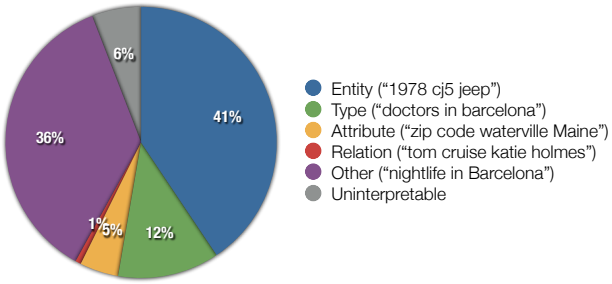
Part II
Entity Retrieval

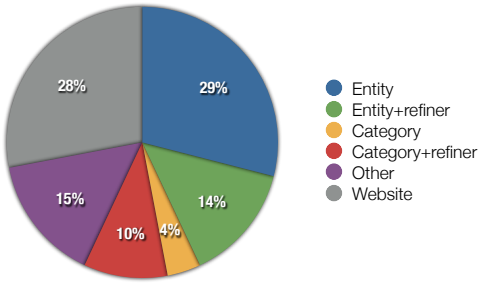
Entity retrieval

*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]

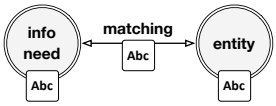


What's so special here?

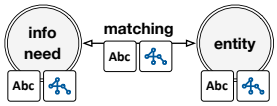
- Entities are not always directly represented
  - Recognize and disambiguate entities in text (that is, entity linking)
  - Collect and aggregate information about a given entity from multiple documents and even multiple data collections
- More structure than in document-based IR
  - Types (from some 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!



Text-only representation



Text+structure representation

Overview of core tasks

	Queries	Data set	Results
(adhoc) entity retrieval	keyword	unstructured/ semi-structured	ranked list
	keyword++ (target type(s))	semi-structured	ranked list
list completion	keyword++ (examples)	semi-structured	ranked list
related entity finding	keyword++ (target type, relation)	unstructured & semi-structured	ranked list

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

Attributes (/Descriptions)

Type(s)

Relationships

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

$$P(A|B) = P(B|A)P(A)/P(B)$$
$$P(A,B|C) = P(A|B,C)P(B|C)$$

Ranking entity descriptions

Attributes (/Descriptions)

Type(s)

Relationships

Task: ad-hoc entity retrieval

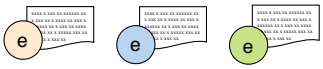
- **Input**: unconstrained natural language query
  - "telegraphic" queries (neither well-formed nor grammatically correct sentences or questions)
- **Output**: ranked list of entities
- **Collection**: unstructured and/or semi-structured documents

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

Ranking with ready-made entity descriptions

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

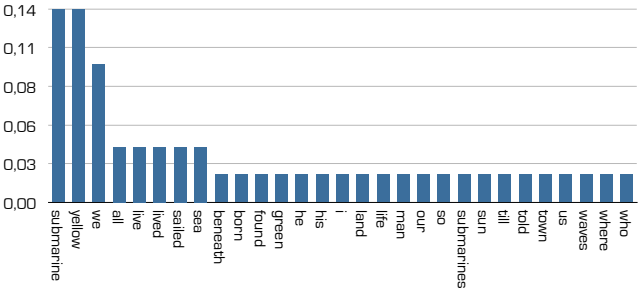
In the town where I was born,
Lived a man who sailed to sea,
And he told us of his life,
In the land of submarines,

So we sailed on to the sun,
Till we found the sea green,
And we lived beneath the waves, In our yellow submarine,

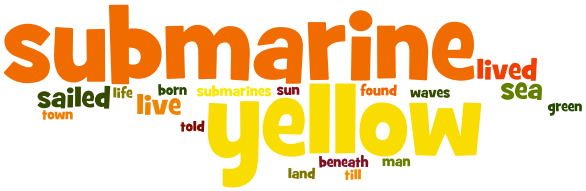
We all live in yellow submarine, yellow submarine, yellow submarine, We all live in yellow submarine, yellow submarine, yellow submarine.

Empirical document LM

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



Alternatively...

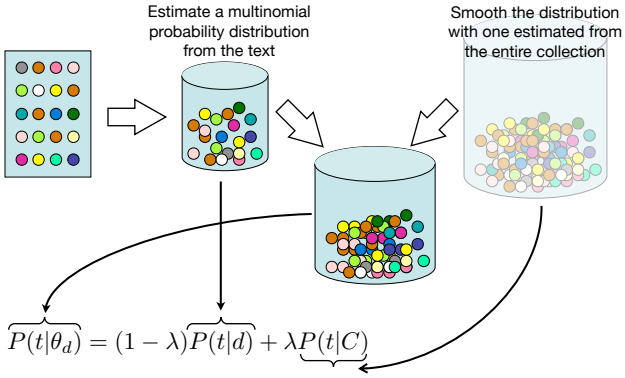


Scoring a query

$q = \{\text{sea}, \text{submarine}\}$

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

Language Modeling



Standard Language Modeling approach

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

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

Query likelihood
Probability that query q was “produced” by document d

Document prior
Probability of the document being relevant to *any* query

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

Standard Language Modeling approach (2)

$$P(q|d) = \prod_{t \in q} \underbrace{P(t|\theta_d)}_{\text{Document language model}}^{\overbrace{n(t,q)}^{\text{Number of times } t \text{ appears in } q}}$$

Document language model
Multinomial probability distribution over the vocabulary of terms

Smoothing parameter

$$P(t|\theta_d) = (1-\lambda)\underbrace{P(t|d)}_{\text{Empirical document model}} + \lambda \underbrace{P(t|C)}_{\text{Collection model}}$$

Maximum likelihood estimates

$$\frac{n(t,d)}{|d|} \qquad \frac{\sum_d n(t,d)}{\sum_d |d|}$$

Scoring a query

$$q = \{\text{sea}, \text{submarine}\}$$

$$P(q|d) = \underbrace{P(\text{“sea”}|\theta_d)}_{0.03602} \cdot P(\text{“submarine”}|\theta_d)$$

$$\overset{0.9}{(1-\lambda)}\overset{0.04}{P(\text{“sea”}|d)} + \overset{0.1}{\lambda}\overset{0.0002}{P(\text{“sea”}|C)}$$

t	P(t|d)
submarine	0,14
sea	0,04
...	

t	P(t|C)
submarine	0,0001
sea	0,0002
...	

Scoring a query

$$q = \{\text{sea}, \text{submarine}\}$$

$$\overset{0.04538}{P(q|d)} = \overset{0.03602}{P(\text{“sea”}|\theta_d)} \cdot \overset{0.12601}{P(\text{“submarine”}|\theta_d)}$$

$$\overset{0.9}{(1-\lambda)}\overset{0.14}{P(\text{“submarine”}|d)} + \overset{0.1}{\lambda}\overset{0.0001}{P(\text{“submarine”}|C)}$$

t	P(t|d)
submarine	0,14
sea	0,04
...	

t	P(t|C)
submarine	0,0001
sea	0,0002
...	

Here, documents==entities, so

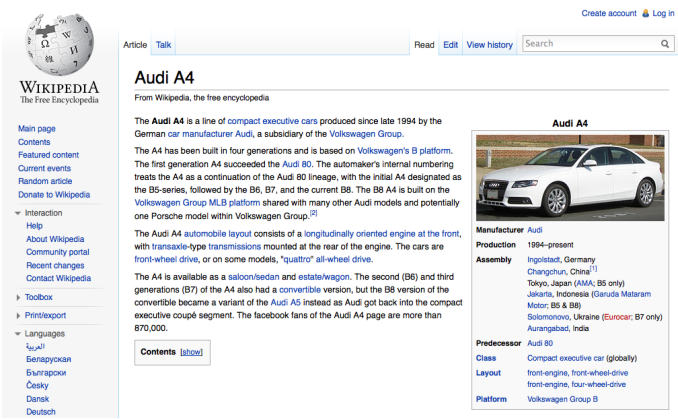
$$P(e|q) \propto P(e)P(q|\theta_e) = \underbrace{P(e)}_{\text{Entity prior}} \prod_{t \in q} \underbrace{P(t|\theta_e)}_{\text{Entity language model}}^{n(t,q)}$$

Entity prior
Probability of the entity being relevant to *any* query

Entity language model
Multinomial probability distribution over the vocabulary of terms

Semi-structured entity representation

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




```
dbpedia:Audi_A4

foaf:name                        Audi A4
rdfs:label                       Audi A4
rdfs:comment                     The Audi A4 is a compact executive car
                                 produced since late 1994 by the German car
                                 manufacturer Audi, a subsidiary of the
                                 Volkswagen Group. The A4 has been built [...]
dbpprop:production               1994
                                 2001
                                 2005
                                 2008
rdf:type                         dbpedia-owl:MeanOfTransportation
                                 dbpedia-owl:Automobile
dbpedia-owl:manufacturer         dbpedia:Audi
dbpedia-owl:class                dbpedia:Compact_executive_car
owl:sameAs                       freebase:Audi_A4
is dbpedia-owl:predecessor of    dbpedia:Audi_A5
is dbpprop:similar of            dbpedia:Cadillac_BLS
```

Mixture of Language Models

[Ogilvie & Callan 2003]

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

$$P(t|\theta_d) = \sum_{j=1}^{m} \underbrace{\mu_j}_{\text{Field weights}} P(t|\underbrace{\theta_{d_j}}_{\text{Field language model}})$$

Field weights
$\sum_{j=1}^{m} \mu_j = 1$

Field language model
Smoothed with a collection model built from all document representations of the same type in the collection

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]

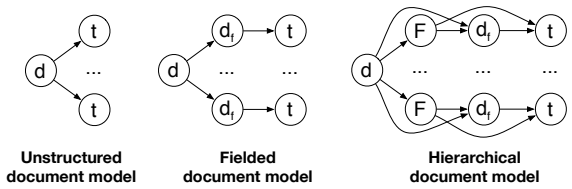
- Organize fields into a 2-level hierarchy
  - Field types (4) on the top level
  - Individual fields of that type on the bottom level
- Estimate field weights
  - Using training data for field types
  - Using heuristics for bottom-level types

Two-level hierarchy

[Neumayer et al. 2012]

Name	foaf:name	Audi A4
	rdfs:label	Audi A4
Attributes	rdfs:comment	The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group. The A4 has been built [...]
	dbpprop:production	1994 2001 2005 2008
Out-relations	rdf:type	dbpedia-owl:MeanOfTransportation dbpedia-owl:Automobile
	dbpedia-owl:manufacturer	dbpedia:Audi
	dbpedia-owl:class	dbpedia:Compact_executive_car
	owl:sameAs	freebase:Audi_A4
In-relations	is dbpedia-owl:predecessor of	dbpedia:Audi_A5
	is dbpprop:similar of	dbpedia:Cadillac_BLS

Comparison of models



Unstructured document model **Fielded document model** **Hierarchical document model**

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

$$P(t|\theta_d) = \sum_{j=1}^{m} \mu_j P(t|\theta_{d_j})$$

Mapping probability
Estimated for each query term

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

Estimating the mapping probability

$$P(t|C_j) = \frac{\sum_d n(t, d_j)}{\sum_d |d_j|}$$

Term likelihood
Probability of a query term occurring in a given field type

Prior field probability
Probability of mapping the query term to this field before observing collection statistics

$$P(d_j|t) = \frac{P(t|d_j)P(d_j)}{P(t)}$$

$$P(t) = \sum_{d_k} P(t|d_k)P(d_k)$$

Example

meg ryan war

d_j	$P(t\|d_j)$
cast	0,407
team	0,382
title	0,187

d_j	$P(t\|d_j)$
cast	0,601
team	0,381
title	0,017

d_j	$P(t\|d_j)$
genre	0,927
title	0,07
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

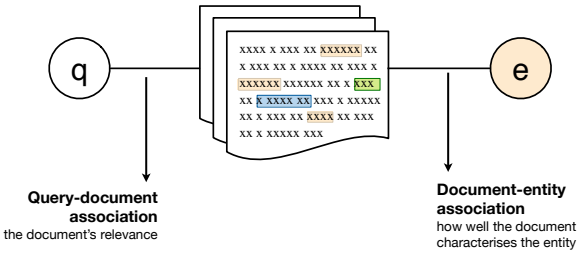
Ranking without explicit entity representations

Scenario

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

The basic idea

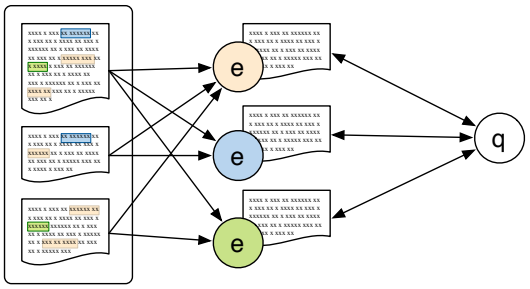
Use documents to go from queries to entities



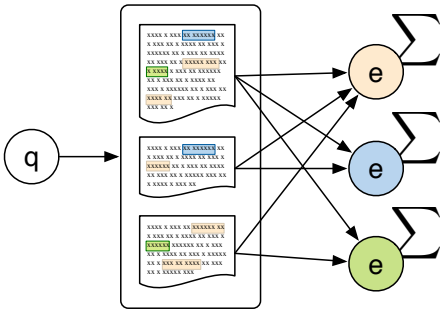
Two principal approaches

- **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



Document-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]

$$P(q|\theta_e) = \prod_{t \in q} \underbrace{P(t|\theta_e)}_{\text{Smoothing}}{}^{n(t,q)}$$

Smoothing
With collection-wide background model

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

$$\sum_d \underbrace{P(t|d,e)}_{\text{Term-candidate co-occurrence}} \underbrace{P(d|e)}_{\text{Document-entity association}}$$

Term-candidate co-occurrence
In a particular document.
In the simplest case: $P(t|d)$

Document-entity association

Document models ("Model 2")

[Balog et al. 2006]

$$P(q|e) = \sum_d \underbrace{P(q|d,e)}_{\text{Document relevance}} \underbrace{P(d|e)}_{\text{Document-entity association}}$$

Document relevance
How well document d supports the claim that e is relevant to q

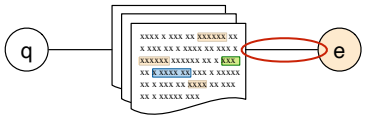
Document-entity association

$$\prod_{t \in q} \underbrace{P(t|d,e)}_{\text{Simplifying assumption}}{}^{n(t,q)}$$

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]

$$P(t|d,e) = \underbrace{\frac{1}{Z}}_{\text{Normalizing contant}} \sum_{i=1}^{N} \underbrace{\delta_d(i,t)}_{\text{Indicator function}} \underbrace{k(t,e)}_{\text{Proximity-based kernel}}$$

Normalizing contant

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

Proximity-based kernel
- constant function
- triangle kernel
- Gaussian kernel
- step function

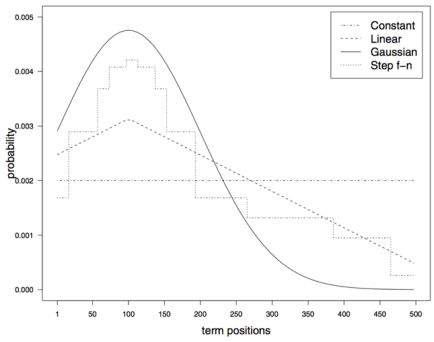


Figure taken from D. Petkova and W.B. Croft. Proximity-based document representation for named entity retrieval. CIKM'07.

Many possibilities in terms of modeling

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

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

Arithmetic Mean Discriminative (AMD) model

[Yang et al. 2010]

$$P_\theta(r=1|e,q) = \sum_d \underbrace{P(r_1=1|q,d)}_{\text{Query-document relevance}} \underbrace{P(r_2=1|e,d)}_{\text{Document-entity relevance}} \underbrace{P(d)}_{\text{Document prior}}$$

$$\underbrace{\sigma}_{\text{standard logistic function}}\Big(\sum_{i=1}^{N_f} \underbrace{\alpha_i}_{\text{weight parameters (learned)}} \underbrace{f_i(q,d_t)}_{\text{features}}\Big) \quad \sigma\Big(\sum_{j=1}^{N_g} \beta_j g_j(e,d_t)\Big)$$

logistic function over a linear combination of features

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]**

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

Voting models

Many different variants, including...

- Votes
  - Number of documents mentioning the entity

    $Score(e,q) = |M(e) \cap R(q)|$
- Reciprocal Rank
  - Sum of inverse ranks of documents

    $$Score(e,q) = \sum_{\{M(e)\cap R(q)\}} \frac{1}{rank(d,q)}$$
- CombSUM
  - Sum of scores of documents

    $$Score(e,q) = |\{M(e)\cap R(q)\}| \sum_{\{M(e)\cap R(q)\}} s(d,q)$$

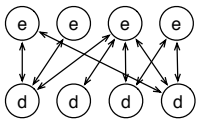
Graph-based models

[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

Infinite random walk

[Serdyukov et al. 2008]



$$\begin{aligned} P_i(d) &= \lambda P_J(d) + (1-\lambda)\sum_{e\to d} P(d|e)P_{i-1}(e), \\ P_i(e) &= \sum_{d\to e} P(e|d)P_{i-1}(d), \\ P_J(d) &= P(d|q), \end{aligned}$$

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

Incorporating entity types

Attributes (/Descriptions)

Type(s)

Relationships

people

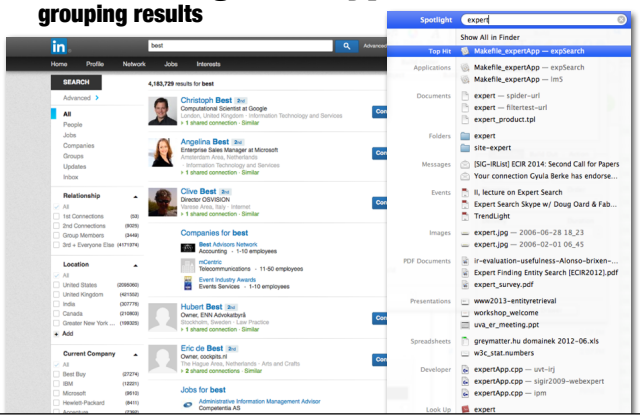
organizations

locations

products

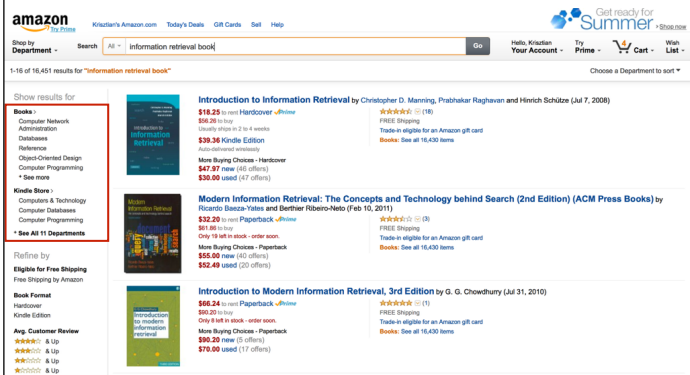
Interacting with types

grouping results



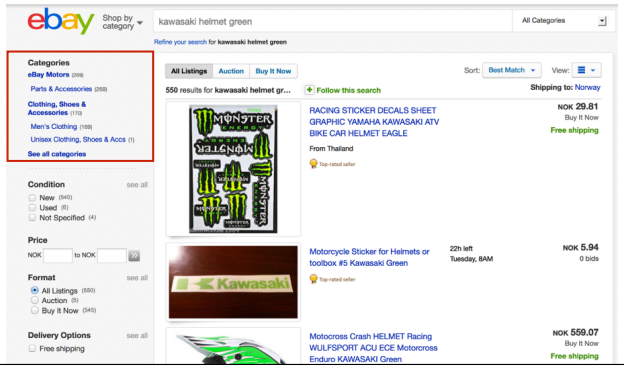
Interacting with types

filtering results



Interacting with types

filtering results



Type-aware ranking

- Typically, a two-component model:

$$P(q|e) = P(q_T, q_t|e) = \underbrace{P(q_T|e)}_{\text{Type-based similarity}}\underbrace{P(q_t|e)}_{\text{Term-based similarity}}$$

#1 **Where to get the *target type* from?**

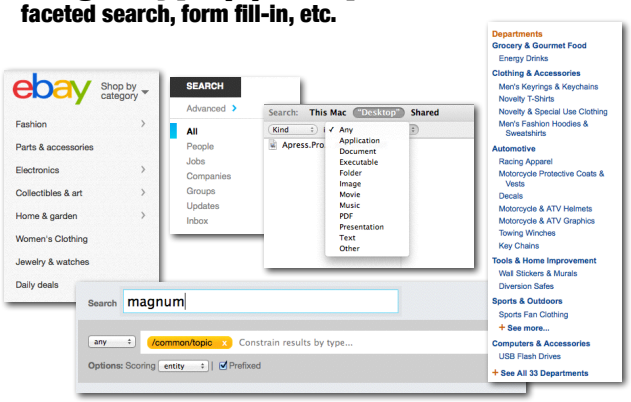
#2 **How to estimate *type-based similarity*?**

Target type

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

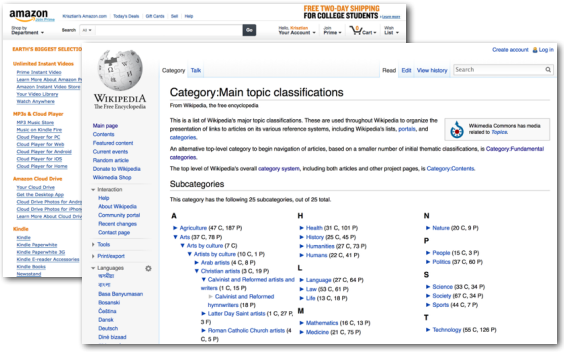
Target type(s) are provided

faceted search, form fill-in, etc.



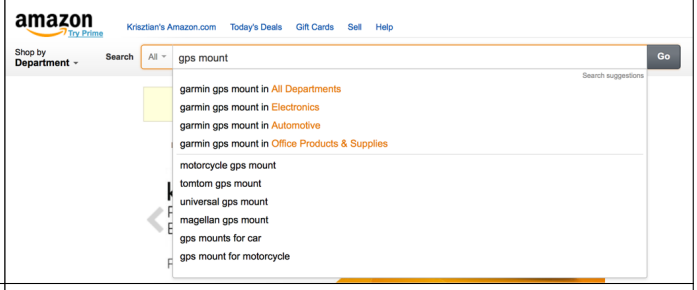
But what about very many types?

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

DBpedia

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

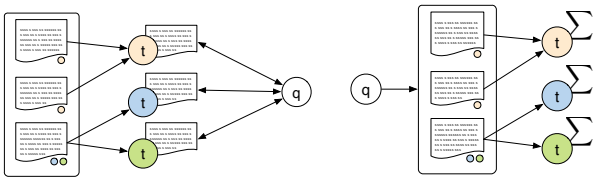
Two settings

- ~~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



Type-centric

Entity-centric

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

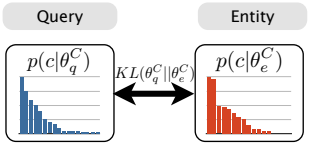
Type-based similarity

$$P(q|e) = P(q_T, q_t|e) = P(q_T|e)P(q_t|e)$$

- 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



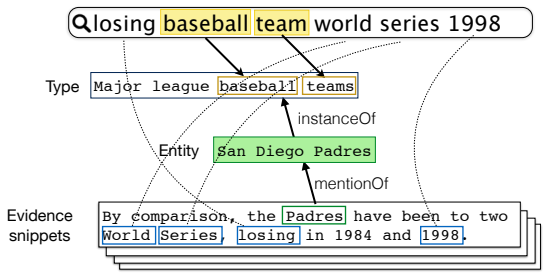
- Advantages
  - Sound modeling of uncertainty associated with category information
  - Category-based feedback is possible

Joint type detection and entity ranking [Sawant & Chakrabarti 2013]

- Assumes "telegraphic" queries with target type
  - woodrow wilson president <u>university</u>
  - dolly clone <u>institute</u>
  - lead <u>singer</u> 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" ($h(\vec{q}, \vec{z})$) or a "word matcher" ($s(\vec{q}, \vec{z})$)
  - Number of possible partitions is manageable ($2^{|q|}$)



Generative approach

Generate query from entity

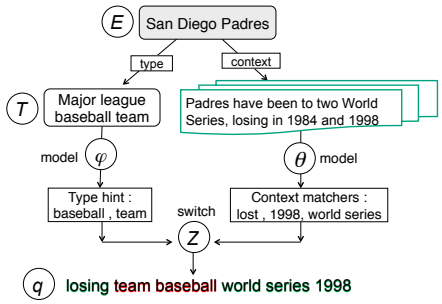


Figure taken from Sawant & Chakrabarti (2013). Learning Joint Query Interpretation and Response Ranking. In WWW '13. (see presentation)

Generative formulation

$$P(e|q) \propto \underbrace{P(e)}_{\text{Entity prior}} \sum_{t,\vec{z}} \underbrace{P(t|e)}_{\text{Type prior}} \underbrace{P(\vec{z})}_{\text{Query switch}} \underbrace{P(h(\vec{q}, \vec{z})|t)}_{\text{Type model}} \underbrace{P(s(\vec{q}, \vec{z})|e)}_{\text{Entity model}}$$

Type prior Estimated from answer types in the past

Query switch Probability of the interpretation

Type model Probability of observing t in the type model

Entity model Probability of observing t in the entity model

Discriminative approach

Separate correct and incorrect entities

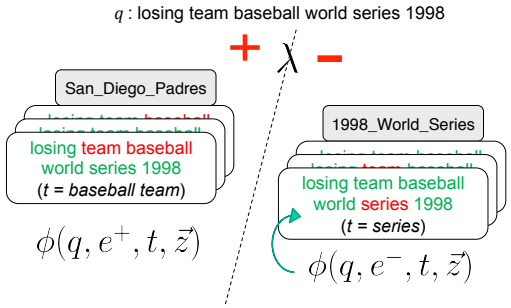


Figure taken from Sawant & Chakrabarti (2013). Learning Joint Query Interpretation and Response Ranking. In WWW '13. (see presentation)

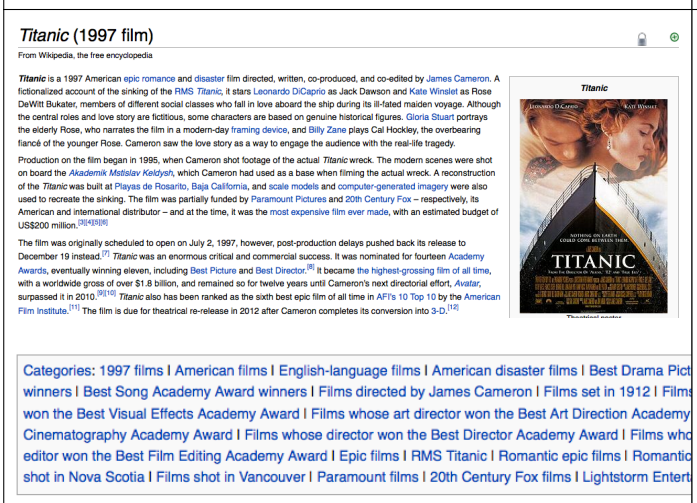
Discriminative formulation

$$\phi(q, e, t, \vec{z}) = \langle \underbrace{\phi_1(q, e)}_{\text{Models the type prior P(t|e)}}, \underbrace{\phi_2(t, e)}_{\text{Models the entity prior P(e)}}, \underbrace{\phi_3(q, \vec{z}, t)}_{\text{Comparability between hint words and type}}, \underbrace{\phi_4(q, \vec{z}, e)}_{\text{Comparability between matchers and snippets that mention e}} \rangle$$

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



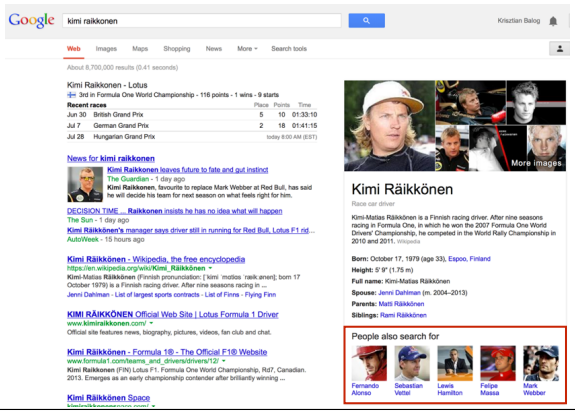
Entity relationships

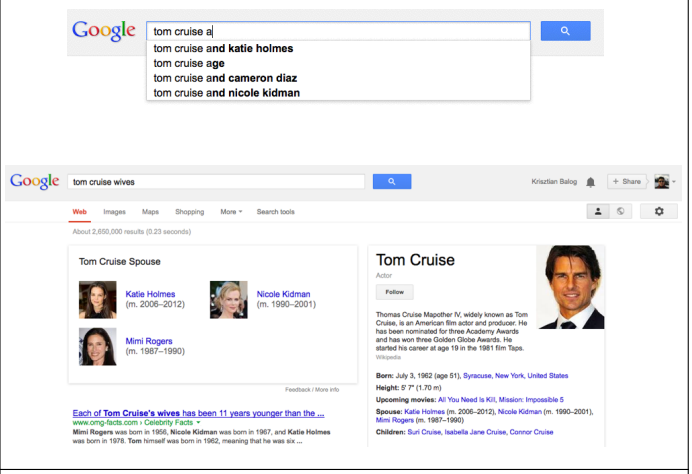
Attributes (/Descriptions)

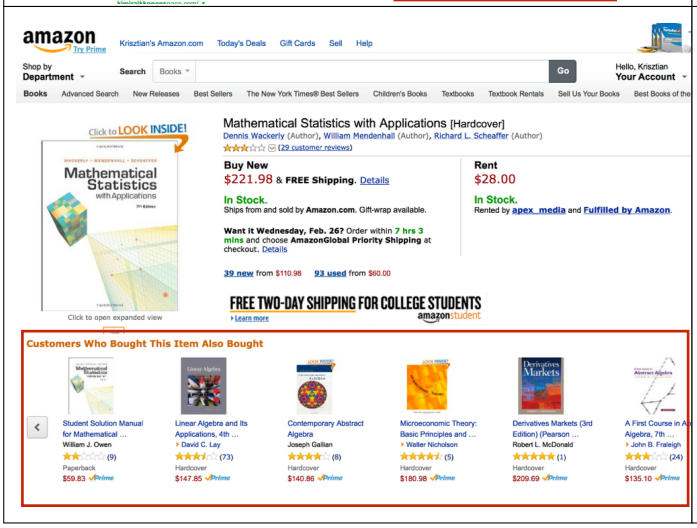
Type(s)

Relationships

Related entities







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

$$p(e|E,T,R) \propto \underbrace{p(e|E)}_{\text{Co-occurrence model}} \cdot \underbrace{p(T|e)}_{\text{Type filtering}} \cdot \underbrace{p(R|E,e)}_{\text{Context model}}$$

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