Part II
Entity Retrieval

Entity retrieval
Addressing information needs that are better answered by returning specific objects (entities) instead of just any type of documents.

Distribution of web search queries [Pound et al. 2010]

Distribution of web search queries [Lin et al. 2011]

What’s so special here?
- Entities are not always directly represented
  - Recognize and disambiguate entities in text (that is, entity linking)
  - Collect and aggregate information about a given entity from multiple documents and even multiple data collections
- More structure than in document-based IR
  - Types (from some taxonomy)
  - Attributes (from some ontology)
  - Relationships to other entities ("typed links")

Semantics in our context
- working definition: references to meaningful structures
- How to capture, represent, and use structure?
  - It concerns all components of the retrieval process!

Overview of core tasks

<table>
<thead>
<tr>
<th>Queries</th>
<th>Data set</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(adhoc) entity retrieval</td>
<td>keyword</td>
<td>unstructured/semi-structured</td>
</tr>
<tr>
<td></td>
<td>keyword++</td>
<td>semi-structured</td>
</tr>
<tr>
<td>list completion</td>
<td>keyword++ (example)</td>
<td>semi-structured</td>
</tr>
<tr>
<td>related entity finding</td>
<td>keyword++ (target type, relation)</td>
<td>unstructured &amp; semi-structured</td>
</tr>
</tbody>
</table>

In this part
- Input: keyword(++) query
- Output: a ranked list of entities
- Data collection: unstructured and (semi)structured data sources (and their combinations)
- Main RQ: How to incorporate structure into text-based retrieval models?
**Outline**

1. Ranking based on entity descriptions
2. Incorporating entity types
3. Entity relationships

<table>
<thead>
<tr>
<th>Attributes (/Descriptions)</th>
<th>Type(s)</th>
<th>Relationships</th>
</tr>
</thead>
</table>

**Probabilistic models (mostly)**

- Estimating conditional probabilities
  \[ P(A|B) \]
  \[ P(A, B|C) \]

- Conditional independence
  \[ P(A|B) = P(A) \]
  \[ P(A, B|C) = P(A|C) \cdot P(B|C) \]

- Conditional dependence
  \[ P(A|B) = P(B|A)P(A)/P(B) \]
  \[ P(A, B|C) = P(A|B, C)P(B|C) \]

**Task: ad-hoc entity retrieval**

- **Input:** unconstrained natural language query
- "telegraphic" queries (neither well-formed nor grammatically correct sentences or questions)
- **Output:** ranked list of entities
- **Collection:** unstructured and/or semi-structured documents

**Example information needs**

- american embassy nairobi
- ben franklin
- Chernobyl
- meg ryan war
- Worst actor century
- Sweden Iceland currency

**Two settings**

1. With ready-made entity descriptions

2. Without explicit entity representations

**This is not unrealistic...**
Document-based entity representations

- Most entities have a “home page”
- I.e., each entity is described by a document
- In this scenario, ranking entities is much like ranking documents
  - unstructured
  - semi-structured

Crash course into Language modeling

Example

In the town where I was born,
Lived a man who sailed to sea,
And he told us of his life,
In the land of submarines,
So we sailed on to the sun,
Till we found the sea green,
And we lived beneath the waves,
In our yellow submarine,
We all live in yellow submarine, yellow submarine, yellow submarine.

Empirical document LM

\[
P(t|d) = \frac{n(t,d)}{|d|}
\]

Scoring a query

\[
q = \{\text{sea}, \text{submarine}\}
\]

\[
P(q|d) = P(\text{“sea”}|\theta_d) \cdot P(\text{“submarine”}|\theta_d)
\]

Language Modeling

Standard Language Modeling approach

- Rank documents \(d\) according to their likelihood of being relevant given a query \(q\): \(P(d|q)\)

\[
P(d|q) = \frac{P(q|d)P(d)}{P(q)} \propto P(q|d)P(d)
\]

\[
P(q) = \prod_{t \in q} P(t|\theta_d)^{n(t,q)}
\]

\[
P(t|\theta_d) = (1 - \lambda)P(t|d) + \lambda P(t|C)
\]
**Semi-structured entity representation**

- Entity description documents are rarely unstructured
- Representing entities as
  - Fielded documents – the IR approach
  - Graphs – the DB/SW approach

**Mixture of Language Models**

[Ogilvie & Callan 2003]

- Build a separate language model for each field
- Take a linear combination of them

\[
P(t|\theta_d) = \sum_{j=1}^{m} \mu_j P(t|\theta_d^j)
\]

Field language model

Smoothened with a collection model built from all document representations of the same type in the collection

<table>
<thead>
<tr>
<th>t</th>
<th>P(d)</th>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>submarine</td>
<td>0.14</td>
<td>0.0001</td>
</tr>
<tr>
<td>sea</td>
<td>0.04</td>
<td>0.0002</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Setting field weights**

- **Heuristically**
  - Proportional to the length of text content in that field, to the field’s individual performance, etc.
- **Empirically** (using training queries)
- **Problems**
  - Number of possible fields is huge
  - It is not possible to optimise their weights directly
- **Entities are sparse w.r.t. different fields**
  - Most entities have only a handful of predicates

**Predicate folding**

- **Idea**: reduce the number of fields by grouping them together
- **Grouping based on** (BM25F and)
- **type** ([Pérez-Agüera et al. 2010])
- **manually determined importance** ([Blanco et al. 2011])

**Hierarchical Entity Model**

[Neumayer et al. 2012]

- Organize fields into a 2-level hierarchy
  - Field types (4) on the top level
  - Individual fields of that type on the bottom level
- **Estimate field weights**
  - Using training data for field types
  - Using heuristics for bottom-level types

**Comparison of models**

**Unstructured document model**

**Fielded document model**

**Hierarchical document model**

**Two-level hierarchy**

[Neumayer et al. 2012]

- Extension to the Mixture of Language Models
- Find which document field each query term may be associated with

\[
P(t|\theta_d) = \sum_{j=1}^{m} P(t|d_j) P(\theta_d|t)
\]

**Probabilistic Retrieval Model for Semistructured data**

[Kim et al. 2009]

- Extension to the Mixture of Language Models
- Find which document field each query term may be associated with

\[
P(t|\theta_d) = \sum_{j=1}^{m} P(t|d_j) P(\theta_d|t)
\]

**Example**

- **Genre**: 9.927
- **Cast**: 0.407
- **Title**: 0.187
- **Team**: 0.382
- **Location**: 0.002
**Evaluation initiatives**

- INEX Entity Ranking track (2007-09)
  - Collection is the (English) Wikipedia
  - Entities are represented by Wikipedia articles
- Semantic Search Challenge (2010-11)
  - Collection is a Semantic Web crawl (BTC2009)
  - 1 billion RDF triples
  - Entities are represented by URIs
- INEX Linked Data track (2012-13)
  - Wikipedia enriched with RDF properties from DBpedia and YAGO

**Ranking without explicit entity representations**

**Scenario**

- Entity descriptions are not readily available
- Entity occurrences are annotated
  - manually
  - automatically (~entity linking)

**The basic idea**

Use documents to go from queries to entities

**Two principal approaches**

- **Profile-based methods**
  - Create a textual profile for entities, then rank them (by adapting document retrieval techniques)
- **Document-based methods**
  - Indirect representation based on mentions identified in documents
  - First ranking documents (or snippets) and then aggregating evidence for associated entities

**Profile-based methods**

**Many possibilities in terms of modeling**

- Generative (probabilistic) models
- Discriminative (probabilistic) models
- Voting models
- Graph-based models
**Generative probabilistic models**

- Candidate generation models ($P(e|q)$)
  - Two-stage language model
- Topic generation models ($P(q|e)$)
  - Candidate model, a.k.a. Model 1
  - Document model, a.k.a. Model 2
  - Proximity-based variations
- Both families of models can be derived from the Probability Ranking Principle [Fang & Zhai 2007]

**Candidate models (“Model 1”)**

[Balog et al. 2006]

$$P(q|e) = \prod_{t \in q} P(t|e)^{n(t,e)}$$

Smoothing with collection-wide background model

$$(1 - \lambda)P(t|e) + \lambda P(t)$$

Term-candidate co-occurrence in a particular document.

In the simplest case: $P(t|d)$

**Document models (“Model 2”)**

[Balog et al. 2006]

$$P(q|e) = \sum_d P(q|d, e)P(d|e)$$

Document relevance

How well document supports the claim that $e$ is relevant to $q$

$$\prod_{t \in q} P(t|d, e)^{n(t,e)}$$

Simplifying assumption $t$ and $e$ are conditionally independent given $d$

$P(t|\theta_d)$

**Document-entity associations**

- Boolean (or set-based) approach
- Weighted by the confidence in entity linking
- Consider other entities mentioned in the document

**Proximity-based variations**

- So far, conditional independence assumption between candidates and terms when computing the probability $P(t|d,e)$
- Relationship between terms and entities that in the same document is ignored
  - Entity is equally strongly associated with everything discussed in that document
  - Let's capture the dependence between entities and terms
  - Use their distance in the document

**Using proximity kernels**

[Petkova & Croft 2007]

$$P(t|d,e) = \frac{1}{Z} \sum_{i=1}^{N} \delta_{i}(t,e) k(t,e)$$

Normalizing constant

Indicator function

1 if the term at position $i$ is $t$, 0 otherwise

**Many possibilities in terms of modeling**

- Generative probabilistic models
- Discriminative probabilistic models
- Voting models
- Graph-based models

Figure taken from D. Petkova and W.B. Croft. Proximity-based document representation for named entity retrieval. CIKM’07.
Infinite random walk

\[ P_i(d) = \lambda P_j(d) + (1 - \lambda) \sum_{e \rightarrow d} P(d|e)P_{i-1}(e), \]

\[ P_i(e) = \sum_{d \rightarrow e} P(e|d)P_{i-1}(d), \]

\[ P_j(d) = P(d,q), \]

Discriminative models

- Vs. generative models:
  - Fewer assumptions (e.g., term independence)
  - "Let the data speak"
  - Sufficient amounts of training data required
  - Incorporating more document features, multiple signals for document-entity associations
  - Estimating \( P(r=1|e,q) \) directly (instead of \( P(e,q|r=1) \))
  - Optimization can get trapped in a local maximum/minimum

\[ P_i(d) = \sum_{e} P_i(e)P(d|e)P_{i-1}(e), \]

\[ P_i(e) = \sum_{d} P(e|d)P_{i-1}(d), \]

\[ P_J(d) = P(d|q), \]

Arithmetic Mean

Discriminative (AMD) model

\[ P_0(r = 1|e, q) = \sum_d P(r_1 = 1|q, d)P(r_2 = 1|e, d)P(d) \]

Voting models

[Macdonald & Ounis 2006]

- Inspired by techniques from data fusion
  - Combining evidence from different sources
- Documents ranked w.r.t. the query are seen as "votes" for the entity

\[ Score(e, q) = \sum_{d \rightarrow e} \frac{1}{\text{rank}(d, q)} \]

\[ Score(e, q) = |\{M(e) \cup R(q)\}| \sum_{d \rightarrow e} s(d, q) \]

Graph-based models

[Serdyukov et al. 2008]

- One particular way of constructing graphs
  - Vertices are documents and entities
  - Only document-entity edges
- Search can be approached as a random walk on this graph
  - Pick a random document or entity
  - Follow links to entities or other documents
  - Repeat it a number of times

Evaluation

- Expert finding task @TREC Enterprise track
  - Enterprise setting (intranet of a large organization)
  - Given a query, return people who are experts on the query topic
  - List of potential experts is provided
- We assume that the collection has been annotated with <person>...</person> tokens

Learning to rank & entity retrieval

- Pointwise
  - AMD, GMD [Yang et al. 2010]
  - Multilayer perceptrons, logistic regression [Sorg & Cimiano 2011]
  - Additive Groves [Moreira et al. 2011]
- Pairwise
  - Ranking SVM [Yang et al. 2009]
  - RankBoost, RankNet [Moreira et al. 2011]
- Listwise
  - AdaRank, Coordinate Ascent [Moreira et al. 2011]

Discriminative models

- Vs. generative models:
  - Fewer assumptions (e.g., term independence)
  - "Let the data speak"
  - Sufficient amounts of training data required
  - Incorporating more document features, multiple signals for document-entity associations
  - Estimating \( P(r=1|e,q) \) directly (instead of \( P(e,q|r=1) \))
  - Optimization can get trapped in a local maximum/minimum

\[ P_i(d) = \sum_{e} P_i(e)P(d|e)P_{i-1}(e), \]

\[ P_i(e) = \sum_{d} P(e|d)P_{i-1}(d), \]

\[ P_J(d) = P(d|q), \]

\[ Score(e, q) = \sum_{d \rightarrow e} \frac{1}{\text{rank}(d, q)} \]

\[ Score(e, q) = |\{M(e) \cup R(q)\}| \sum_{d \rightarrow e} s(d, q) \]
### Incorporating entity types

<table>
<thead>
<tr>
<th>Attributes / (Descriptions)</th>
<th>Type(s)</th>
<th>Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>people</td>
<td>organizations</td>
</tr>
<tr>
<td></td>
<td>locations</td>
<td>products</td>
</tr>
</tbody>
</table>

### Interacting with types

#### Grouping results
- Filtering results

#### Filtering results
- eBay
- Amazon

### Type-aware ranking

- Typically, a two-component model:

\[
P(q|e) = P(q_T, q_t|e) = P(q_T|e)P(q_t|e)
\]

#### Target type

- Provided by the user
  - keyword++ query
- Need to be automatically identified
  - keyword query

#### Target type(s) are provided
- faceted search, form fill-in, etc.
But what about very many types? which are typically hierarchically organized

Challenges
- Users are not familiar with the type system

In general, categorizing things can be hard
- What is King Arthur?
  - Person / Royalty / British royalty
  - Person / Military person
  - Person / Fictional character

Which King Arthur?!

Upshot for type-aware ranking
- Need to be able to handle the imperfections of the type system
  - Inconsistencies
  - Missing assignments
  - Granularity issues
    - Entities labeled with too general or too specific types
  - User input is to be treated as a hint, not as a strict filter

Two settings
- Target type(s) are provided by the user
  - keyword++ query
- Target types need to be automatically identified
  - keyword query

Identifying target types for queries
- Types can be ranked much like entities
  [Balog & Neumayer 2012]
  - Direct term-based representations (“Model 1”)
  - Types of top ranked entities (“Model 2”)
  [Vallet & Zaragoza 2008]

Type-centric vs. entity-centric type ranking
**Hierarchical target type identification**
- Finding the single most specific type [from an ontology] that is general enough to cover all entities that are relevant to the query.
- Finding the right granularity is difficult...
  - Models are good at finding either general (top-level) or specific (leaf-level) types

**Type-based similarity**
\[
P(q | e) = P(qr | e) = P(q | e) E
\]
- Measuring similarity
  - Set-based
  - Content-based (based on type labels)
- Need “soft” matching to deal with the imperfections of the category system
  - Lexical similarity of type labels
  - Distance based on the hierarchy
  - Query expansion

**Modeling types as probability distributions** [Balog et al. 2011]
- Analogously to term-based representations

- Advantages
  - Sound modeling of uncertainty associated with category information
  - Category-based feedback is possible

**Joint type detection and entity ranking** [Sawant & Chakrabarti 2013]
- Assumes “telegraphic” queries with target type
  - woodrow wilson president university
  - dolly clone institute
  - lead singer led zeppelin band
- Type detection is integrated into the ranking
  - Multiple query interpretations are considered
- Both generative and discriminative formulations

**Approach**
- Each query term is either a “type hint” \(h(q, \tilde{z})\) or a “word matcher” \(s(q, \tilde{z})\)
- Number of possible partitions is manageable \(2^{|q|}\)

**Generative approach**
Generate query from entity

**Discriminative approach**
Separate correct and incorrect entities

**Generative formulation**
\[
P(e | q) \propto P(e) \sum_{t, \tilde{z}} P(t | e)P(\tilde{z} | e)P(h(t, \tilde{z}) | e)P(s(t, \tilde{z}) | e)
\]
- Type model
  - Estimated from answer types in the past
- Entity prior
  - Probability of observing \(t\) in the type model
- Type prior
  - Probability of observing \(t\) in the entity model

**Joint type detection and response ranking**
- San Diego Padres
  - Major league baseball team
  - Padres have been to two World Series, losing in 1984 and 1998

- World Series, losing in 1984 and 1998
- San Diego Padres
  - Major league baseball team
  - Padres have been to two World Series, losing in 1984 and 1998

**Type-based similarity**
- Measuring similarity
  - Set-based
  - Content-based (based on type labels)
- Need “soft” matching to deal with the imperfections of the category system
  - Lexical similarity of type labels
  - Distance based on the hierarchy
  - Query expansion

**Modeling types as probability distributions** [Balog et al. 2011]
- Analogously to term-based representations

- Advantages
  - Sound modeling of uncertainty associated with category information
  - Category-based feedback is possible

**Joint type detection and entity ranking** [Sawant & Chakrabarti 2013]
- Assumes “telegraphic” queries with target type
  - woodrow wilson president university
  - dolly clone institute
  - lead singer led zeppelin band
- Type detection is integrated into the ranking
  - Multiple query interpretations are considered
- Both generative and discriminative formulations

**Approach**
- Each query term is either a “type hint” \(h(q, \tilde{z})\) or a “word matcher” \(s(q, \tilde{z})\)
- Number of possible partitions is manageable \(2^{|q|}\)

**Generative approach**
Generate query from entity

**Discriminative approach**
Separate correct and incorrect entities

**Generative formulation**
\[
P(e | q) \propto P(e) \sum_{t, \tilde{z}} P(t | e)P(\tilde{z} | e)P(h(t, \tilde{z}) | e)P(s(t, \tilde{z}) | e)
\]
- Type model
  - Estimated from answer types in the past
- Entity prior
  - Probability of observing \(t\) in the type model
- Type prior
  - Probability of observing \(t\) in the entity model

**Joint type detection and response ranking**
- San Diego Padres
  - Major league baseball team
  - Padres have been to two World Series, losing in 1984 and 1998

- World Series, losing in 1984 and 1998
- San Diego Padres
  - Major league baseball team
  - Padres have been to two World Series, losing in 1984 and 1998

**Type-based similarity**
- Measuring similarity
  - Set-based
  - Content-based (based on type labels)
- Need “soft” matching to deal with the imperfections of the category system
  - Lexical similarity of type labels
  - Distance based on the hierarchy
  - Query expansion
Discriminative formulation

\[
\phi(q, e, t, \sim z) = (\phi_1(q, e), \phi_2(t, e), \phi_3(q, \sim z, t), \phi_4(q, \sim z, e))
\]

Models the type prior \(P(t|e)\)
Models the entity prior \(P(e)\)
Comparability between hint words and type
Comparability between matchers and snippets that mention \(e\)

Evaluation

- INEX Entity Ranking track
- Entities are represented by Wikipedia articles
- Topic definition includes target categories

Movies with eight or more Academy Awards
best picture oscar british films american films

Entity relationships

Searching for arbitrary relations*

*given an input entity and target type

airlines that currently use Boeing 747 planes
ORG Boeing 747

Members of The Beaux Arts Trio
PER The Beaux Arts Trio

What countries does Eurail operate in?
LOC Eurail
Modeling related entity finding
[Bron et al. 2010]

- Ranking entities of a given type (T) that stand in a required relation (R) with an input entity (E)
- Three-component model

\[
p(e|E,T,R) \propto p(e|E) \cdot p(T|e) \cdot p(R|E,e)
\]

Evaluation

- TREC Entity track
- Given
  - Input entity (defined by name and homepage)
  - Type of the target entity (PER/ORG/LOC)
  - Narrative (describing the nature of the relation in free text)
- Return (homepages of) related entities

Wrapping up

- Entity retrieval in different flavors using generative approaches based on language modeling techniques
- Increasingly more discriminative approaches over generative ones
  - Increasing amount of components (and parameters)
  - Easier to incrementally add informative but correlated features
  - But, (massive amounts of) training data is required

Future challenges

- It’s “easy” when the “query intent” is known
  - Desired results: single entity, ranked list, set, …
  - Query type: ad-hoc, list search, related entity finding, …
- Methods specifically tailored to specific types of requests
- Understanding query intent still has a long way to go