So how can I ask for it?

Prof. Wolfgang Nejdl

L3S and University of Hannover, Germany

http://ww.l3s.de/
http://ww.kbs.uni-hannover.de/~nejdl

Overview

Searching for something - beyond documents …

1. Searching Personal Collections
2. Keyword Search with Structural Feedback
3. Exploiting Tags for Music Recommendation

1. Searching Personal Collections

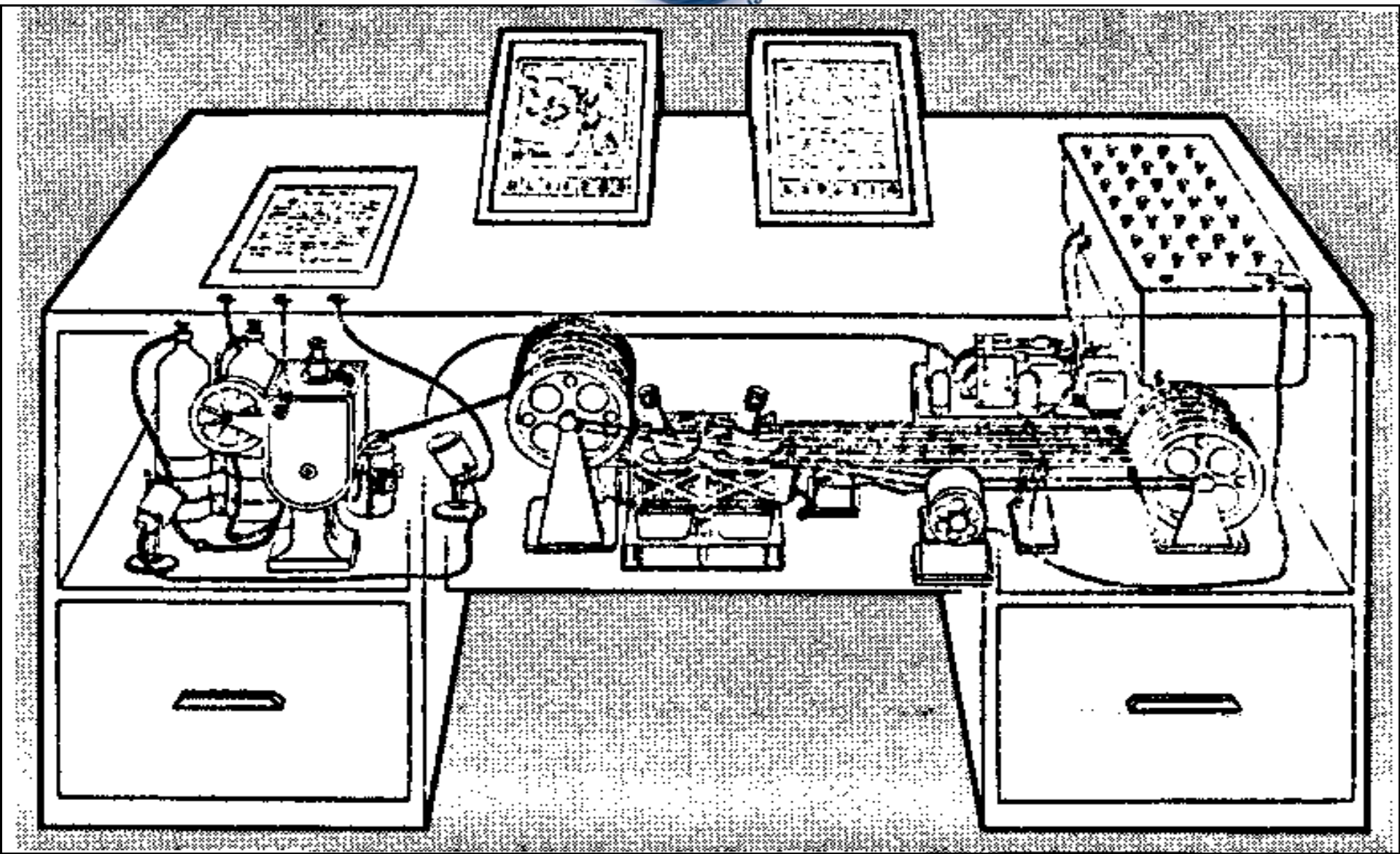
Memex

Posited by Vannevar Bush in "As We May Think"
The Atlantic Monthly, July 1945

"A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility"

Supports: Annotations, links between documents, and "trails" through the documents

"yet if the user inserted 5000 pages of material a day it would take him hundreds of years to fill the repository, so that he can be profligate and enter material freely"



The 1 TB Life (Gordon Bell)

1TB gives you 65+ years of:

- 100 email messages a day (5KB each)
- 100 web pages a day (50KB each)
- 5 scanned pages a day (100KB each)
- 1 book every 10 days (1 MB each)
- 10 photos per day (400 KB JPEG each)
- 8 hours per day of sound - e.g. telephone, voice annotations, and meeting recordings (8 Kb/s)
- 1 new music CD every 10 days (45 min each at 128 Kb/s)

It will take you 10 years to fill up your 160 GB drive

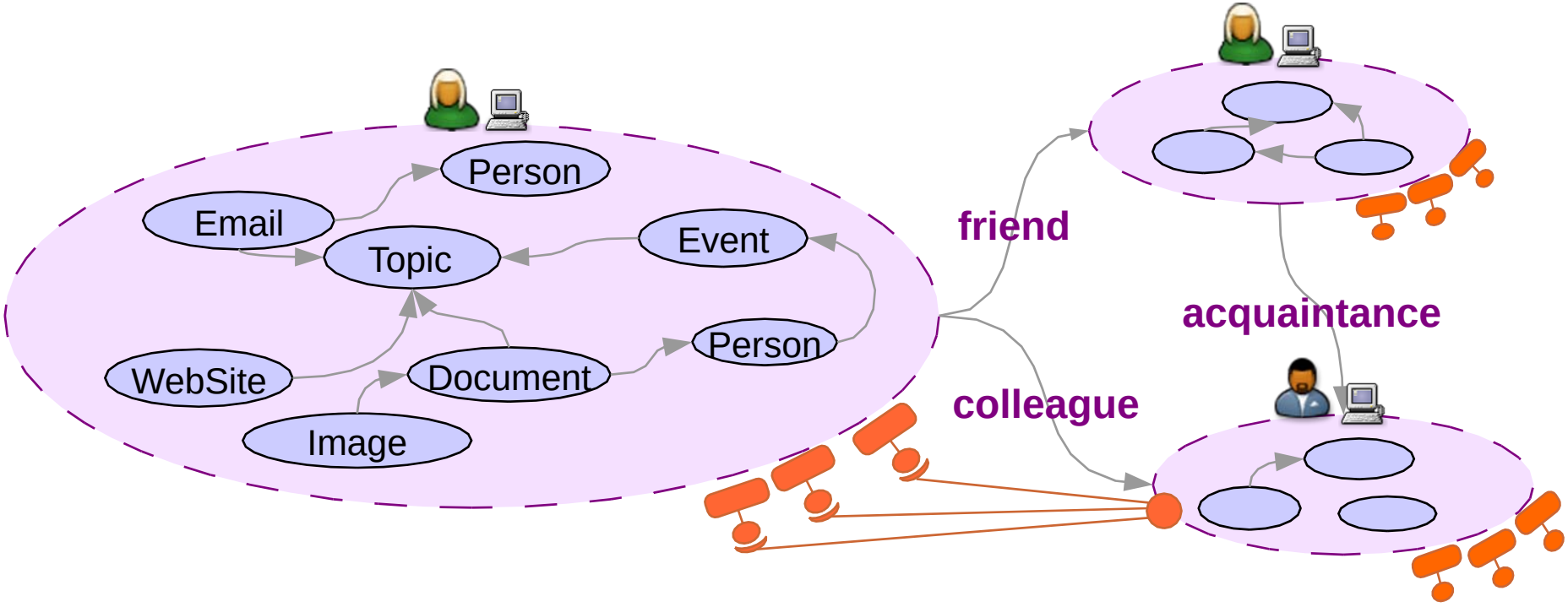
Want video? Buy more cheap drives (1 TB/year lets you record 4 hours/day of 1.5 Mb/s video)

NEPOMUK: Social Semantic Desktop

NEPOMUK - *Desktop*: Help individuals in managing information on their PC

- *Semantic*: Make content available to automated processing

- *Social*: Enable exchange across individual boundaries

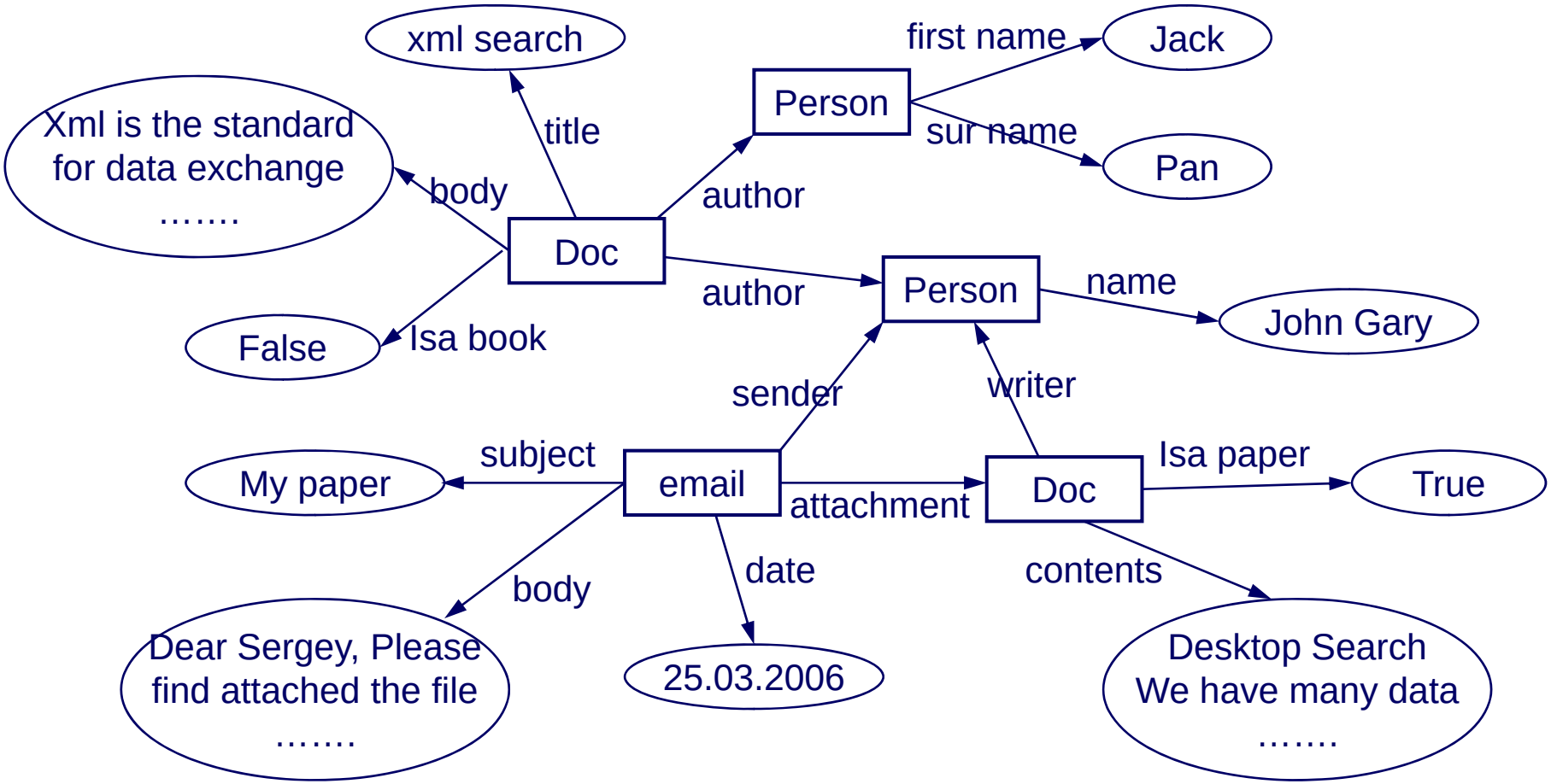


Personal Semantic Web: ***a semantically enlarged intimate supplement to memory***

Social protocols and distributed search

NEPOMUK enabled peers

2. Keyword Search with Structural Feedback



Database Usability [Jagadish SIGMOD 2007]

Challenge: Unknown Schema


```
for   $a
      $s
let   $b
where
   $s/co                    azon" and
     $b/a
return {                              }
```

Conceptually simple real-world objects are normalized into a number of tables

[Jagadish SIGMOD 2007]

Information Retrieval Approach

i.e. Keyword Search

Advantages:

- Intuitive
- Highly Flexible
- Easy to Use

However:

- Ignoring structured information
- Limited expressiveness → Limited ability in catching users' intents
- Example: "essay George Bush"
  Essay *about* George Bush or Essay *written by* George Bush ?

Querying the IMDB Database

- User Intent
  - Movie „Hot Fuzz“, directed by Wright. Action takes place in London.

- Query
  - „Fuzz London Wright“

- Ambiguous Result Examples:
  - “Run Ronnie Run”
    - Actor Wright and a character „Fuzz“
  - “One life to live”
    - Actors: Mary Wright, Cindy London
  - ...

...

- 81 combinations in the IMDB

The SUITS Approach

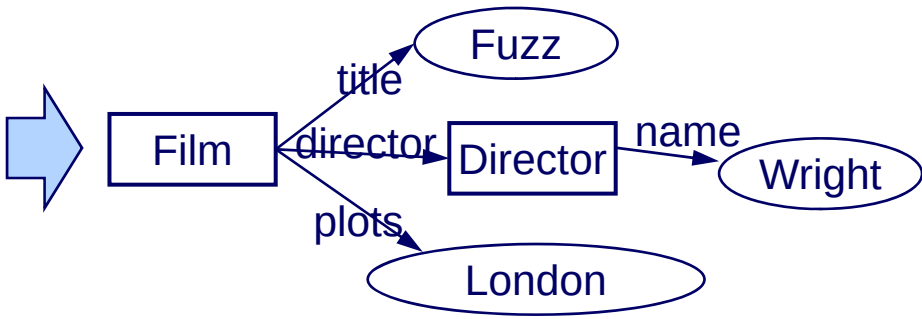
Let user issue a keywords query in the beginning

"Fuzz Wright London"

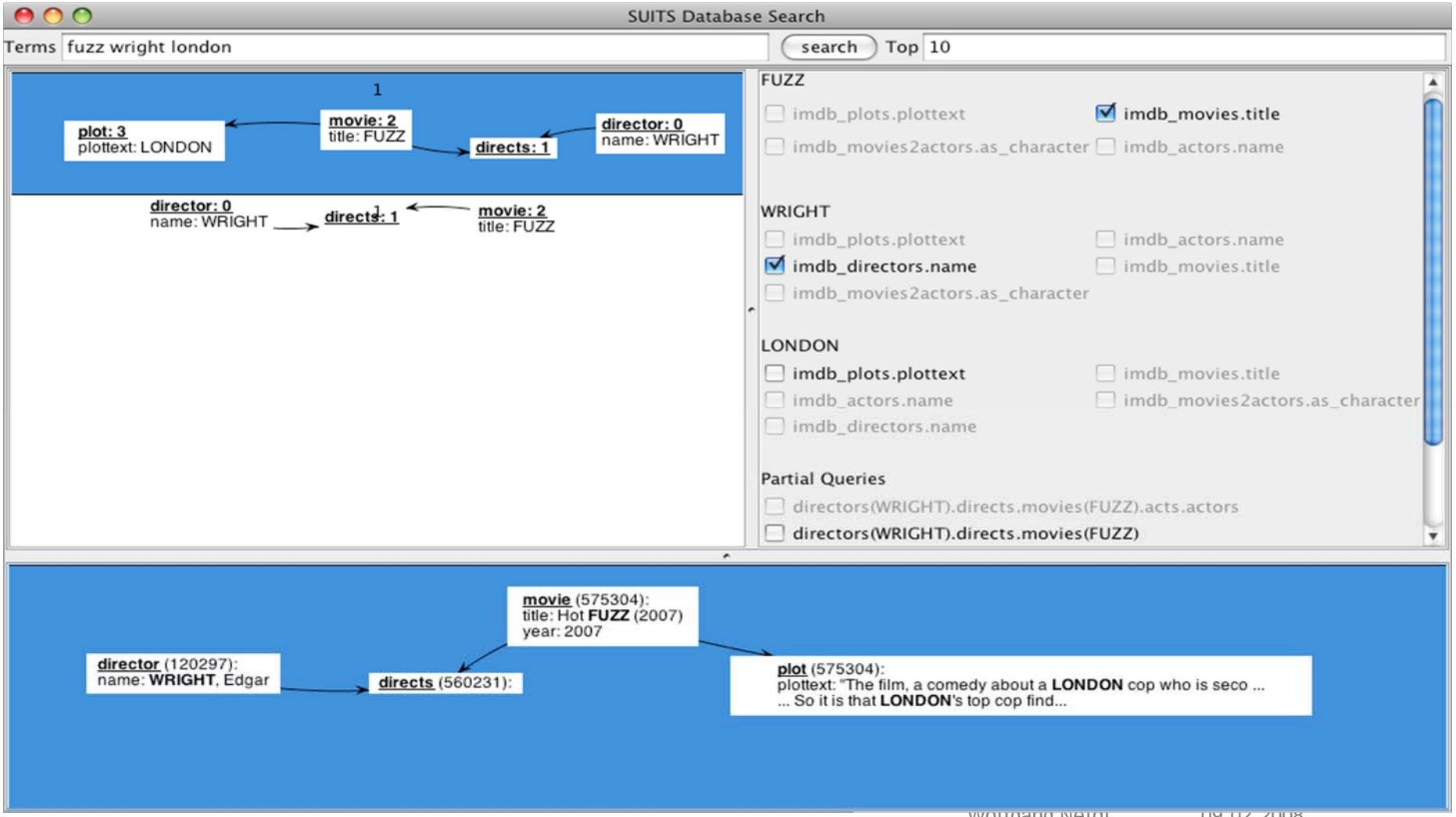
Help user construct a structured query through interactions

System feedback:

Is Fuzz in movie title?	Yes
Is Wright in movie title ?	No
Is Wright a name?	Yes
Is Wright a director's name?	Yes
Are you looking for a film?	Yes
Should London be in plots?	Yes

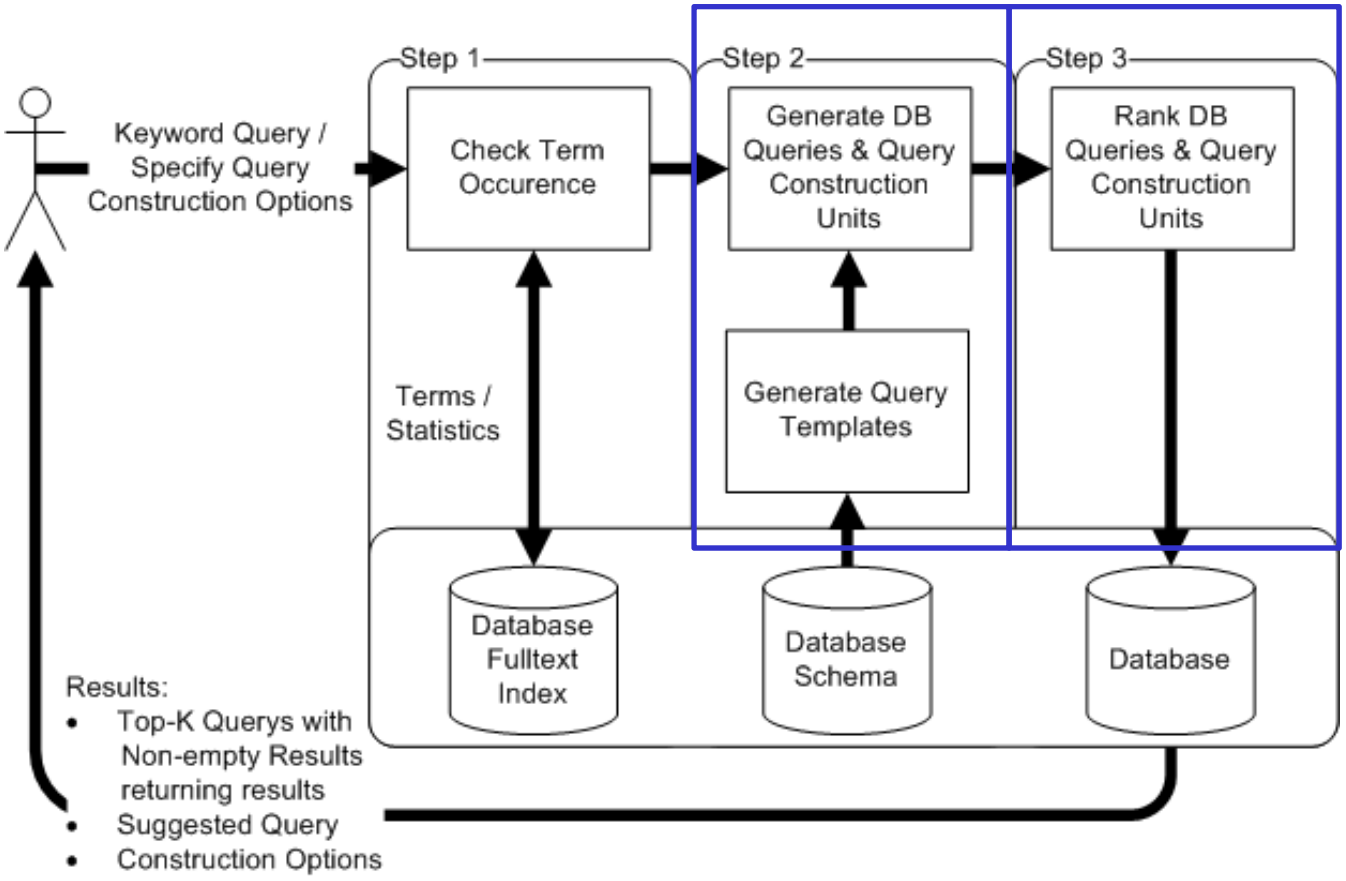


Suits Interface

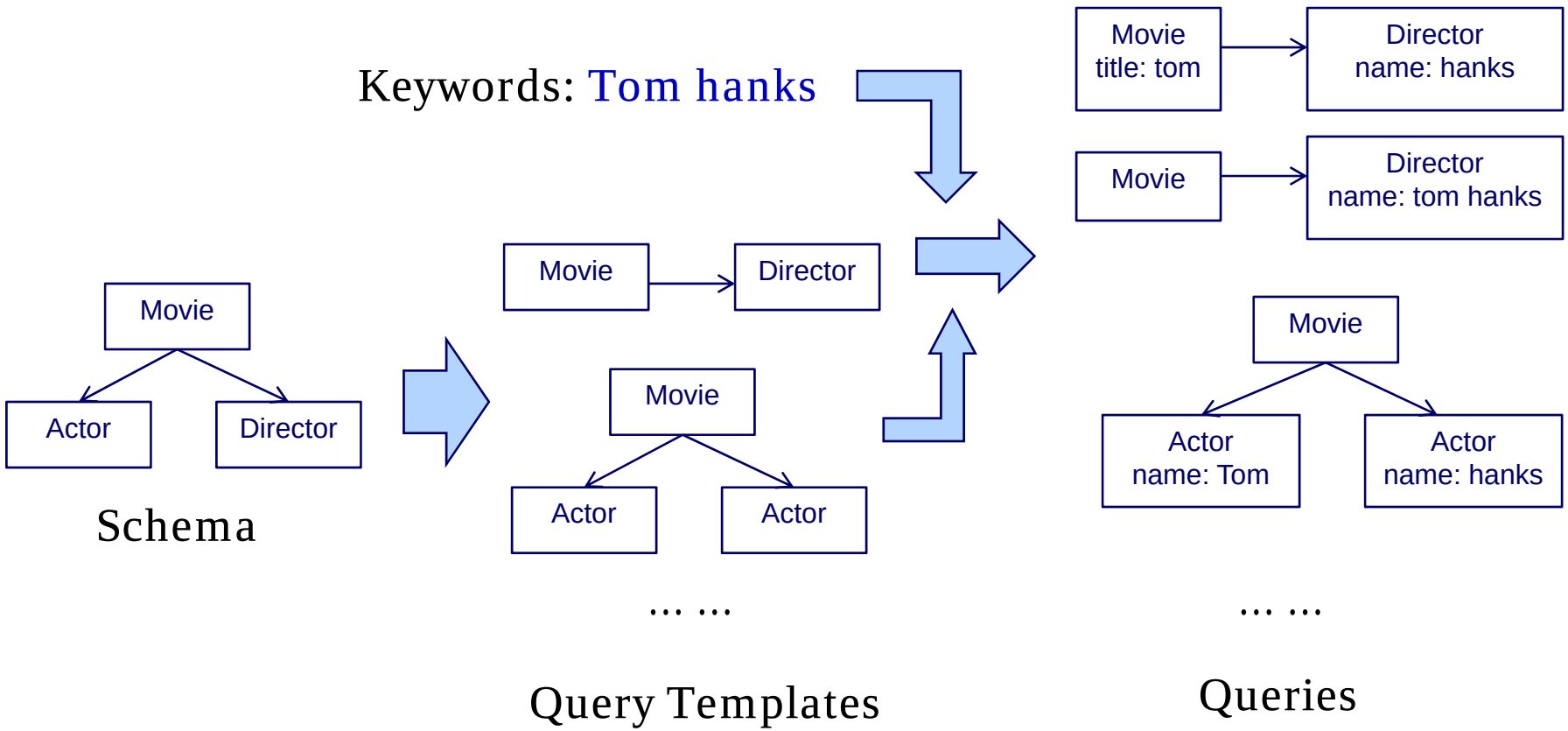


SUITS Architecture

Key Issues



Constructing Structured Queries



Ranking Structured Queries

Rank structured queries base on the likelihood of matching user's intent

Top-k Queries

return the first k structured queries with non-empty results

Ranking function: SER×AC×TC

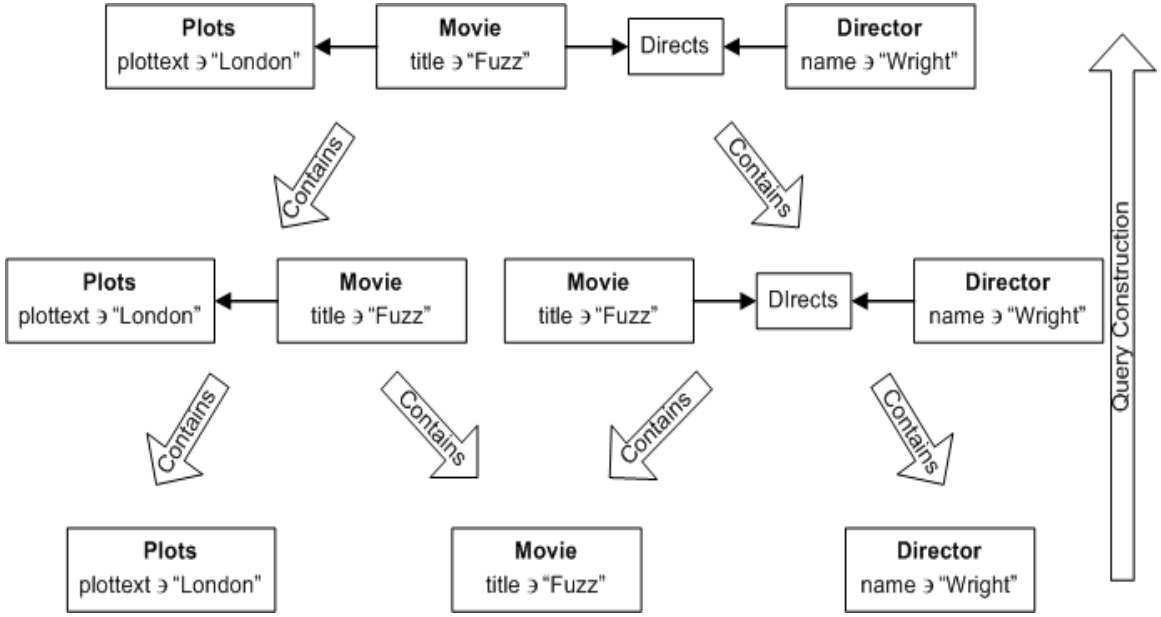
- SER: expected number of results
- AC: attribute completeness
- TC: term completeness

Query Construction Options = Partial Queries

The smallest partial queries are keyword attribute pairs:

e.g. Fuzz: movie.title OR movie.plots OR actor.name
Wright: movie.title OR director.name

User can construct his structured query by climbing the hierarchy of partial queries.



Ranking Partial Queries

When the database schema is big, there will be a big number of query construction options.

return the k options user like most

Ranking function: SEL×AC

- SEL: selectivity
- AC: attribute completeness

Experiment

Datasets:

IMDB

Relation Schema	# Tuples
movies(mID, title, year)	858,967
directs(mID, dID)	572,638
directors(dID, name)	123,178
acts(aID, mID, character)	6,727,186
actors(aID, name)	1,199,918
plots(mID, plottext)	91,565
genres(mID, genre)	637,976
Total number of tuples	10,211,428

Lyrics

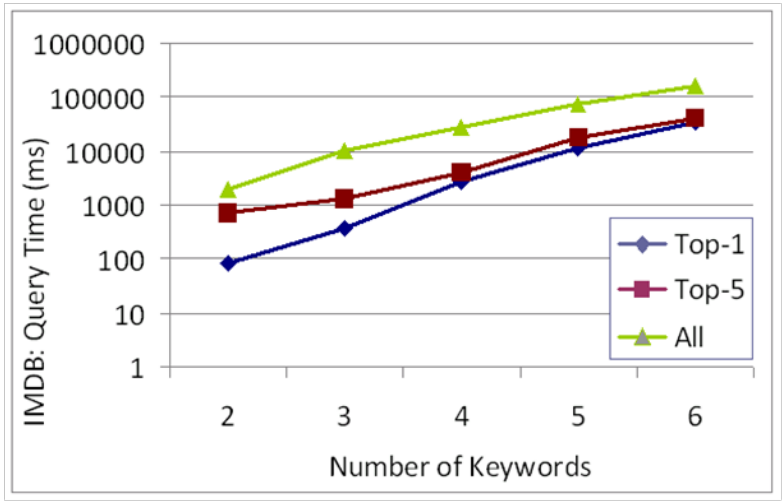
Relation schema	# Tuples
artist(aID, name)	3,691
artistalbum(aID, bID)	15,160
album(bID, title)	15,160
albumsong(bID, sID)	177,231
song(sID, title, lyrics)	177,231
Total number of tuples	388,473

Query sets: the query log of a real search engine

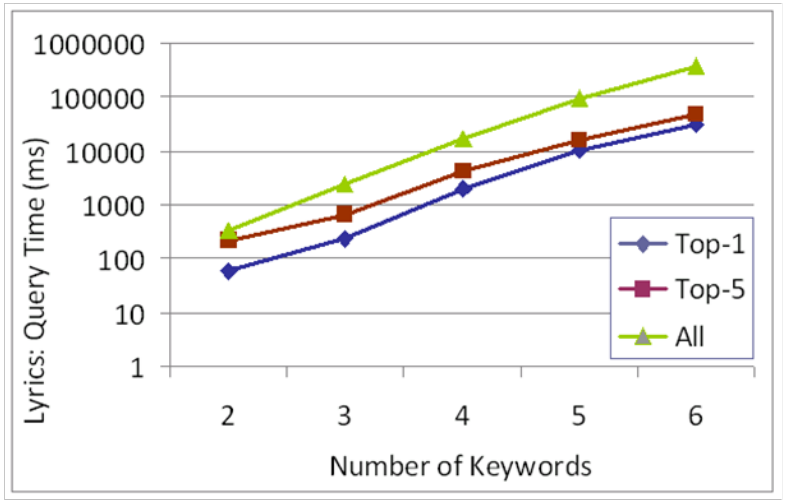
SUITS implemented in Java

Relational Database: MySQL

Performance



IMDB



Lyrics

Observation: I/O dominant, good performance for short and median keyword queries

SUITS - Constructing Structured Queries Using Keywords: Xuan Zhou, Elena Demidova, Gideon Zenz, Wolfgang Nejdl. Technical Report, submitted for publication.

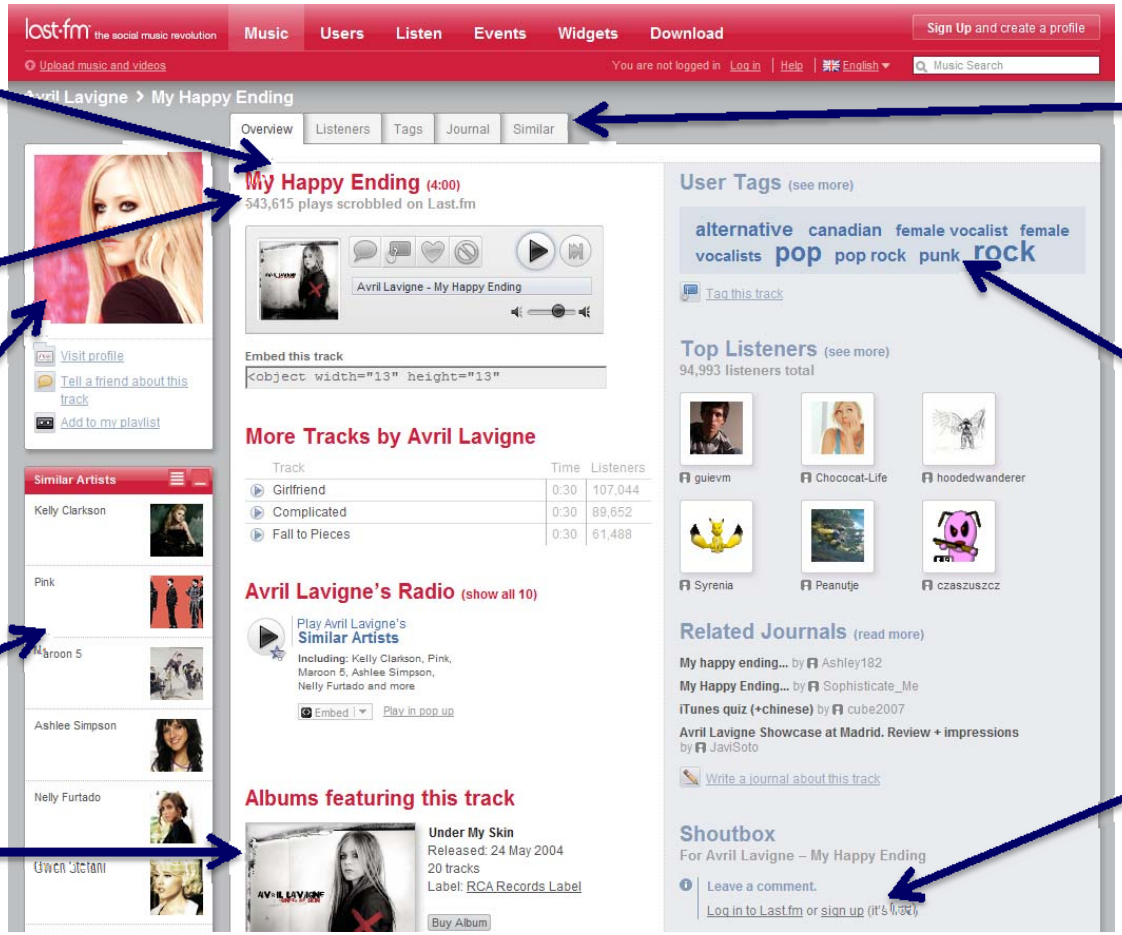
3. Exploring Tags for Music Recommendation

Tags are:

- Written chaotically
- Not verified
- Unstructured
- Heterogeneous
- Unreliable

But if many,
the **correct ones** arise
"Wisdom of the masses"

Last.fm – “The Social Music Revolution”



Track

Track Usage Info

Artist

Similar Artists

Albums

Similar Tracks

Tags (with weight)

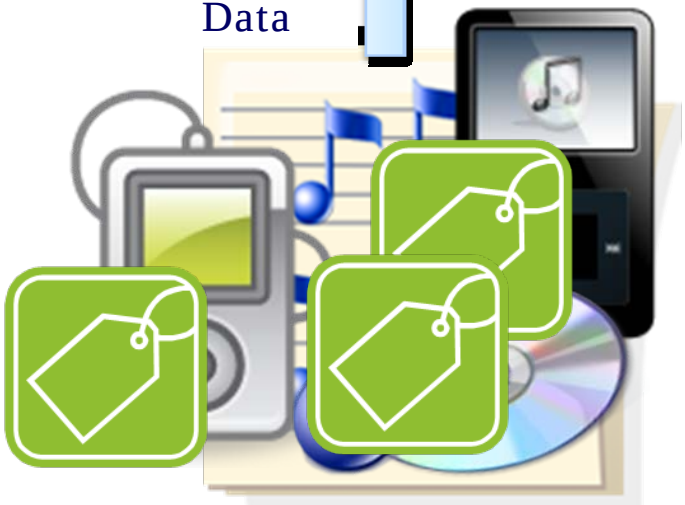
User Comments

Music Recommendation

Personal

Personal Tags

Community Data



Search

Collaborative Filtering

Last.fm Tag Data

21,177 unique tags

- Number of times used
- Number of users who have used each tag
- Tag similarity

In total, tags have been used 18,735,549

60% of the top 100 tags describe a genre

40% : Personal Impressions, Artists, Time Period, Country of Provenance, Soundtrack, Tempo or Instruments

Music Recommendation – Algorithms

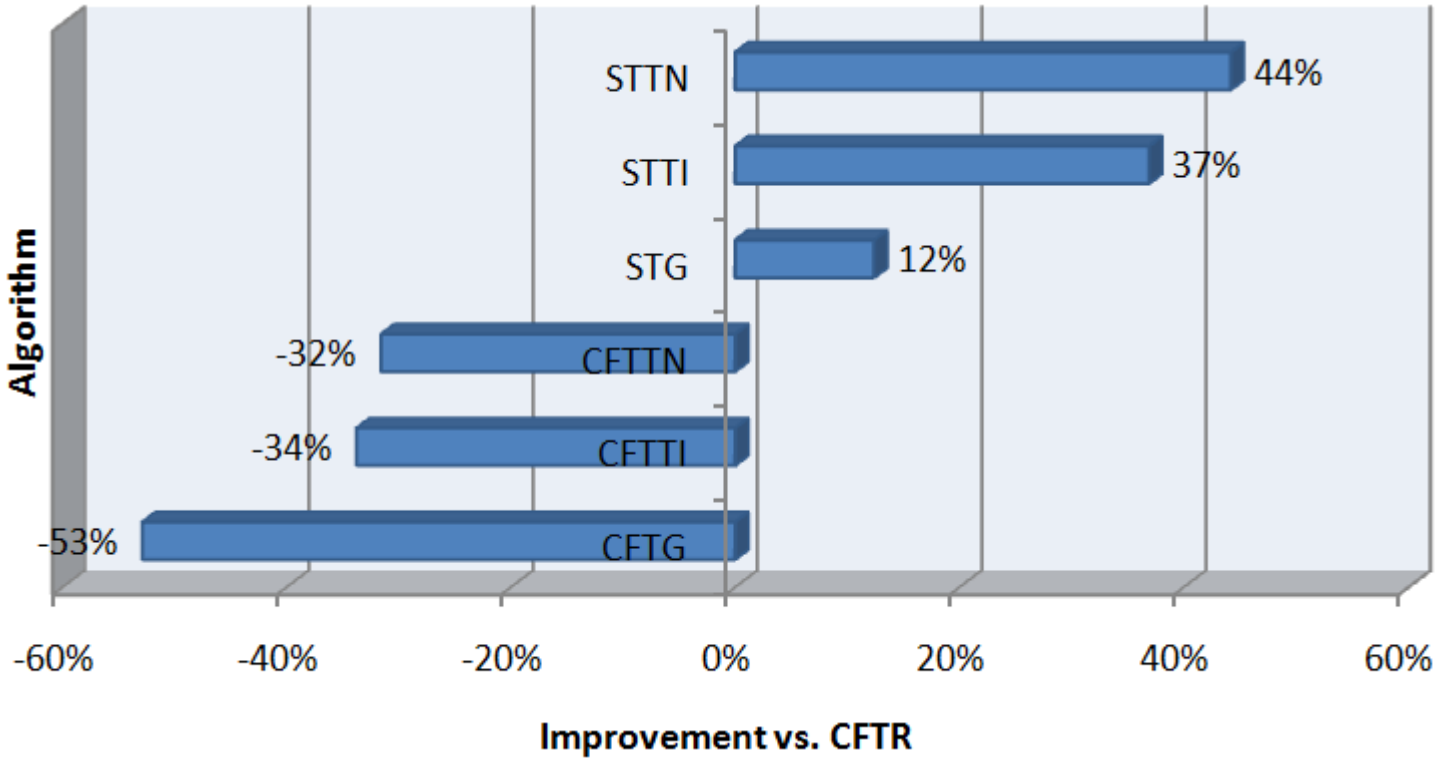
Three types of recommendation algorithms were investigated

- Collaborative Filtering based on Tracks
  Baseline algorithm

- Collaborative Filtering based on Tags
  Recommended tags are used to search for tracks

- Search based on Tags
  Tags in user profile are used directly to search for tracks

Gain over the Baseline (CF on Tracks)



The Benefit of Using Tag-Based Profiles. Claudiu S. Firan, Wolfgang Nejdl and Raluca Paiu. LA-WEB 2007

„Knowing better how to ask for it …“