Semantic search

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

September 26, 2022

In this module

1. Understanding information needs

2. Leveraging entities in document retrieval

Understanding information needs

Understanding information needs

- Goal: infer a semantically enriched representation of the information need
- Various techniques
  - Classifying the query according to higher-level goals or intent
  - Segmenting the query and interpreting its structure
  - Identify the types or categories of entities targeted $\Leftarrow$ this lecture
  - Recognizing and disambiguating the mentioned entities $\Leftarrow$ this lecture

Understanding information needs

- Identifying target entity types
- Entity linking in queries

Identifying target entity types

One way to understand the meaning behind a search query is to identify the entity types targeted by the query.

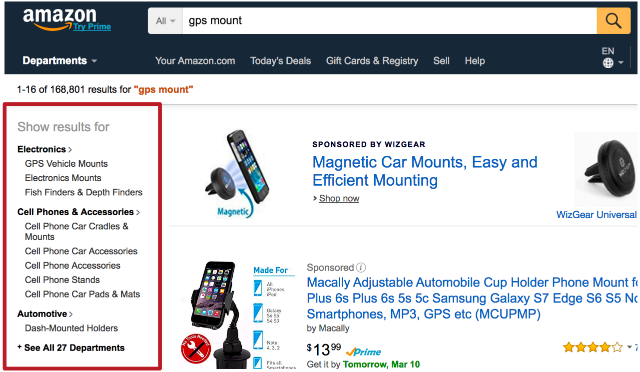


Figure: Product categories displayed on Amazon in response to the query "gps mount."

Problem definition

Definition

Target entity type identification is the task of finding the target types of a given input query, from a type taxonomy, such that these types correspond to most specific types of entities that are relevant to the query. Target types cannot lie on the same branch in the taxonomy. If no matching entity type can be found in the taxonomy, then the query is assigned a special NIL-type.

Approach

- Cast as a ranking problem: Given a keyword query q, estimate the relevance of an entity type $y \in \mathcal{T}$, $score(y; q)$
  - $\mathcal{T}$ is the set of possible types (i.e., *type taxonomy*)
  - Note: NIL-type detection is not considered
- Unsupervised and supervised approaches

Example type taxonomies

DBpedia Ontology

- Well-designed hierarchy
- 700+ types organized in 7 levels

- Agent (edit)
  - Deity (edit)
  - Employer (edit)
  - Family (edit)
    - NobleFamily (edit)
  - FictionalCharacter (edit)
    - ComicsCharacter (edit)
      - AnimangaCharacter (edit)
      - انیمنگا_کردار (edit)
    - DisneyCharacter (edit)
    - MythologicalFigure (edit)
    - NarutoCharacter (edit)
    - SoapCharacter (edit)
    - مُضحِکہ_خیزکردار (edit)
  - Organisation (edit)
    - Broadcaster (edit)
      - BroadcastNetwork (edit)
      - RadioStation (edit)

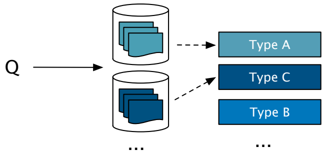
Wikipedia categories

- Not a well-defined "is-a" hierarchy, but a graph
  - A category may have multiple parent categories and there might be cycles along the path to ancestors
- Contains over 1.16M categories

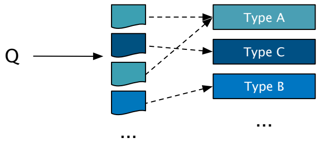
- ▼ Formula One (18 C, 57 P)
  - ▶ British Formula One Championship (1 C, 6 P)
  - ▶ Formula One cars (78 C, 37 P)
  - ▶ Formula One circuits (2 C, 80 P)
  - ▼ Formula One constructors (32 C, 132 P)
    - ▶ Alex von Falkenhausen Motorenbau (1 C, 2 P)
    - ▶ Alfa Romeo in Formula One (2 C, 7 P)
    - ▶ All American Racers (2 C, 1 P)
    - ▶ Alpine F1 Team (2 C, 2 P)
    - ▶ Arrows Grand Prix International (2 C, 10 P)
    - ▶ Aston Martin in Formula One (2 C, 6 P)
    - ▶ Benetton Formula (2 C, 28 P)
    - ▼ BMW in Formula One (2 C, 1 P)
      - ▶ BMW (1950s–60s) Formula One drivers (3 P)
      - ▶ BMW Sauber Formula One cars (4 P)

Unsupervised approaches

- **Type-centric model**
  - A term-based representation is built for each type
  - Types can then be ranked using existing document retrieval models
  - Analogous to constructing term-based entity representations from unstructured documents (cf. M6)



- **Entity-centric model**
  - Rank entities with respect to the query
  - Determine each type's relevance by considering the retrieval scores of entities of that type



Type-centric model

- A term-based representation ("type description" document) is constructed for each type
- Term pseudo-counts for a type are computed using:

$$\tilde{c}(t;y) = \sum_{e \in \mathcal{E}} c(t;e)\, w(e,y)$$

  - $c(t;e)$ is the count of term t in the description of entity e
  - $w(e,y)$ denotes the entity-type association weight; can be set uniformly across all entities that are typed with y:

$$w(e,y) = \begin{cases} \frac{1}{|\{e' : y \in \mathcal{T}_{e'}\}|}, & y \in \mathcal{T}_e \\ 0, & y \notin \mathcal{T}_e \end{cases}$$

- Given the term-based type representation, as defined by $\tilde{c}(t;y)$, types can be ranked using any standard retrieval method

Entity-centric model

- First, entities are ranked based on their relevance to the query
  - $score(q; e)$ may be computed using any entity retrieval model discussed earlier
  - $\mathcal{E}_q(k)$ denotes the set of top-k ranked entities for query q
- Then, the score for a given type y is computed by aggregating the relevance scores of the top-k entities with that type:

$$score_{EC}(y; q) = \sum_{e \in \mathcal{E}_q(k)} score(e; q)\, w(e, y)$$

  - $w(e, y)$ denotes the entity-type association weight (same as in type-centric model)

Supervised approach

- The type-centric model tends to return more specific types, while the entity-centric models favors more general types
- $\Rightarrow$ Can the two be combined?
- Also, additional ranking signals may be incorporated in a supervised feature-based approach
- Types of features
  - Baseline (entity- and type-centric scores)
  - Type taxonomy (depth, #children, #siblings, etc.)
  - Type label (various ways of measuring the similarity between type label and query)

Evaluation

- Evaluated using standard rank-based measures
- Ground truth type annotations: $\hat{\mathcal{T}}_q$
- *Strict* evaluation: binary decision, i.e., each returned type y either matches one of the ground truth types (1) or not (0)
- However, not all types of mistakes are equally bad
  - For example, the target entity type is *racing driver*
  - Returning a more specific type (*rally driver*) or a more general type (*athlete*) is less of a problem than returning a type from a completely different branch of the taxonomy, like *organization* or *location*
- $\Rightarrow$ Need for a more *lenient* evaluation that rewards near-misses

Lenient evalution

- First, the distance between types in the taxonomy $d(y, \hat{y})$ is set to
  - the number of steps between the two types, if they lie on the same branch (i.e., one of the types is a subtype of the other)
  - ∞ otherwise
- Then, the relevance level or gain of a type can be defined
  - in a *linear* fashion:

$$r(y) = \max_{\hat{y} \in \hat{\mathcal{T}}_q} \left(1 - \frac{d(y, \hat{y})}{h}\right)$$

  - or using an *exponential* decay function:

$$r(y) = \max_{\hat{y} \in \hat{\mathcal{T}}_q} \left(b^{-d(y, \hat{y})}\right)$$

  - where h is the depth of the type taxonomy and b is the base of the exponential function
  - max is used in order to consider the closest match from the set of ground types $\hat{\mathcal{T}}_q$
- The final measure is normalized discounted cumulative gain (NDCG), using the above (linear or exponential) relevance gain values

Exercise

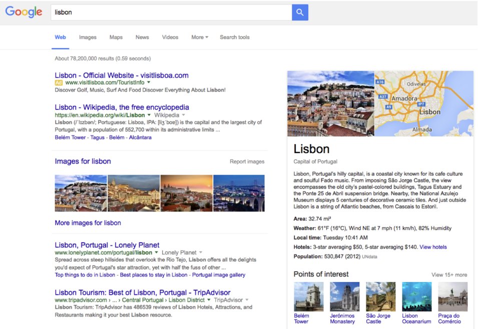
E8-1 Target Entity Type Identification Evaluation

Understanding information needs

- Identifying target entity types
- Entity linking in queries

Entities in queries

- A large fraction of queries in Web search are reported to contain named entities (40-70%, depending on the study)
- Recognize entities in queries would lead to an improved search experience, e.g.,
    - Direct answers/knowledge panels
    - Better document ranking (next lecture)

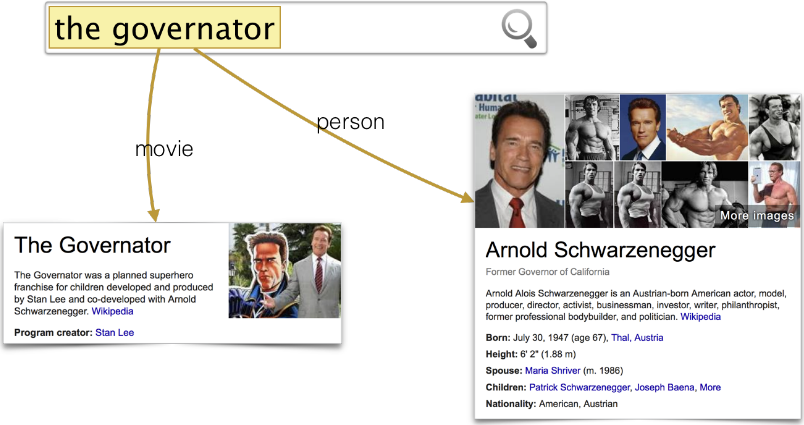


Challenges

- Search queries are short
- Limited context
- Lack of proper grammar, spelling
- Multiple interpretations
- Needs to be fast

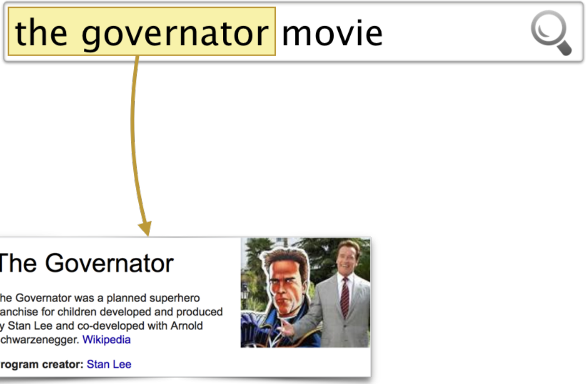
Example

Inherent ambiguity due to lack of context



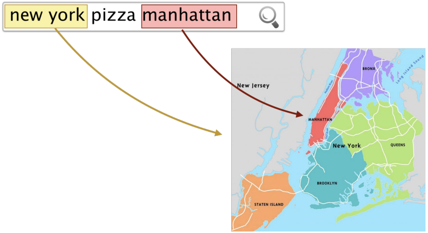
Example

Ambiguity resolved



Example

Multiple possible interpretations



Entity annotation tasks

- We distinguish between three entity annotation tasks in queries
    - *Named entity recognition* is the task of identifying mentions of named entities and tagging the mentions with their respective types
    - *Semantic linking* seeks to find a ranked list of entities that are semantically related to the query string
        - Unlike in entity retrieval, the goal is not to answer the user's underlying information need with a ranked list of entities, but to identify entities that are referenced (either explicitly or implicitly) in the query
    - *Interpretation finding aims* to discover all plausible meanings of the query; each interpretation consists of a set of non-overlapping and semantically compatible entity mentions, linked to a knowledge repository

Comparison of tasks

	Named Entity Recognition	**Semantic Linking**	**Interpretation Finding**
Result format	set/ranked list	ranked list	sets of sets
Explicit entity mentions?	yes	no	yes
Mentions can overlap	no	yes	no[2]
Evaluation criteria	recognized entities[1]	relevant entities	interpretations
Evaluation measures	set/rank-based	rank-based	set-based
Examples			
"*obama mother*"	"*obama*"/PER	Barack Obama Ann Dunham	{{Barack Obama}}
"*new york pizza manhattan*"	"*new york*"/LOC "*manhattan*"/LOC	New York City New York-style pizza Manhattan Manhattan Pizza ...	{{New York City, Manhattan}, {New York-style pizza, Manhattan}}

[1] Along with their respective types.
[2] Not within the same interpretation.

Interpretation finding

- Addresses the ambiguity of queries head-on: a query can legitimately have more than one interpretation
  - An interpretation is a set of non-overlapping linked entity mentions that are semantically compatible with the query text
- Pipeline architecture

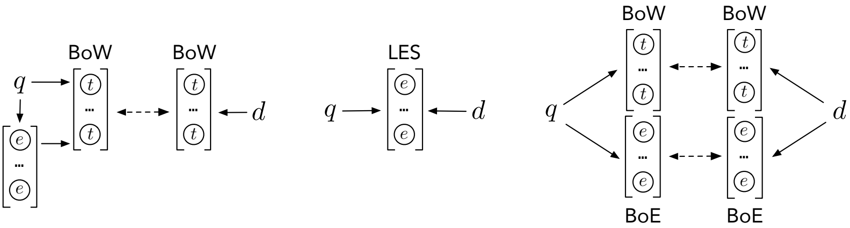
  - *Mention detection* is performed similarly as it is done for documents (i.e., dictionary-based)
  - *Candidate entity ranking* corresponds to the task of semantic linking
  - *Interpretation finding* is the counterpart of the disambiguation component in conventional entity linking

Leveraging entities in document retrieval

Leveraging entities in document retrieval

- Traditionally, document retrieval methods compare query and document representations, which are based on terms (words)
- Entities facilitate a semantic understanding of both queries and documents
- Different approaches for leveraging entities
  - Expansion-based
  - Projection-based
  - Entity-based
- Prerequisite: Mapping queries to entities

Three main ways to leverage entities



(a) Expansion-based (b) Projection-based (c) Entity-based

- *Expansion-based*: utilize entities as a source of expansion terms to enrich the query
- *Projection-based*: treat entities as a latent layer and map both queries and documents to a latent entity space (LES)
- *Entity-based*: consider entity-based representations (bag-of-entities) in "duet" with traditional bag-of-words representations in the retrieval model

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Mapping queries to entities

- Goal: Identify a set of entities that may be semantically related to the query
- Shared by all approaches for leveraging entities
- Use the probability $P(e|q)$ to express the likelihood of entity e being related to query q
- The estimation may be based on:
  - entities mentioned in the query
  - entities retrieved directly from a knowledge base
  - entities retrieved indirectly, through pseudo-relevant documents

Entities mentioned in the query

- Entities can be identified and disambiguated using entity linking techniques
- Single entity with the highest annotation score

$$P(e|q) = \begin{cases} 1, & e = argmax_{e \in \mathcal{E}_q} score_{ELQ}(e; q) \\ 0, & otherwise \end{cases}$$

  - $\mathcal{E}_q$ is set the of entities that have been identified in query q
  - $score_{ELQ}(e; q)$ is the associated confidence score
- Multiple entities

$$P(e|q) = \begin{cases} \frac{1}{Z} score_{ELQ}(e; q), & score_{ELQ}(e; q) > \gamma \\ 0, & otherwise \end{cases}$$

  - γ is a threshold parameter
  - Z is a normalization coefficient such that $0 \leq P(e|q) \leq 1$

Entities retrieved directly from a KB

- Query a knowledge base directly for relevant entities

$$P(e|q) = \begin{cases} \frac{1}{Z} score_{ER}(e;q), & e \in \mathcal{E}_q(k) \\ 0, & otherwise \end{cases}$$

  - $score_{ER}(e;q)$ is the relevance score of entity e given q
  - Z is a normalization coefficient
  - Note that for pragmatical reasons only the top-k entities are considered

Entities retrieved indirectly

-
- Uses the top-ranked documents retrieved in response to the query
    - This is in the spirit of pseudo relevance feedback
    - Corresponds to the setting of ranking entities without direct representations

$$P(e|q) \propto \sum_{d \in \mathcal{D}_q(k)} P(e|d)P(d|q)$$

- $\mathcal{D}q(k)$ denotes the set of top-k highest scoring documents retrieved in response to query q
- $P(d|q)$ corresponds to document d's relevance to the query
- $P(e|d)$ is the probability of observing the entity in d

Entities retrieved indirectly

- $P(e|d)$ may be taken as a maximum-likelihood estimate

$$P(e|d) = \frac{c(e;d)}{\sum_{e' \in d} c(e';d)}$$

- $c(e;d)$ is the number of times entity e occurs in document d
- The frequency of e across the document collection may also be taken into account by adding an IDF-like component

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Projection-based methods

- Addresses the inherent limitation of standard IR retrieval models to retrieve (relevant) documents that have no explicit term matches with the query
- The overall idea:
  - Construct a high-dimensional latent entity space where each dimension corresponds to one entity and then map both queries and documents to the latent space accordingly
- The relevance between a query and a document is estimated based on their projections to this latent entity space
- Allows to uncover hidden (latent) semantic relationships between queries and documents
- Approaches:
  - Explicit Semantic Analysis
  - Latent Entity Space model
  - EsdRank

Explicit Semantic Analysis

- The semantics of a given word are described by a vector storing the word's association strengths to a KR concept
- Primarily focused on using Wikipedia as the underlying knowledge repository
- Two components:
  - Concept-based indexing
  - Concept-based retrieval

Concept-based indexing

- The semantic representation of a given term t is a concept vector of length $|\mathcal{E}|$

$$t = \langle w(e_1, t), \ldots, w(e_{|\mathcal{E}|}, t) \rangle$$

- Each element of the vector corresponds to an entity in the knowledge repository
- $w(e, t)$ quantifies the strength of the association between term t and the given entity. It is the normalized TF-IDF weight of t in the description of e

$$w(e, t) = \frac{TFIDF(t, e)}{\sqrt{\sum_{t' \in \mathcal{V}} TFIDF(t', e)^2}}$$

Concept-based indexing

- The semantic representation is computed by taking the centroid of the individual terms' concept vectors

$$w(ej, z) = \frac{1}{l_z} \sum_{t \in \mathcal{Z}} c(t; z) w(e_j, t)$$

- l_z is the length of z
- $c(t; z)$ is the number of times term t appears in z
- Concept-based vectors are sparse, with most weights being zero
- they can be efficiently represented using an inverted index.

Concept-based retrieval

- semantic similarity between query q and document d may be computed by
  - mapping both query q and document d to the ESA concept space
  - taking cosine similarity of their concept vectors

$$score(q; d) = cos(\mathbf{q}, \mathbf{d})$$

- issue with long documents where only a small part might be relevant to the current query
  - The solution is to split the document into passages where each passage $s \in d$ is represented by its own concept vector $\mathbf{s}$ is matched against the query
  - The final retrieval score combines the full document's similarity score with that of the best performing passage

$$score_{ESA}(q; d) = cos(\mathbf{q}, \mathbf{d}) + \max_{s \in d} cos(\mathbf{q}, \mathbf{s})$$

Latent Entity Space model

- Based on a generative probabilistic framework.
- The document's retrieval score is a linear combination of the latent entity space score and the original query likelihood score

$$score_{LES}(q; d) = \alpha \underbrace{\sum_{e \in \mathcal{E}} P(q|e)P(e|d)}_{\text{LES score}} + (1 - \alpha)P(q|d)$$

- α is the interpolation parameter
- $P(q|d)$ is the query likelihood score
- $P(q|e)$ and $P(e|d)$ are query projection and document projection respectively

Query projection

- May be interpreted as the likelihood of the query q being generated by entity e
- One approach is to base it on the language model of the entity, θ_e

$$P(q|e) = \prod_{t \in q} P(t|\theta_e)^{c(t;q)}$$

- Another approach is to leverage the set of query entities $\mathcal{E}_q$ in a pairwise manner

$$P(q|e) \propto \sum_{e' \in \mathcal{E}_q} sim(e, e') P(e'|q)$$

- $sim(e, e')$ may be any symmetric pairwise similarity measure
- $P(e'|q)$ is the query association probability for e'

Document projection

- The probability $P(e|d)$ may be interpreted as the projection of document d to the latent entity space
- May be estimated using existing document retrieval models, e.g., by computing entity likelihood (the probability of e generated from the language model of d)
- Example using cross-entropy between the document and entity language models:

$$P(e|d) = exp(-CE(\theta_e||\theta_d)) = exp\left(\sum_{t \in \mathcal{V}} P(t|\theta_e) log(P(t|\theta_d))\right)$$

EsdRank

- Learning-to-Rank Model (Latent-ListMLE)
- Uses a combination of
  - Query-entity features (query projection)
  - Entity-document features (document projection)
  - Other features (e.g., entity popularity, document quality)

Features in EsdRank

Group	Description
Query-entity features	
$P(e\|q)$	Query-entity association probability (cf. Sect. 8.1)
$score_{ER}(e;q)$	Entity retrieval score (relevance of e given q)
$sim(\mathcal{T}_e, \mathcal{T}_q)$	Type-based similarity between the entity ($\mathcal{T}_e$) and the query ($\mathcal{T}_q$)
$maxSim(e, \mathcal{E}_q)$	Max. pairwise similarity between e and other query entities $e' \in \mathcal{E}_q$
$avgSim(e, \mathcal{E}_q)$	Avg. pairwise similarity between e and other query entities $e' \in \mathcal{E}_q$
Entity-document features	
$sim(e,d)$	Textual similarity between e and d
$sim(\mathcal{T}_e, \mathcal{T}_d)$	Type-based similarity between the entity ($\mathcal{T}_e$) and the document ($\mathcal{T}_d$)
$numMentioned(\mathcal{E}_e^i, d)$	Number of related entities (at most i hops from e) mentioned in d
Other features	
$IDF(e)$	IDF score of the entity (based on the number of documents that are annotated with e)
$quality(d)$	Quality score of d (document length, SPAM score, PageRank, etc.)

Note that many of these features are instantiated multiple times, using different similarity methods or parameter configurations

EsdRank

- ListMLE
  - it is a listwise learning-to-rank algorithm that uses a parametric model to estimate the probability of a document ranking being generated by a query
  - It employs maximum likelihood estimation (MLE) to find the parameters that maximize the likelihood of the best ranking
- Latent-ListMLE extends ListMLE by adding a latent layer of candidate entities in the generation process
- The probability of a document ranking $d = \langle d_1, \ldots, d_k \rangle$ given q is

$$P(d|q; w, \theta) = \prod_{i=1}^{k} \sum_{e \in \mathcal{E}_{\mathrm{II}}} P(d_i|e, \mathcal{D}_q(i,k)) P(e|q)$$

- $\mathcal{D}_q(i,k) = \{d_i, \ldots, d_k\}$ is the set of documents that were not ranked in positions $1, \ldots, i-1$
- The parameters of the model, ω and θ, are learned using MLE and the expectation-maximization (ME) algorithm

Leveraging entities in document retrieval

- Mapping queries to entities
- Projection-based methods
- Entity-based representations

Entity-based representations

- We make use of explicit entity annotations of documents
  - In contrast to projecting documents to a latent entity layer
- Several approaches
  - Entity-Based Document Language Models - entities are blended with terms in a single representation layer
  - Bag-of-Entities Representation - a separate bag-of-entities representation is introduced and combined with the traditional bag-of-terms representation

Entity-Based Document Language Models

- Consider individual terms as well as term sequences that have been annotated as entities
- Extend the vocabulary of terms ($\mathcal{V}$) with entities ($\mathcal{E}$)
- The maximum likelihood estimate of vocabulary token x (may be a term or an entity, $x \in \mathcal{V} \cup \mathcal{E}$) given d is defined as

$$P(x|d) = \frac{c(x;d)}{l_d}$$

- $c(x;d)$ is the (pseudo) count of x in document d
- The representation length of the document is then given by $l_d = \sum_{x \in d} c(x;d)$

Entity-Based Document Language Models (cont'd)

- This maximum likelihood estimate is then smoothed with a background (collection-level) language model

$$P(x|\theta_d) = \frac{c(x;d) + \mu P(x|D)}{l_d + \mu}$$

- μ is a smoothing parameter
- The collection language model is also a maximum likelihood estimate, computed over the set $\mathcal{D}$ of documents

$$P(x|D) = \frac{\sum_{d \in \mathcal{D}} c(x;d)}{\sum_{d \in \mathcal{D}} l_d}$$

Entity-Based Document Language Models (cont'd)

- How to compute the token pseudo-counts?
- **Hard confidence-level thresholding**
  - Only consider those entity annotations that are above a given (pre-defined) threshold $\tau \in [0, 1]$

$$\tilde{c}(x; d) = \begin{cases} \lambda c(x; d), & x \in \mathcal{V} \\ (1 - \lambda) \sum_{i=1}^{l_d} \mathbb{1}(x_i = x, score_{EL}(x_i; d) \geq \tau), & x \in \mathcal{E} \end{cases}$$

  - x_i refers to the token at position i in the document
  - $score_E L(x_i; d)$ is the entity linking confidence associated with that token
  - The λ parameter controls the relative importance given to term vs. entity tokens.
- **Soft confidence-level thresholding**
  - Recognizes all entities that are linked in the document and weighs them by their corresponding confidence levels

$$\tilde{c}(x; d) = \begin{cases} \lambda c(x; d), & x \in \mathcal{V} \\ (1 - \lambda) \sum_{i=1}^{l_d} \mathbb{1}(x_i = x) score_{EL}(x_i; d), & x \in \mathcal{E} \end{cases}$$

Entity-Based Document Language Models (cont'd)

- The ranking of documents is based on the negative cross-entropy (CE) between the query and document language models

$$score_{ELM}(d;q) = -CE(\theta_q || \theta_d) = \sum_{x \in \mathcal{V} \cup \mathcal{E}} P(x|\theta_q) log P(x|\theta_d)$$

- θ_q and θ_d are query and document language models respectively

Bag-of-Entities Representation

- So far, entity-based language models used a single representation layer, in which terms and entities are mixed together
- Now, the term-based and entity-based representations are kept apart and are used in "duet"
  - Queries and documents are represented in the term space as well as in the entity space (*bag-of-entities*)
- Several ranking models
  - Basic Ranking Models
  - Explicit Semantic Ranking (ESR)
  - Word-Entity Duet Framework

Basic Ranking Models

- Two basic ranking models based on bag-of-entities representations
- **Coordinate Match** ranks documents based on the number of query entities they mention

$$score_{CM}(d;q) = \sum_{e \in \mathcal{E}_q} \mathbb{1}(c(e;d) > 0)$$

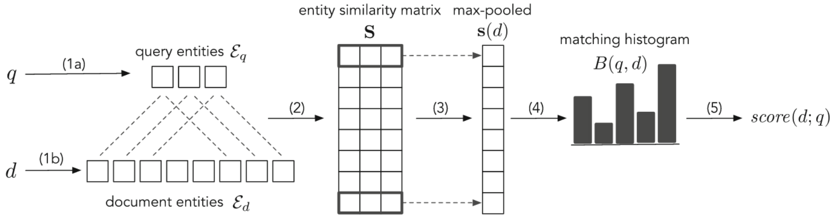
- **Entity Frequency** also considers the frequency of query entities in documents

$$score_{EF}(d;q) = \sum_{e \in \mathcal{E}_q} c(e;q) \log c(e;d)$$

- These ranking functions are used to re-rank the top-k documents retrieved by a standard term-based retrieval model

Explicit Semantic Ranking (ESR)

- Incorporates relationship information from a knowledge graph to enable "soft matching" in the entity space



- First, creates a query-document entity similarity matrix S.
  - Each element $S(e, e')$ in this matrix represents the similarity between a query entity $ein\mathcal{E}_q$ and a document entity $e' \in \mathcal{E}_d$

$$S(e, e') = cos(\mathbf{e}, \mathbf{e'})$$

  - $\mathbf{e}$ is the embedding vector of entity e

Explicit Semantic Ranking (cont'd)

- ESR performs two pooling steps
- First max-pooling along the query dimension

$$s(d) = \max_{e \in \mathcal{E}_q} S(e, \mathcal{E}_d)$$

- The second is bin-pooling to group and count the number of document entities according to the strength of their matches to the query

$$B_i(q, d) = log \sum_j \mathbb{1}(st_i \leq \mathbf{s}_j(d) < ed_i)$$

- $[st_i, ed_i)$ is the score range for the ith bin
- $B_i(q, d)$ is the number of entities that fall into that bin

Word-Entity Duet Framework

- Incorporates cross-space interactions between term-based and entity-based representations
  - Query terms to document terms ($match(\mathbf{q}_t, \mathbf{d}_t)$)
  - Query entities to document terms ($match(\mathbf{q}_e, \mathbf{d}_t)$)
  - Query terms to document entities ($match(\mathbf{q}_t, \mathbf{d}_e)$)
  - Query entities to document entities ($match(\mathbf{q}_e, \mathbf{d}_e)$)

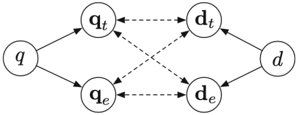


Figure: $\mathbf{q}_t$ and $\mathbf{d}_t$ denote the bag-of-words, $\mathbf{q}_e$ and $\mathbf{d}_e$ denote the bag-of-entities representations

Summary

- Understanding information needs
  - Identifying target entity types
  - Entity linking in queries
- Leveraging entities in document retrieval
  - Mapping queries to entities
  - Projection-based methods
  - Entity-based representations

Reading

- Entity-Oriented Search (Balog)
  - Chapter 7, Chapter 8
    - Section 7.2, until 7.2.4.1 (inclusive)
    - Section 7.3, until 7.3.2 (inclusive)
    - Introduction and Section 8.1
    - Sections 8.3–8.4.2.3 (inclusive)