Information Retrieval

WS 2012 / 2013

Lecture 12, Wednesday January 30th, 2013
(Ontologies, SPARQL)

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

Overview of this lecture

- Organizational
  - Your results + experiences with Ex. Sheet 11 (SVMs)
- Ontologies
  - Ontologies = fact databases ... ask questions like:

    Pairs of actors that are married and starred in the same movie
  - The SPARQL ontology query language
  - Translate SPARQL queries to SQL queries on a database
  - Performance issues
  - **Exercise Sheet 12:** Write a program that automatically translates a given SPARQL query to an equivalent SQL query

Experiences with ES#11 (SVMs)

$dist(x, H) = \frac{|w \cdot x - b|}{|w|}$

- Summary / excerpts last checked January 30, 16:00
  - The actual task was not so difficult / it was fun
  - Need to get used to the SVM software
  - Not clear how to compute band-width for Naïve Bayes

    I meant: compute dist from H to closest -1 and closest +1 sample
  - Also not clear how to compute b-w for SVM with outliers

    I meant: just take 2/|w| from the output of svn_learn

Your results for ES#11 (SVM vs. NB)

UNI FREIBURG

- For the new dataset (10.752 documents, **2** classes)
  - Accuracy of SVM strict and NB both around 95%
  - Accuracy of SVM with outliers is only around 90%
  - Band width of NB is much smaller than for SVM, but that does not seem to matter here for accuracy
  - How can NB be so good, despite its "naiveness" ?
  - Experience shows that

    NB indeed **estimates badly** ... the Prob(C=c|doc)

    But nevertheless often **classifies well**

    (Understand that, in practice, we are not interested in accurate probabilities, but just that the correct class gets the highest one)

Ontologies 1/5

- **Ontology** = a database of **facts** on **entities**
  - With unique identifiers for each entity
  - Without loss of generality, facts are expressed as

    subject predicate object triples ... understand why w.l.o.g.

    Brad_Pitt acted_in Mr._and_Mrs._Smith
    Brad_Pitt acted_in Burn_After_Reading
    Angelina_Jolie acted_in Mr._and_Mrs._Smith
    Joel_Cohen directed Burn_After_Reading
    Ethan_Cohen directed Burn_After_Reading
    Brad_Pitt married_to Angelina_Jolie

Ontologies 2/5

- Relation to the "Semantic Web" (SW)
  - The classical web contains a lot of facts hidden in text

    for example: infos about an actor or a movie on IMDB
  - The Semantic Web initiative is concerned with making ontology data **explicitly** available on the web
  - The challenges are really about standardization:

    Unique identifiers ... use URIs + namespaces

    Diff. identifiers meaning the same thing ... use owl:sameAs

    Well-defined syntax ... RDF has become common
  - Not the topic of this lecture / course ... but let's look at a few examples: GeoNames, Yago, FreeBase, ...

Ontologies 3/5

- The GeoNames ontology
  - Very complete database of geographical features:

    cities, countries, rivers, mountains, roads, ...
  - Around 10M entities, 250MB compressed
  - Download from http://www.geonames.org
  - RDF endpoint: http://www.geonames.org/ontology

Great dataset, but for this lecture we want something more general-purpose ...

Ontologies 4/5

- The YAGO ontology (Yet Another Great Ontology)
  - From Suchanek et al, WWW 2007 & J.Web.Sem 2008
  - General-purpose facts, extracted from Wikipedia +WordNet
  - About 2M entities and 15M facts
  - Download from http://www.mpi-inf.mpg.de/yago
  - Accuracy is good, but many popular facts are missing

    for example, 1.1 actors per movie on average

  Nevertheless, small and simple and was hence quite popular with researchers (including us) for a while ...

Ontologies 5/5

- The FreeBase Ontology
  - A general-purpose ontology, community-maintained
  - Developed by Metaweb, aquired by Google in 2010
  - Around 22M entities, 7.5GB compressed
  - Download from http://download.freebase.com
  - RDF endpoint: http://rdf.freebase.com
  - Rather complex, somewhat inconsistent structure

The currently most complete and most accurate general-purpose ontology, we extracted some parts for you ...

SPARQL 1/5

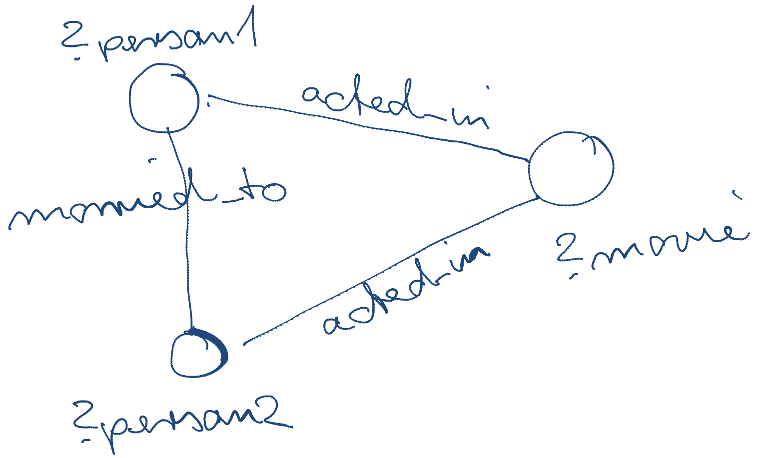
pronounced: Sparkle

- Structured queries on ontologies
  - For example: pairs of actors who are married and starred together in at least one movie
  - Difference to text search: given an ontology, the result set is well-defined
  - **SPARQL** = **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage
  - The standard query language for ontology queries

```
SELECT ?person1 ?person2 ?movie WHERE {
  ?person1  acted_in  ?movie  .
  ?person2  acted_in  ?movie  .
  ?person1  married_to  ?person2  }
```

SPARQL 2/5

- Viewing SPARQL queries as subgraphs
  - In this view, ontology = entity-relation graph
  - A SPARQL query = a sub-graph with variables at some or all of the nodes
  - We want to find all matches in the ontology graph



SPARQL 3/5

acted_in

actor	movie
Brad Pitt	Mr and Mrs Smith
...	...

- SPARQL looks very much like SQL
  - Indeed, ontology data is naturally stored in databases
  - The standard query language for databases is SQL
  - Assume we have two tables acted_in and married_to

```
SELECT married_to.person1, married_to.person2
FROM married_to as m, acted_in as a1, acted_in as a2
WHERE a1.actor = m.person1
AND a2.actor = m.person2
AND a1.movie = a2.movie;
```

SPARQL 4/5

- SPARQL to SQL: generic translation
  - Assume we have a separate table for each relation, each with two columns, generically named: subject and object
  - Explanation by example ... + implem. advice on next slide

subject	object

SPARQL

```
SELECT ?p1 ?p2 ?m
WHERE {
   ?p1 acted_in ?m .
   ?p2 acted_in ?m .
   ?p1 married_to ?p2
}
```

SQL

```
SELECT a1.subject, a2.subject, a1.object
FROM acted_in as a1,
     acted_in as a2,
     married_to as m1
WHERE
     a1.object = a2.object
 AND a1.subject = m1.subject
 AND a2.subject = m1.object;
```

SPARQL 5/5

if a variable occurs n times → n-1 equalities

$o_1, \ldots, o_n \Rightarrow o_1 = o_2$ AND … AND $o_{n-1} = o_n$

- SPARQL to SQL: implementation advice
  - For each triple in the SPARQL query, have a table in the FROM clause of the SQL query

    If a relation occurs multiple times, have the corresponding table multiple times in the FROM clause, using as

    FROM acted_in as acted_in1, acted_in as acted_in2, …
  - For each variable from the SPARQL query, store an array of its occurrences in the query

    ?person1: acted_in1.subject, married_to1.subject, ...
  - Add the corresponding equalities to the WHERE clause

    WHERE acted_in1.subject = married_to1.subject AND ...

Performance 1/3

- Cross product of tables
  - Understand that, conceptually, an SQL statement like

    FROM T_1, T_2, ..., T_k WHERE ... = ... AND ... = ... AND ...

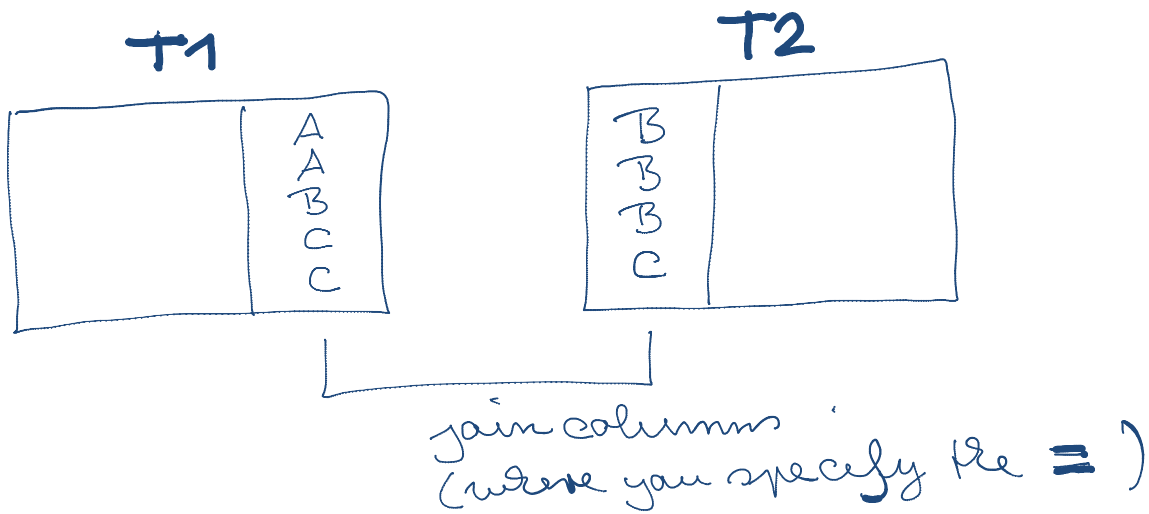
    selects elements from the cross-product

    $T_1 \times \cdots \times T_k$ (which has $|T_1| \cdot \cdots \cdot |T_k|$ elements)

    (where some or all of the T_i can be the same table)

Performance 2/3

- Joining of tables
  - The WHERE ... = ... effectively ask for a JOIN
  - This JOIN effectively asks for a list intersection
  - If we CREATE an index for the respective tables on the respective join attributes, this list intersection gets fast



Performance 3/3

- Join ordering
  - Typical SQL-from-SPARQL queries require multiple joins
  - Order of joins can make a **huge** performance difference
  - Assume married_to table is small, acted_in table is large
  - Join order 1: look at all married couples and for each get their movies and check whether they overlap

    materializes list of movies of all married people (small)
  - Join order 2: look at all pairs of actors who played in the same movie, and for each check whether they are married

    materialized all pairs of actors from same movie (large)
  - We leave this to the DB engine ... listen to DB lecture for more

SQLite 1/3

- A full-fledged database, easy to install and use
  - Download from http://www.sqlite.org
  - On Debian/Ubuntu install with: sudo apt-get install sqlite3
  - Two types of commands ... examples on next slides

    SQL commands: must end with a semicolon

    SQLite commands: start with a dot, no semicolon at end
  - Two modes to start SQLite:

    sqlite3 — will work on an in-memory database

    sqlite3 <name>.db — create database in that file, and if file exists, use database from that file

SQLite 2/3

- Some useful SQLite commands by example
  - Specifies the column separator used for input and output

    .separator "	" use Ctrl+V TAB for TAB !
  - Read table from TSV (tab-separated values) file

    .import acted_in.tsv acted_in
  - Show execution time of every command

    .timer on
  - Output to file (use stdout for output to console again)

    .output <file name>
  - Execute commands from script file (typical suffix .sql)

    .read <file with commands>

SQLite 3/3

- Some useful SQL commands by example
  - Create a table with a given schema

    CREATE TABLE acted_in (actor TEXT, movie TEXT);
  - Create an index for a column of a table

    CREATE INDEX acted_in_index ON acted_in (actor);
  - Extract / combine data from tables

    SELECT * FROM acted_in WHERE ... LIMIT 100;
  - Delete table / index (without error msg if it's not there)

    DROP TABLE IF EXISTS acted_in;

    DROP INDEX IF EXISTS acted_in_index;

References

- Textbook
  - Nothing about this topic in the IR book by Manning et al !
- Wikipedia
  - http://en.wikipedia.org/wiki/Ontology_(information_science)
  - http://en.wikipedia.org/wiki/SPARQL
  - http://en.wikipedia.org/wiki/SQL
  - http://en.wikipedia.org/wiki/SQLite
  - http://en.wikipedia.org/wiki/Freebase