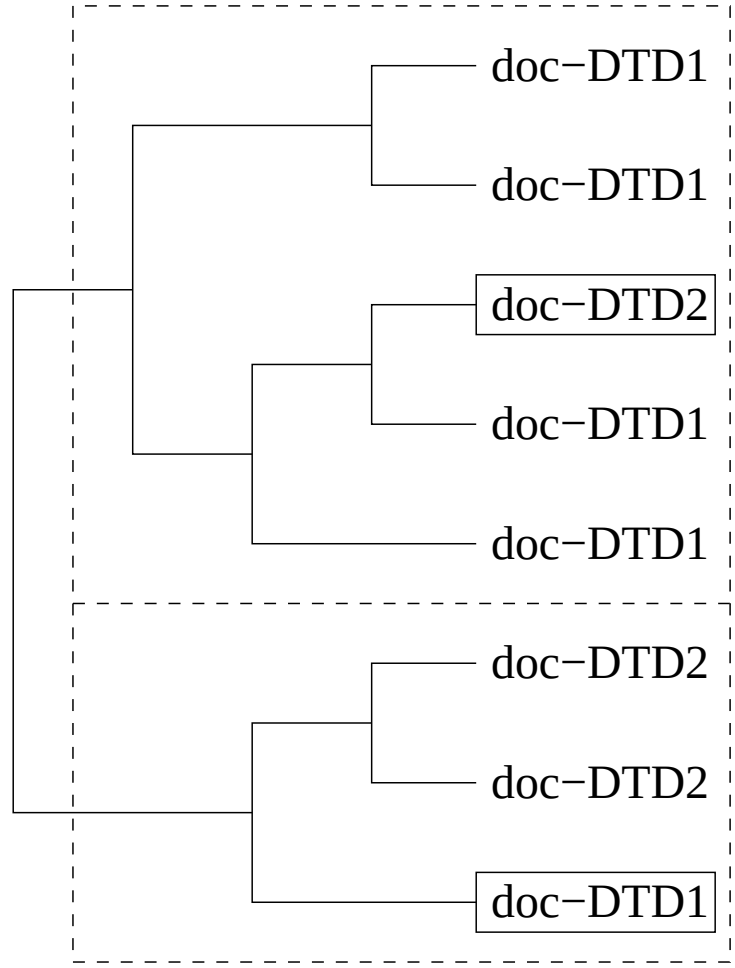
- Discrete Fourier Transformation (Flesca et al.):
  - encode XML document as a time series
    - rotate document by 90°, interpret indentations as time series
  - use DFT transform to compute similarity
  - Runtime: $N \log N$ (N size of larger document)

Competitors(3)

- Path shingles (Buttler)
  - extract structural information using the Full Path variant
  - compute a hash value h_j for each path
  - a shingle of width w is the combination of w consecutive hash values
  - compute similarity between two documents using Dice coefficient on the two sets of shingles
  - Original version is not linear, can be made linear by using different extraction technique

Clustering Quality

- Measure quality of similarity measure by clustering
- We used hierarchical agglomerative clustering
- Quality expressed in number of misclusterings in dendrogram



Document Collections

- We used three different document collections for experimental evaluation:
  - Real data sets: SIGMOD record, INEX 2005, music sheets encoded in XML
  - Synthetically generated data sets from the DFT paper
  - Own synthetically generated data sets, varying:
    - element names
    - element frequencies
    - element positions
    - element depths

Overall Results

tree-edit	15.3%
DFT	
direct ML	22.4%
pairwise ML	19.7%
Shingles	
tags	20.4%
pairwise	17.8%
full path	15.3%

gzip	
simple	26.1%
tags	17.7%
pairwise	20.8%
full path	16.9%
family order	18.9%
Ziv-Merhav	
tags	11.7%
pairwise	13.8%
full path	11.3%
family order	10.6%

More Detailed Results

- Different methods have different strengths and weaknesses:
  - tree-edit: generally good, has problems with largely varying document sizes
  - DFT: good at frequencies, bad at element names, position, and depth
  - gzip/Ziv-Merhav: bad at frequencies, good at element names, position, and depth
- DFT and gzip/Ziv-Merhav are complementary to each other; idea: combine them

Hybrid Version

Hybrid (DFT/Ziv-Merhav)	
pairw. ML/tags	8.8%
pairw. ML/pairw.	12.4%
pairw. ML/path	9.7%
pairw. ML/family	21.4%

- Clustering performance becomes even better (except family order)
- Hybrid approach does not have linear run time

Conclusion and Outlook

- Our approach totally different to previous approaches
- Can be done in linear time (important for large document collections)
- Possible future work:
  - more sophisticated ways of encoding document structure?
  - different entropy measure better suited for structural information?