A Generic Data Model for Schema-driven Design in Information Retrieval Applications

Hany Azzam, and Thomas Roelleke

{hany,thor}@eecs.qmul.ac.uk

Queen Mary University of London

1. Introduction

- Large-scale knowledge bases can be automatically generated from high-quality knowledge sources such as Wikipedia and other semantically explicit data repositories
- These knowledge bases are leveraged by IR application developers to develop more *semantically-aware* retrieval systems
- Developing new applications or incorporating new data formats usually requires "*reimplementing APIs, introducing new query languages, and even introducing new indexing and storage structures*"

2. Motivation

- How diverse applications and data formats can be supported by a *single unifying* framework?
- How techniques developed for a particular data format such as text can be *transferred/extended* to other data formats?

3. Contributions

- A *generic* data model is proposed that represents *facts* (e.g. objects and their relationships) and *content* knowledge (e.g. text in documents) in one congruent data model and can be used to transfer *text* retrieval models such as language modelling to *classification*, *relationship* and *attribute*-based retrieval models
- A simplified *design* that distinguishes between three modelling layers: *basic*, *structural* and *semantic*, which can then be used to derive *application-independent* and *application-specific* schemas

References

[1] R. Cornacchia and A . de Vries. A parameterised search system. In *ECIR*, 2007.

[2] N. Fuhr. Towards data abstraction in networked information retrieval systems. *IP&M*, 35(2):101–119, 1999.

[3] D. Hiemstra and V. Mihajlovic. A database approach to information retrieval: The remarkable relationship between language models and region models. *CTIT Technical Report*, 2010.

[4] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *SIGIR*, 1993.

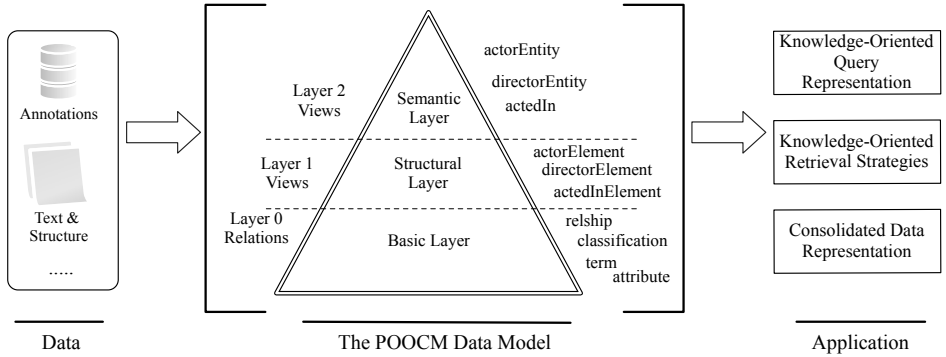
4. The Data Model

- The data model, the Probabilistic Object-Oriented Content Model (POOCM), combines probability theory, object-oriented modelling and content-oriented modelling
- It consists of *classifications*, *relationships*, *attributes* and *terms*, which are all referred to as *propositions*

```
<movie id="329191" title="Gladiator"
    year="2000" genre="action">
   <actor> Russell Crowe </actor>
   <team>  Ridley Scott </team>
   <plot> When a [ARG1 general]
     is [TARGET betrayed by ] a
       [ARG0 prince] ... </plot>
</movie>
```

```
# Term 'gladiator' in movie_329191
0.5 gladiator(movie_329191);
# Classification 'russell_crowe is an actor' in imdb
0.7 actor(russell_crowe, imdb);
# Classification 'maximus is a general' in movie_329191
general(maximus, movie_329191);
# Relationship 'russell_crowe playsRoleOf
# maximus' in movie_329191
playsRoleOf(russell_crowe, maximus, movie_329191);
# Attribute 'movie_329191 has genre action' in imdb
genre(movie_329191, action, imdb);
```

5. Modelling Layers



- *Application-independent* and *application-specific* schemas can be instantiated:
  - **Layer-**0 is the basic, *application-independent layer*
  - **Layer-**1 is the *element-based* layer. It contains rules that can derive structural predicates from the L0. These rules can *lift* the basic classifications and attributes into structural classifications and attributes
  - **Layer-**2 is the *entity-based* layer. It contains rules that derive semantic classification and relationships. For example, the rules extract objects by combining structural information about element types and their attributes
  - Note that L2 relation names bear a meaning, L1 relation names indicate the type of the context, and L0 relation names reflect classifications and relationships
- The layers form an abstraction hierarchy that helps to achieve *data independence*. Any data format (e.g. XML, RDF, text) can be represented
- Indexing and processing strategies developed for one type of an application become *transferable*
- Explicitly stating how the layers are related can *impact* the modelling of probability estimations and aggregations

6. Probability Estimation

- As probabilities are an *inherent* component of the data model, *probabilistic relations* used to develop retrieval models can be estimated for each of the layers
- **L1** consists of relations tailored to modelling the *structure* of data, such as relations for context-based segmentations, e.g. term_doc(Term,Doc) to index documents, and term_sec(Term,Sec) to index sections.
- **L2** comprises of relations reflecting the *semantics* of the data (semantic lifting of L1 leads to L2 relations). For instance, "actedIn(Actor,Movie,Context)" is L2.
- $P_{\text{VF}}(t|i)$: *Value Frequency-based* probability of term t derived from an index i such as "term(Term, Doc)" where the occurrence in documents (values) is the evidence.
- $P_{\text{TF}}(t|i)$: *Tuple Frequency-based* probability of term t derived from an index i where the occurrence in locations (tuples) is the evidence.
- $P_{\text{IVF}}(t|i)$: *Inverse Value Frequency-based* (IVF) probability of term t, e.g. $-\log P_{\text{VF}}(t|i) / \max(\{-\log P_{\text{VF}}(t'|i)\})$. For document retrieval IDF=IVF, and for actor retrieval InvActorFreq=IVF.

7. Benefits

- The data model allows the handling of the *physical* data structures to remain *decoupled* from the rest of the system design and the retrieval models
- The object-oriented and content-oriented concepts provide the ability to *instantiate* retrieval models consisting of term, classification, relationship and attribute propositions
- This *schema-driven* approach helps to *quickly* incorporate different data formats into a *standard* representation and provides models that support *constraint-checking* and *ranking* with respect to these data formats.
- Overall, the data model can improve the *development process* of IR applications