Chapter 7
Understanding Information Needs

Understanding what the user is looking for is at the heart of delivering a quality search experience. After all, it is rather difficult to serve good results, unless we can comprehend the intent and meaning behind the user's query. *Query understanding* is the first step that takes place before the scoring of results. Its overall aim is to infer a semantically enriched representation of the information need. This involves, among others, classifying the query according to higher-level goals or intent, segmenting it into parts that belong together, interpreting the query structure, recognizing and disambiguating the mentioned entities, and determining if specific services or verticals[1] should be invoked. Such semantic analysis of queries has been a long-standing research area in information retrieval. In Sect. 7.1, we give a brief overview of IR approaches to query understanding.

In the rest of the chapter, we direct our focus of attention to representing information needs with the help of structured knowledge repositories. The catchphrase "things, not strings" was coined by Google when introducing their Knowledge Graph.[2] It aptly describes the current chapter's focus: Capturing what the query is about by automatically annotating it with entries from a knowledge repository. These semantic annotations can then be utilized in downstream processing for result ranking (see Chap. 4) and/or result presentation. Specifically, in Sect. 7.2, we seek to identify the types or categories of entities that are targeted by the query. In Sect. 7.3, we perform entity linking in queries, which is about recognizing specific entity mentions and annotating them with unique identifiers from the underlying knowledge repository. Additionally, we consider the case of unresolvable ambiguity, when queries have multiple possible interpretations.

[1] A *vertical* is a specific segment of online content. Some of the most common verticals include shopping, travel, job search, the automotive industry, medical information, and scholarly literature.

[2] https://googleblog.blogspot.no/2012/05/introducing-knowledge-graph-things-not.html.

K. Balog, *Entity-Oriented Search*, The Information Retrieval Series 39,
https://doi.org/10.1007/978-3-319-93935-3_7

Finally, in Sect. 7.4 we automatically generate query templates that can be used to determine what vertical services to invoke (e.g., weather, travel, or jobs) as well as the parameterization (attributes) of those services.

7.1 Semantic Query Analysis

The purpose of this section is to provide an overview of the range of tasks and techniques that have been proposed for *semantic query analysis*. These methods all aim to capture the underlying intent and meaning behind the user's query. Each of these techniques addresses query understanding from a specific angle, has its own particular uses, and is often complementary to the other means of query analysis. Query understanding is a vast area, one which would probably deserve a book on its own. In this chapter, we will discuss in detail only a selection of query analysis techniques, chosen for their relevance to an entity-oriented approach. In contrast, this section is meant to help see those methods in a broader perspective.

In particular, we will look at three groups of approaches.

- *Query classification* is the task of automatically assigning a query to one or multiple pre-defined categories.
- *Query annotation* is about generating semantic markup for a query.
- *Query interpretation* aims at determining the meaning of a query as a whole, by finding out how the segmented and annotated parts of the query relate to each other.

7.1.1 Query Classification

Query classification is the problem of automatically assigning a query to one or multiple pre-defined categories, based on its intent or its topic.

7.1.1.1 Query Intent Classification

The first group of techniques aims to classify queries into categories based on the underlying user intent. Jansen and Booth [38] define user intent as "the expression of an affective, cognitive, or situational goal in an interaction with a Web search engine." When we talk about user intent, we are more concerned with how the goal is expressed and what type of resources the user desires to fulfill her information need than with the goal itself (which is something "external") [38].

In his seminal work, Broder [16] introduced a taxonomy of query intents for web search, according to the following three main categories of user goals:

- *Navigational*, where the intent is to reach a particular site ("take me to X").
- *Informational*, where the intent is to acquire information about a certain topic ("find out more about X").
- *Transactional*, where the intent is to perform some web-mediated activity (purchase, sell, download, etc.).

Broder's categorization is broadly accepted and is the most commonly used one. Rose and Levinson [63] classify queries into *informational*, *navigational*, and *resource* categories, with further finer-grained subcategories for informational and resource queries. Jansen et al. [39] employ a three-level classification, based on [16] and [63], and elaborate on how to operationalize each category. Lee et al. [43] classify queries as navigational or informational using past user-click behavior and anchor-link distribution as features.

There are many other ways to categorize user intent. For example, Dai et al. [21] detect whether the query has a *commercial intent* by training a binary classifier, using the content of search engine result pages (SERPs) and of the top ranked pages. Ashkan and Clarke [4] use query-based features along with the content of SERPs to classify queries along two independent dimensions: commercial/noncommercial and navigational/informational. Zhou et al. [83] predict the *vertical intent* of queries, such as image, video, recipe, news, answer, scholar, etc.; this is also related to the problem of vertical selection in aggregated search [3]. There have been studies on specific verticals, e.g., identifying queries with a *question intent*, which can be successfully answered by a community question answering vertical [75], or determining *news intent*, in order to integrate content from a news vertical into web search results [22]. Yin and Shah [81] represent generic search intents, for a given type of entity (e.g., musicians or cities), as *intent phrases* and organize them in a tree. An intent phrase is a word or phrase that frequently co-occurs with entities of the given type in queries. Phrases that represent the same intent are grouped together in the same node of the tree (e.g., "songs" and "album" for musicians); sub-concepts of that intent are represented in child nodes (e.g., "song lyrics," "discography," or "hits"). Hu et al. [35] classify search intent by mapping the query onto Wikipedia articles and categories, using random walks and explicit semantic analysis [25].

7.1.1.2 Query Topic Classification

User goals may also be captured in terms of topical categories, which may be regarded as a multiclass categorization problem. The *query topic classification* task, however, is considerably more difficult than other text classification problems, due to data sparsity, i.e., the brevity of queries.

One landmark effort that inspired research in this area was the 2005 KDD Cup competition. It presented the Internet user search query categorization challenge: Classify 800,000 web queries into 67 predefined categories, organized in a two-level hierarchy. For each query, participants may return a set of up to five categories (the order is not important). Evaluation is performed in terms of precision, recall, and F1-score. Participants were given only a very small set of 111 queries with labeled categories, necessitating creative solutions. We refer to Li et al. [48] for an overview. In a follow-up study to their winning solution, Shen et al. [68] present a method that uses as intermediate taxonomy (the Open Directory Project, ODP[3]), as a bridge connecting the target taxonomy with the queries to be classified. Another key element of their solution is that they submit both search queries and category labels to a web search engine to retrieve documents, in order to obtain a richer representation for the queries. The same idea of issuing the given query against a web search engine and then classifying the highest scoring documents is employed by Gabrilovich et al. [24] for classifying queries onto a commercial taxonomy (which is intended for web advertising). The main differences with [68] is that they (1) build the query classifier directly for the target taxonomy, which is two magnitudes larger (approx. 6000 nodes), and (2) focus on rare ("tail") queries. Instead of viewing it as a text categorization problem, Ullegaddi and Varma [76] approach query topic classification as an IR problem and attempt to learn category rankings for a query, using a set of term-based features. They also use ODP as an intermediate taxonomy and utilize documents that are assigned to each category. Again, the query is submitted against a web search engine, then the top-k highest weighted terms are extracted from the highest ranked documents to enrich its representation.

A related task is that of classifying questions in community-based question answering sites; we refer to Srba and Bielikova [70, Sect. 5.3] for an overview. In Sect. 7.2, we discuss the problem of identifying the target types of entity-oriented queries, with reference to a given type taxonomy.

7.1.2 Query Annotation

Query annotation is an umbrella term covering various techniques designed to automatically generate semantic markup for search queries, which can contribute to a better understanding of what the query is about. Query annotation includes a number of tasks, such as phrase segmentation, part-of-speech and semantic tagging, named entity recognition, abbreviation disambiguation [79], and stopword and "stop structure" detection [37]. Many of these annotation tasks have been studied by the databases and natural language processing communities as well. Most approaches focus on a particular annotation task in isolation. However, given that these annotations are often related, it is also possible to obtain them jointly by combining several independent annotations [9, 29]. Below, we lay our attention on two of the most important and widely studied query annotation tasks: *segmentation* (structural annotations) and *tagging* (linguistic and semantic annotations). We shall discuss entity annotations of queries in detail in Sect. 7.3.

[3] http://dmoz.org/.

7.1.2.1 Query Segmentation

Query segmentation is the task of automatically grouping the terms of the query into phrases. More precisely, a segmentation s for the query q consists of a sequence of disjunct segments $\langle s_1, \ldots, s_n \rangle$, such that each s_i is a contiguous subsequence of q and the concatenation of $s_1, \ldots, s_n$ equals q. For example, the query "*new york travel guides*" may be segmented as "*[new york] [travel guides]*."

One of the first approaches to web query segmentation is by Risvik et al. [62], who segment queries based on *connexity*, which is defined as a product of the segment's frequency in a query log and the mutual information within the segment. Pointwise mutual information, as a measure of word association, has been used in many of the later studies as well, computed either on the term level [40] or on the phrase level [36, 71], and is commonly used as a baseline. Bergsma and Wang [11] employ a supervised learning approach, where a classification decision, whether to segment or not, is made between each pair of tokens in the query. They use three groups of features: context features (preceding and following tokens in the query, if available), dependency features (POS tags), and statistical features (frequency counts on the Web). Another important contribution of this work is a manually annotated gold standard corpus (Bergsma-Wang-Corpus, BWC) comprising 500 queries sampled from the AOL query log dataset. BWC has been used as a standard test collection for query segmentation in subsequent work [15, 30, 47, 71]. Rather than using a supervised approach that requires training data, Tan and Peng [71] suggest an unsupervised method that uses n-gram frequencies from a large web corpus as well as from Wikipedia. Many other works adopt a similar rationale for segmentation, e.g., Huang et al. [36] use web-scale n-gram language models, while Mishra et al. [56] exploit n-gram frequencies from a large query log. Hagen et al. [30] present a simple frequency-based method, relying only on web n-gram frequencies and Wikipedia titles, that achieves comparable performance to state-of-the-art approaches while being less complex and more robust. Further, Hagen et al. [30] enrich the Bergsma-Wang-Corpus by means of crowdsourcing; they also introduce the Webis Query Segmentation Corpus, which is a larger sample from the AOL query log, consisting of 50k queries. Finally, there is a line of work on segmenting queries based on retrieval results, the idea being that it is hard to make a segmentation decision based on the query terms alone. Instead, one can bootstrap segmentation decisions based on the document corpus, using the top retrieved results [8] or snippets [15]. Note that these methods involve an extra retrieval round for obtaining the segmentation.

Evaluating against manual annotations "implicitly assumes that a segmentation technique that scores better against human annotations will also automatically lead to better IR performance" [65]. Instead, Saha Roy et al. [65] propose a framework for extrinsic evaluation of query segmentation. This involves assessing how well the given segmentation performs on the end-to-end retrieval task, while treating the retrieval algorithm as a black box. Their results confirm that segmentation indeed benefits retrieval performance. Further, they find human annotations to be a good proxy, yet "human notions of query segments may not be the best for maximizing retrieval performance, and treating them as the gold standard limits the scope for improvement for an algorithm" [65].

7.1.2.2 Query Tagging

It is possible to obtain semantically more meaningful segments than what is yielded by query segmentation techniques. *Query tagging* (or *semantic tagging*) refers to the process of annotating query terms with labels from a predefined label set. *Part-of-speech (POS) tagging* is one of the basic techniques in natural language processing to capture the meaning of text. The task is to label each word with a tag that describes its grammatical role, such as noun (NN), verb (VB), adjective (JJ), etc. POS tags may have different granularity; it is possible, e.g., to distinguish between singular (NN) and plural (NNS) nouns or between regular (RB), comparative (RBR), and superlative (RBS) adverbs. On regular text, POS tagging can be performed with high accuracy; e.g., the Stanford POS tagger [74], one of the most widely used systems, achieves 97% accuracy on the Penn Treebank-3 corpus [52]. Applied to short queries, that tend to lack proper grammar, punctuation, or capitalization, existing NLP techniques are much less successful [6]. Based on a manually annotated sample of queries from a commercial search engine, Barr et al. [6] show that the distribution of POS tag types in queries is rather different from that of standard (edited and published) text. Specifically, they find that among keyword queries the "most common tag is the proper noun, which constitutes 40% of all query terms, and proper nouns and nouns together constitute 71% of query terms" [6], while many standard POS tags (e.g., verbs and determiners) seldom appear in web search queries. In query annotation, therefore, a great emphasis is placed on detecting noun phrases and entities. Barr et al. [6] further show that tagger performance is severely affected by the lack of capitalization in queries.

Bendersky et al. [8] mark up queries using three types of annotations: capitalization, POS tags, and segmentation indicators. Rather than relying on the query itself, they draw on the (latent) information need behind the query and leverage the document corpus using pseudo relevance feedback techniques. In follow-up work, instead of solving the above annotation tasks in isolation, Bendersky et al. [9] perform them jointly and leverage the dependencies between the different types of markup.

Another line of work focuses on the identification of "key concepts" (i.e., the most important noun phrases) in verbose natural language queries. By assigning higher weights to these key concepts during document scoring, one can attain better retrieval effectiveness. Bendersky and Croft [7] use the noun phrases in the query as candidate concepts and use a supervised machine learning approach to classify each as being a key concept or not, and set a concept's importance based on the classifier's confidence score. Features include corpus-based frequency statistics, computed from the document collection, from an external collection (Google n-grams [14]), and also from a large query log.

Query tagging may also be performed with respect to an underlying domain-specific schema. It is often assumed that queries have already been classified onto a given domain (like movies, books, products, etc.). The task then is to assign each query term a label indicating which field it belongs to [46, 51]. A great deal of attention has been directed toward the product search domain, "since this is one representative domain where structured information can have a substantial influence on search experience" [46] (not to mention the obvious commercial value). Nevertheless, the proposed methods should be applicable to other domains as well. In the product search domain, the set of fields comprises type, brand, model, attribute, etc. It is possible to construct field-specific lexicons from a knowledge repository that enumerate all possible values for each field. A simple lexicon-based approach, however, is insufficient due to ambiguity (e.g., "*iphone*" may refer to model or to attribute) and the presence of out-of-vocabulary terms [46]. Li et al. [46] approach query tagging as a sequential labeling task and employ semi-supervised conditional random fields (CRF). A small amount of hand-labeled queries are combined with a large amount of queries with derived labels, which are obtained in an unsupervised fashion (by leveraging search click-through data in conjunction with a product database). Manshadi and Li [51] present a hybrid method, which consists of a generative grammar model and parser, and a discriminative re-ranking module. Li [44] further distinguishes between the semantic roles of the constituents in noun phrase queries and makes a distinction between *intent heads* (corresponding to attribute names) and *intent modifiers* (referring to attribute values). For example, given the query "*alice in wonderland 2010 cast*," "*cast*" is the intent head, while "*alice in wonderland*" and "*2010*" are intent modifiers, with labels *title* and *year*, respectively. Li [44] uses CRF-based models with transition, lexical, semantic (lexicon-based), and syntactic (POS tags) features. Pound et al. [61] annotate queries with semantic constructs, such as entity, type, attribute, value, or relationship. This mapping is learned from an annotated query log, using a CRF model with part-of-speech tags as features.

7.1.3 Query Interpretation

Instead of labeling individual query terms (or sequences of terms), *query interpretation* (a.k.a. *semantic query understanding*) aims to determine the meaning of the query as a whole, by figuring out how the segmented and annotated query tokens relate to each other and together form an "executable" expression.

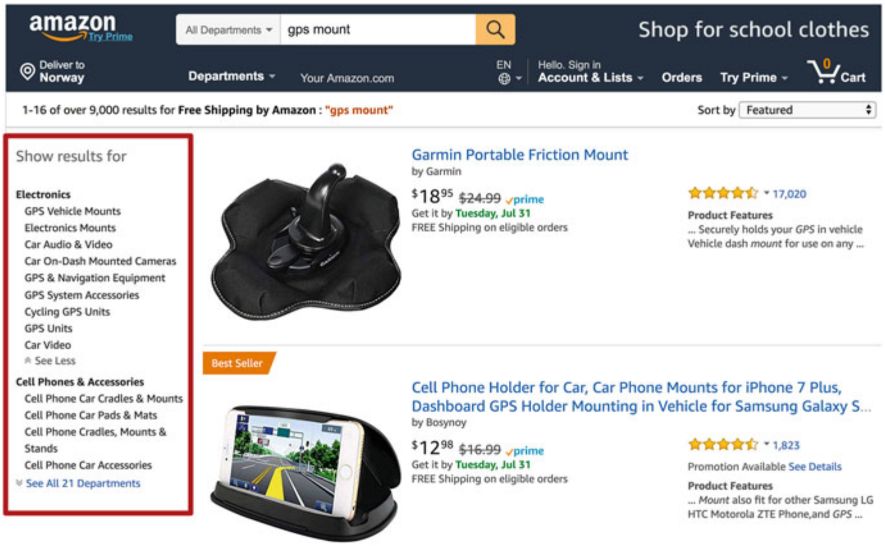
NLP approaches to question answering assume that the user's input is a grammatically well-formed question, from which a logical form (λ-calculus representation) can be inferred via *semantic parsing* [10, 80]. Viewing the knowledge base as a graph, these logical forms are tree-like graph patterns; executing them is equivalent to finding a matching subgraph of the knowledge graph.

In contrast, the database and IR communities generally operate with "telegraphic" queries. Due to the inherent ambiguity of such short keyword queries, systems typically need to evaluate multiple possible interpretations. In databases, the objective is to map queries to structured data tables and attributes; this task may also be referred to as *structured query annotation* [66]. For example, the query "*50 inch LG lcd tv*" could be mapped to the table *"TVs,"* with attributes *diagonal*=*"50 inch," brand*=*"LG,"* and *TV_type*=*"lcd."* We note that this is very similar to the problem of query tagging, which we have discussed in the previous subsection [44, 46, 51]. Nevertheless, there are two main differences. First, query tagging involves supervised or semi-supervised learning (typically using a CRF-based model), while the works on query interpretation presented here strive for an unsupervised solution. Second, query tagging does not describe how the various recognized semantic constructs interact; capitalizing on that structured data is organized in tables, the methods here do not allow for arbitrary combinations of attributes. Commonly, a two-tier architecture is employed, consisting of an offline and an online component [26, 59, 61, 66]. A collection of structured query templates are generated offline by mining patterns automatically from query logs. In the online component, all plausible interpretations are generated for an incoming query, where an interpretation is represented as a semantically annotated query and a query template. These interpretations are then scored to determine a single one that most likely captures the user's intent.

Very similar techniques are used in web search for querying specific verticals, by identifying the domain (vertical) of interest and recognizing specific attributes; we discuss some of these techniques in Sect. 7.4. Generally speaking, compared to NLP and databases, the goals in IR approaches to entity retrieval are somewhat more modest. The structural interpretation of queries consists of identifying mentions of entities and target types, with respect to an underlying knowledge base; see Sects. 7.2 and 7.3. The remaining "un-mapped" query words are used for scoring the textual descriptions of candidate entities. Notably, instead of aiming for a single "best" query interpretation, the system ranks responses for each possible interpretation, and then takes a weighted combination of scores over all interpretations [33, 67].

7.2 Identifying Target Entity Types

One way to understand the meaning behind a search query is to identify the entity types that are targeted by the query. In other words, we wish to map the query to a small set of target types (or categories), which boils down to the task of estimating the relevance of each type with respect to the query. The top-ranked types can then be leveraged in the entity ranking model (as we have seen earlier in this book, in Sect. 4.3) or offered to the user as facets. The latter form of usage is very typical, among others, on e-commerce sites, for filtering the search results; see Fig. 7.1 for an illustrative example. We note that the methods presented in this section are not limited to type taxonomies of knowledge bases, but are applicable to other type categorization systems as well (e.g., product categories of an e-commerce site).



1-16 of over 9,000 results for **Free Shipping by Amazon** : "gps mount"

Show results for

Electronics
- GPS Vehicle Mounts
- Electronics Mounts
- Car Audio & Video
- Car On-Dash Mounted Cameras
- GPS & Navigation Equipment
- GPS System Accessories
- Cycling GPS Units
- GPS Units
- Car Video
- See Less

Cell Phones & Accessories
- Cell Phone Car Cradles & Mounts
- Cell Phone Car Pads & Mats
- Cell Phone Cradles, Mounts & Stands
- Cell Phone Car Accessories
- See All 21 Departments

Garmin Portable Friction Mount
by Garmin
$18.95 ~~$24.99~~ prime
Get it by Tuesday, Jul 31
FREE Shipping on eligible orders
17,020
Product Features
... Securely holds your *GPS* in vehicle Vehicle dash *mount* for use on any ...

Cell Phone Holder for Car, Car Phone Mounts for iPhone 7 Plus, Dashboard GPS Holder Mounting in Vehicle for Samsung Galaxy S...
by Bosynoy
$12.98 ~~$16.99~~ prime
Get it by Tuesday, Jul 31
FREE Shipping on eligible orders
1,823
Promotion Available See Details
Product Features
... *Mount* also fit for other Samsung LG HTC Motorola ZTE Phone,and *GPS* ...

Fig. 7.1 Product categories displayed on Amazon in response to the query "gps mount"

7.2.1 Problem Definition

The problem of finding the target entity type(s) of the query can be formulated in different ways. The *entity type ranking* task, introduced by Vallet and Zaragoza [77], is as follows: Given an input query q, return a ranked list of entity types from a set $\mathcal{T}$ of possible types. Balog and Neumayer [5] investigate a variant of this task, called *hierarchical target type identification* (HTTI), which considers the hierarchical nature of entity type taxonomies. They aim to "find the single most specific type from an ontology [type taxonomy] that is general enough to cover all entities that are relevant to the query" [5]. Garigliotti et al. [27] refine this task definition (HTTIv2) to allow for a query to possibly have multiple "main" target types, provided they are sufficiently different, i.e., they lie on different branches in the type taxonomy. Furthermore, a query is also allowed to not have any target types, by assigning a special NIL-type. We adopt the revised task definition in [27], which is as follows:

Definition 7.1 *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.

Table 7.1 Notation used in Sect. 7.2

Symbol	Meaning
$c(t;x)$	Count of term t in x (x can be, e.g., an entity e or a type y)
e	Entity ($e \in \mathcal{E}$)
$\mathcal{E}$	Entity catalog (set of all entities)
$\mathcal{E}_q(k)$	Top-k ranked entities for query q
q	Query ($q = \langle q_1, \ldots, q_n \rangle$)
t	Term
$\mathcal{T}$	Type taxonomy
$\mathcal{T}_e$	Set of types assigned to entity e
y	Entity type ($y \in \mathcal{T}$)

Approached as a ranking problem, target entity type identification is the task of estimating the relevance of an entity type $y \in \mathcal{T}$ given a query q, expressed as $score(y;q)$. We present both unsupervised and supervised approaches in Sects. 7.2.2 and 7.2.3. Note that the NIL-type element of the problem is currently not considered. The different variants of the task definition essentially boil down to how the relevance assessments are obtained. We discuss this in Sect. 7.2.4. Table 7.1 summarizes the notation used in this section.

7.2.2 *Unsupervised Approaches*

According to one strategy, referred to as the *type-centric model* in [5], a term-based representation is built explicitly for each type; types can then be matched against the query using existing document retrieval models. Another strategy is to rank entities with respect to the query and then determine each type's relevance by considering the retrieval scores of entities of that type [41, 77]. Since types are not modeled directly, this approach is referred to as the *entity-centric model* [5].

An analogy can be drawn between these two approaches and the methods we have looked at earlier in the book for ranking entities. Specifically, the type-centric model corresponds to constructing term-based entity representations from documents mentioning those entities (Sect. 3.2), while the entity-centric model is akin to ranking entities without direct representations (Sect. 3.4). Essentially, one simply needs to replace entities with types and documents with entities in the equations. Zhang and Balog [82] further generalize these two retrieval strategies as design patterns for ranking arbitrary objects, called *early fusion* and *late fusion*, respectively.

7.2.2.1 Type-Centric Model

We construct a term-based representation ("type description") for each type, by aggregating descriptions of entities that are assigned that type. Then, those type representations can be ranked using conventional document retrieval methods. The term pseudo-counts for a type are computed using the following formula:

$$\tilde{c}(t;y) = \sum_{e \in \mathcal{E}} c(t;e)\, w(e,y) \,, \tag{7.1}$$

where $c(t;e)$ is the count of term t in the description of entity e and $w(e,y)$ denotes the entity-type association weight. This latter quantity may be interpreted as the importance of entity e for type y. A naïve but effective option is to set this weight uniformly across all entities that are typed with y:

$$w(e,y) = \begin{cases} \frac{1}{|\{e' : y \in \mathcal{T}_{e'}\}|}, & y \in \mathcal{T}_e \\ 0, & y \notin \mathcal{T}_e \,, \end{cases} \tag{7.2}$$

where $\mathcal{T}_e$ denotes the set of types assigned to entity e. Substituting Eq. (7.2) back to Eq. (7.1) we get:

$$\tilde{c}(t;y) = \frac{1}{|\{e : y \in \mathcal{T}_e\}|} \sum_{e \in \mathcal{E}} \mathbb{1}(y \in \mathcal{T}_e)\, c(t;e) \,,$$

where $\mathbb{1}(y \in \mathcal{T}_e)$ returns 1 if entity e has y as one of its assigned types, otherwise returns 0. As the rewritten equation shows, this particular choice of entity-type association weighting merely serves as a normalization factor, so that types have comparable term pseudo-counts (representation lengths), irrespective of the number of entities that belong to them.

Given the term-based type representation, as defined by $\tilde{c}(t;y)$, types can be ranked using any standard retrieval method. Following [82], the final scoring may be formulated (for bag-of-words retrieval models) as:

$$score_{TC}(y;q) = \sum_{i=1}^{n} score(q_i;\tilde{c},\varphi) \,, \tag{7.3}$$

where $score(q_i;\tilde{f},\varphi)$ assigns a score to each query term q_i, based on the term pseudo-counts $\tilde{c}$, using some underlying term-based retrieval model (e.g., LM or BM25), which is parameterized by φ.

7.2.2.2 Entity-Centric Model

Instead of building a direct term-based representation of types, the *entity-centric model* works as follows. First, entities are ranked based on their relevance to the query. Then, the score for a given type y is computed by aggregating the relevance scores of the top-k entities with that type:

$$score_{EC}(y;q) = \sum_{e \in \mathcal{E}_q(k)} score(e;q)\, w(e,y) \,, \tag{7.4}$$

where $\mathcal{E}_q(k)$ denotes the set of top-k ranked entities for query q. The retrieval score of entity e is denoted by $score(q; e)$, which can be computed using any of the entity retrieval models discussed in Chaps. 3 and 4. The entity-type association weight, $w(e, y)$, is again as defined by Eq. (7.2). One advantage of this approach over the type-centric model is that it can directly benefit from improved entity retrieval. Also, out of the two unsupervised approaches, the entity-centric model is the more commonly used one [5, 41, 77].

7.2.3 Supervised Approach

Balog and Neumayer [5] observe that "the type-centric model tends to return more specific categories [types], whereas the entity-centric model rather assigns more general types." The complementary nature of these two approaches can be exploited by combining them. Additionally, one can also incorporate additional signals, including taxonomy-driven features [73] and various similarity measures between the type label and the query [78]. Table 7.2 presents a set of features for the target entity type identification task, compiled by Garigliotti et al. [27].

Here, we only detail two of the best performing features, both of which are based on distributional similarity between the type's label and the query. The first one, *simAggr*, considers the centroids of embedding vectors of terms in the type's label and terms in the query:

$$simAggr(y, q) = \cos(\mathbf{y}, \mathbf{q}) \ ,$$

where $\mathbf{y}$ and $\mathbf{q}$ denote the respective centroid vectors.

The second one, *simMax*, takes the maximum pairwise similarity between terms in the type's label and in the query:

$$simMax(y, q) = \max_{t_y \in y, t_q \in q} \cos(\mathbf{t}_y, \mathbf{t}_q) \ ,$$

where $\mathbf{t}_y$ and $\mathbf{t}_q$ are the embedding vectors corresponding to terms t_y and t_q, respectively. In both cases, pre-trained 300-dimensional Word2vec [55] vector embeddings were used. Further, the set of terms considered is limited to "content words" (i.e., nouns, adjectives, verbs, or adverbs) [27].

7.2.4 Evaluation

Next, we present evaluation measures and test collections for the target entity type identification task.

Table 7.2 Features for target entity type identification

Group	Feature	Description
Baseline features		
	$score_{TC}(y;q)$	Type-centric type score (cf. Eq. (7.3))[a]
	$score_{EC}(y;q)$	Entity-centric type score (cf. Eq. (7.4))[a, b]
Type taxonomy features		
	$depth(y)$	Hierarchical level of y, normalized by the taxonomy depth
	$children(y)$	Number of children of type y in the taxonomy
	$siblings(y)$	Number of siblings of type y in the taxonomy
	$numEntities(y)$	Number of entities in the KB with type y ($\|\{e \in \mathcal{E} : y \in \mathcal{T}_e\}\|$)
Type label features		
	l_y	Length of (the label of) type y in terms
	$sumIDF(y)$	Sum of IDF for terms in (the label of) type y ($\sum_{t \in y} IDF(t)$)
	$avgIDF(y)$	Avg. of IDF for terms in (the label of) type y ($\frac{1}{l_y}\sum_{t \in y} IDF(t)$)
	$sim_{JAC}(y,q)$	Jaccard similarity between terms of the type label and of the query
	$sim_{JACnouns}(y,q)$	Jaccard similarity between the type and the query, restricted to terms that are nouns
	$simAggr(y,q)$	Cosine sim. between the centroid embedding vectors of q and y
	$simMax(y,q)$	Max. cosine similarity of the embedding vectors between each pair of query and type label terms
	$simAvg(y,q)$	Avg. cosine similarity of the embedding vectors between each pair of query and type label terms

[a] Instantiated using both BM25 and LM as the underlying term-based retrieval model
[b] Instantiated with multiple cutoff thresholds $k \in \{5, 10, 20, 50, 100\}$

7.2.4.1 Evaluation Measures

Being a ranking task, type ranking is evaluated using standard rank-based measures. The ground truth annotations consist of a small set of types for each query, denoted as $\hat{\mathcal{T}}_q$. It might also be the case that a query has no target types ($\hat{\mathcal{T}}_q = \emptyset$); in that case the query might be annotated with a special NIL-type. Detecting NIL-types, however, is a separate task, which is still being researched, and which we do not deal with here. That is, we assume that $\hat{\mathcal{T}}_q \neq \emptyset$. The default setting is to take type relevance to be a binary decision; for each returned type y, it either matches one of the ground truth types (1) or not (0). The evaluation measures are mean average precision (MAP) and mean reciprocal rank (MRR). We shall refer to this as *strict* evaluation.

However, not all types of mistakes are equally bad. Imagine that the target entity type is *racing driver*. Then, 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*). Balog and Neumayer [5] accommodate this by introducing *lenient* evaluation, where relevance is graded and near-misses are rewarded. Specifically, let $d(y, \hat{y})$ denote the distance between two types, the returned type y and the ground truth type $\hat{y}$, in the taxonomy. This distance 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, their distance is set to ∞. With the help of this distance function, the relevance level or gain of a type can be defined in a *linear* fashion:

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

where h is the depth of the type taxonomy. Notice that we consider the closest matching type from the set of ground types $\hat{\mathcal{T}}_q$. Alternatively, distance may be turned into a relevance level using an *exponential* decay function:

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

where b is the base of the logarithm (set to 2 in [5]). If $d(y, \hat{y}) = \infty$, then the value of the exponential is taken to be 0. In the lenient evaluation mode, the final measure is normalized discounted cumulative gain (NDCG), using the above (linear or exponential) relevance gain values.

7.2.4.2 Test Collections

Balog and Neumayer [5] annotated 357 entity-oriented search queries with a single target type, using the DBpedia Ontology as the reference type taxonomy. These queries are essentially a subset of the ones in the DBpedia-Entity collection (cf. Sect. 3.5.2.7). According to their task definition, an "instance of" relation is required between the target type and relevant entities (as opposed to mere "relatedness," as in [77]). The guideline for the annotation process was to "pick a single type that is as specific as possible, yet general enough to cover all correct answers" [5]. For 33% of the queries, this was not possible because of one of the following three main reasons: (1) the query has multiple (legitimate) target types; e.g., "Ben Franklin" is a *Person* but may also refer to the ship (*MeanOfTransportation*) or the musical (*Work*); (2) there is no appropriate target type in the taxonomy for the given query; or (3) the intent of the query is not clear.

In follow-up work, Garigliotti et al. [27] annotated the complete set of queries from the DBpedia-Entity collection, in accordance with their revised task definition (cf. Sect. 7.2.1). Correspondingly, queries can have multiple target types or none (NIL-type option). The relevance assessments were obtained via crowdsourcing, using a newer (2015-10) version of the DBpedia Ontology as the type taxonomy. Around 58% of the queries in the collection have a single target type; the rest of the queries have multiple (mostly two or three) target types, including the NIL-type.

7.3 Entity Linking in Queries

Identifying entities in queries is another key technique that enables a better understanding of the underlying search intents. According to Guo et al. [28], over 70% of queries in web search contain named entities. The study by Lin et al. [49] reports a lower number, 43%, albeit using different annotation guidelines. The bottom line is that by being able to recognize entities in queries, the user experience can be improved for a significant portion of search requests (e.g., by enhanced result ranking or presentation). In Chap. 5, we have dealt in detail with the problem of *entity linking*, i.e., annotating documents with entities from a reference knowledge repository. Why can we not simply apply the same techniques to search queries? The reasons are at least threefold.

- One challenge is that queries are very short, typically consisting only of a few terms, and lack proper spelling and grammar. There is experimental evidence showing that methods perform substantially worse on very short and poorly composed texts (tweets) than on longer documents (news) [18, 50]. What is more, even methods that are designed in particular for short text perform significantly worse on queries than those that are specifically devised for queries [19].
- Another fundamental difference is that when documents are annotated with entities, “it is implicitly assumed that the text provides enough context for each entity mention to be resolved unambiguously” [32]. For queries, on the other hand, there is only limited context, or none at all.[4] It may be impossible to annotate entity mentions unambiguously in the case of queries. That is, a given query segment can possibly be linked to more than a single entity, leading to multiple legitimate interpretations of the query.
- Finally, obtaining entity annotations for queries is an online process that needs to happen during query-time, under serious time constraints. This is unlike annotating documents, which is typically performed offline. Therefore, we are not necessarily looking for the most effective solution, but for “one that represents the best trade-off between effectiveness and efficiency” [34].

The task of annotating queries with entities has been studied in a number of different flavors; we start with presenting an overview of these in Sect. 7.3.1. Then, in Sect. 7.3.2, we introduce a unified pipeline approach. The two main components of this pipeline are detailed in Sects. 7.3.3 and 7.3.4. Table 7.3 shows the notation used in this section.

[4] Search history information may provide contextual anchoring; this, however, is often unavailable. For example, if it is the first query in a search session, with no information about the previous searches of the user (which corresponds to the commonly studied ad hoc search scenario), then the query text is all we have.

Table 7.3 Notation used in Sect. 7.4

Symbol	Meaning
$\mathcal{A}$	Annotation (set of mention-entity pairs)
C	Candidate entity annotations (ordered list)
e	Entity ($e \in \mathcal{E}$)
$\mathcal{E}$	Entity catalog (set of all entities)
$\mathcal{I}$	Query interpretation ($\mathcal{I} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$)
l_x	Length of x (number of terms)
m	Entity mention
$\mathcal{M}_q$	Set of entity mentions identified in query q
q	Query
t	Term

7.3.1 Entity Annotation Tasks

We distinguish between three entity annotation tasks, formulated for queries rather than for regular text. Table 7.4 highlights the differences between them, along with some illustrative examples.

- *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.
- *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.

7.3.1.1 Named Entity Recognition

The task of *named entity recognition in queries* (NERQ) is analogous to the problem of named entity recognition in text (NER, cf. Sect. 5.1.1), namely: Identify named entities in the query text and classify them with respect to a set of predefined types from a taxonomy. NERQ was introduced and first studied by Guo et al. [28], who employed topic modeling techniques with weak supervision (WS-LDA). Pantel et al. [58] expanded upon this work by incorporating latent user intents and click signals.

Table 7.4 Comparison of various entity annotation tasks for queries

	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[b]
Evaluation criteria	Recognized entities[a]	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}}

[a] Along with their respective types
[b] Not within the same interpretation

7.3.1.2 Semantic Linking

Semantic linking refers to the task of identifying entities "that are intended or implied by the user issuing the query" [53]. This problem was introduced as *query mapping* by Meij et al. [53] and is also known as *semantic mapping* [32] and as *(ranked) concepts to Wikipedia* [18]. As the name we have adopted suggests, we seek to find entities that are semantically related to the query. Notably, the entities to be linked are meant for human and not for machine consumption (e.g., to help users acquire contextual information or to provide them with valuable navigational suggestions [53]). Therefore, we are not so much interested in detecting the specific entity mentions in the query, nor do we require the returned entities to form a coherent set. Further, an entity may be semantically related (i.e., relevant) even if it is not explicitly mentioned in the query. Take, e.g., the query "*charlie sheen lohan*," for which Anger Management (TV series) would be a relevant entity. Mind that this is different from the task of entity retrieval; our 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.

7.3.1.3 Interpretation Finding

Interpretation finding is the query counterpart to entity disambiguation, where the inherent ambiguity of queries is addressed head-on. A query "can legitimately have more than one interpretation" [17], where an interpretation is a set of "non-overlapping linked entity mentions that are semantically compatible with the query

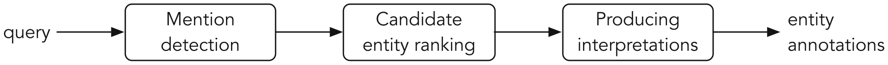


Fig. 7.2 Entity linking in queries pipeline

text" [17]. For example, the query "*new york pizza manhattan*" might be interpreted as the user wanting to eat a pizza in the MANHATTAN borough of NEW YORK CITY, or the user desiring a specific pizza flavor, NEW YORK-STYLE PIZZA, also in MANHATTAN. Interpretation finding aims at machine understanding of queries. The resulting annotations are utilized in the subsequent ranking process. A pioneering effort in this area was the Entity Recognition and Disambiguation (ERD) Challenge in 2014, organized by representatives of major web search engine companies [17]. Our ultimate interest in this section is on interpretation finding. In the next subsection, we shall present a pipeline architecture for addressing this task.

7.3.2 Pipeline Architecture for Interpretation Finding

We present a pipeline architecture for entity linking in queries, i.e., for the interpretation finding task, shown in Fig. 7.2. Notice that it is very similar to the one we employ for entity linking for documents (see Sect. 5.3). One important characteristic of this pipeline approach is it unifies the three tasks we discussed in the previous section under a common framework and shows how these tasks build on each other.

- The first step, *mention detection*, can be performed exactly the same way for queries as it is done for documents, i.e., using an extensive dictionary of entity surface forms; see Sect. 5.4.
- The *candidate entity ranking* step, as the name suggests, produces a ranking of candidate entities for the query. Specifically, given a set of mentions as input from the previous step, it emits a list of mention-entity pairs ordered by their degree of semantic relatedness to the query. Notice that this step directly translates to the task of *semantic linking*. One thing to point out here is that for the semantic linking task the actual mentions are ignored (i.e., for each entity only the highest scoring mention counts), while for interpretation finding the mentions also need to be passed along.
- Finally, *producing interpretations* is the query counterpart of the disambiguation component in conventional entity linking. The candidate entities identified in the previous step are used to form one or multiple query interpretations, where each interpretation consists of a set of semantically coherent entity linking decisions, with non-overlapping entity mentions.

In the next two subsections, we look at the candidate entity ranking and interpretation finding steps in detail. The details of mention detection are relatively straightforward, as it is done analogously to entity linking in documents; see Sect. 5.4.

7.3.3 Candidate Entity Ranking

Given a query q, the problem of *candidate entity ranking* is to return a ranked list of entities $\langle e_1,\ldots,e_k\rangle$ from an entity catalog $\mathcal{E}$ that are semantically related to the query. We shall assume that a set $\mathcal{M}_q$ of entity mentions has already been identified in the query (see Sect. 5.4 for methods). For each mention $m \in \mathcal{M}_q$, let $\mathcal{E}_m$ denote the set of candidate entities, i.e., entities that have a surface form matching m. This candidate set may be further restricted to entities above a certain *commonness* threshold (cf. Sect. 5.5). For example, in [32], a commonness threshold of 0.1 is used. The goal, then, is to rank all candidate entities mentioned in q, $\mathcal{E}_q = \{e : e \in \mathcal{E}_m, m \in \mathcal{M}_q\}$, based on $score(e;q,m)$, their semantic relatedness to the query. Below, we present both unsupervised and supervised solutions for estimating this score. Additionally, in Sect. 7.3.3.3, we introduce the "piggybacking" technique, which is directed to alleviating the brevity of queries.

7.3.3.1 Unsupervised Approach

Hasibi et al. [32] propose to rank entities by combining term-based similarity score with the commonness measure, using the following formula:

$$score(e;q,m) = P(e|q,m) \propto P(e|m)P(q|e)\ .$$

For the commonness computation, $P(e|m)$, we refer back to Eq. (5.3). While written as a probability, the term-based similarity, $P(q|e)$, may in fact be computed using any of the methods we presented in Chap. 3. What is important is that if the final task is interpretation finding, these scores need to be comparable across queries. One specific instantiation of this approach, referred to as *MLMcg* in [32], estimates $P(q|e)$ using the query length normalized language model similarity [42]:

$$P(q|e) = \frac{\prod_{t\in q} P(t|\theta_e)^{P(t|q)}}{\prod_{t\in q} P(t|\mathcal{E})^{P(t|q)}}\ , \tag{7.5}$$

where $P(t|\theta_e)$ and $P(t|\mathcal{E})$ are the entity and collection language models, respectively, computed using the mixture of language models (MLM) approach, cf. Sect. 3.3.2.1. $P(t|q)$ is the term's relative frequency in the query $(c(t;q)/l_q)$.

Table 7.5 Features for candidate entity ranking

Group	Feature	Description
Mention		
	l_m	Length of mention m (number of terms)
	$\|\mathcal{E}_m\|$	Number of candidate entities for the mention
	$P(\text{link}\|m)$	Link probability (cf. Eq. (5.2))
	$\mathit{nameMatch}(m)$	Number of entities with a surface form equal to mention m
	$\mathit{partialMatch}(m)$	Number of entities with a surface form partially matching m
Entity		
	$\mathit{redirects}(e)$	Number of Wikipedia redirect pages linking to the entity
	$\mathit{links}(e)$	Number of in/out-links of the entity in the knowledge graph
	$\mathit{pageRank}(e)$	PageRank of e in the knowledge graph
	$\mathit{pageViews}(e)$	Number of (Wikipedia) page views e received
Mention-entity		
	$P(e\|m)$	Commonness (the probability of e being the link target of m)
	$\mathit{contains}(m,e)$	Whether the mention contains a surface form of the entity
	$\mathit{contains}(e,m)$	Whether a surface form of the entity contains the mention
	$\mathit{equals}(e,m)$	Whether a surface form of the entity equals the mention
	$\mathit{editDist}(m,e)$	Edit distance between the mention and the (best matching) surface form of the entity
	$\mathit{firstPos}(e,m)$	Position of the first occurrence of the mention in the entity's description in the knowledge repository
	$\mathit{sim}(m, f_e)$	Similarity between m and field f of the entity (cf. Sect. 3.3)
Query		
	$\mathit{lenRatio}(m,q)$	Mention to query length ratio (l_m/l_q)
	$\mathit{contains}(q,e)$	Whether the query contains a surface form of the entity
	$\mathit{contains}(e,q)$	Whether a surface form of the entity contains the query
	$\mathit{equals}(e,q)$	Whether a surface form of the entity equals the query
	$\mathit{sim}(q,e)$	Similarity between the query and entity (cf. Sect. 3.3)
	$\mathit{sim}(q, f_e)$	Similarity between q and field f of the entity (cf. Sect. 3.3)

7.3.3.2 Supervised Approach

Using a supervised learning approach, each (entity, query, mention) triple is described using a set of features. Table 7.5 displays a selection of features, assembled from the literature [19, 34, 54]. These are organized into four main groups:

- *Mention features* represent the characteristics of the specific mention.
- *Entity features* draw only from entity properties.
- *Mention-entity features* capture the binding between mentions and entities.
- *Query features* express query-mention and query-entity relationships.

We note that our assortment of features is by no means exhaustive. Also, we have limited ourselves to features that can be computed from publicly available resources; we refer to Blanco et al. [12] for additional query log-based features.

The supervised ranking model is trained on a set of labeled examples; for each mention and candidate entity pair, the target label is 1 if the entity is the correct link target of the given mention and is 0 otherwise.

7.3.3.3 Gathering Additional Context

One of the main challenges when annotating queries with entities is the lack of context. Meij et al. [53] develop features based on previous queries that the user issued in the same session. This method, however, is subject to the availability of session history. (Also, these features do not seem to make a significant contribution to the best results in [53].) Cornolti et al. [19] employ the so-called piggybacking technique (first introduced in [64]): Submitting the query to a web search engine API, and using the returned result snippets as additional context for entity disambiguation. The top-ranked result snippets are usually of very high quality, thanks to "sophisticated algorithms that leverage huge indexed document collections (the whole web), link graphs, and log analysis" [19]. The piggybacking technique has an additional benefit of being able to automatically correct spelling errors, by accessing the spelling correction feature of web search APIs. On the downside, it should be pointed out that the reliance on an external search service can seriously hinder the efficiency of the annotation process. Furthermore, there is no control over the underlying document ranking algorithm (which may change without notice).

Specifically, Cornolti et al. [19] retrieve the top 25 snippets using the original query. In addition to that, they also concatenate the original query with the string "*wikipedia*" and take the first 10 snippets, with the intention to boost results from Wikipedia. These search engine results are then used both for identifying candidate entities and for scoring them. Candidate entities are recognized in two ways:

1. Wikipedia articles occurring in the top-ranked results can be directly mapped to entities.
2. Annotating the result snippets with entities (using an entity linker designed for short text; Cornolti et al. [19] use WAT [60]) and keeping only those annotations that overlap with bold-highlighted substrings (reflecting query term matches) in the snippets.

In the candidate entity scoring phase, this information is utilized via various features, including the frequency and rank positions of mentions in snippets.

7.3.3.4 Evaluation and Test Collections

As we have explained earlier, the candidate entity ranking component of our pipeline corresponds to the semantic linking task. It is evaluated as a ranking problem, using standard rank-based measures, such as (mean) average precision and (mean) reciprocal rank. For each entity, only the highest scoring mention is considered:

$$score(e;q) = \arg\max_{m \in \mathcal{M}_q} score(e;q,m) \ .$$

Table 7.6 Test collections for evaluating semantic linking

Name	Reference KR	#Queries total	#Queries ≥1 annot.	Annotations can overlap	Sessions considered
YSQLE[a]	Wikipedia	2653	2583	Yes	Yes
GERDAQ [19][b]	Wikipedia	1000	889	No	No

[a] Yahoo! Webscope L24, https://webscope.sandbox.yahoo.com/
[b] http://acube.di.unipi.it/datasets/

At the time of writing, two publicly available test collections exist, which are summarized in Table 7.6.

Yahoo Search Query Log to Entities (YSQLE) The dataset consists of 2635 web search queries, out of which 2583 are manually annotated with entities from Wikipedia. The annotations have been performed within the context of a search session (there are 980 sessions in total). Each linked entity is aligned with the specific mention (query span). Additionally, a single entity for each query may be labeled as "main," if it represents the main intent of the query. For example, the query "*France 1998 final*" is annotated with three entities: FRANCE NATIONAL FOOTBALL TEAM, FRANCE, and 1998 FIFA WORLD CUP FINAL, the last one being the main annotation.

GERDAQ This collection, created by Cornolti et al. [19], consists of 1000 queries sampled from the KDD-Cup 2005 dataset [48]. The queries were annotated with Wikipedia entities via crowdsourcing, in two phases (first recall-oriented, then precision-oriented). The resulting dataset was then randomly split into training, development, and test sets, comprising 500, 250, and 250 queries, respectively. Each entity mention is linked to the highest scoring entity, according to the human annotators. Entity annotations do not overlap; entities below a given score threshold are discarded. On average, each query is annotated with two entities.

7.3.4 Producing Interpretations

In conventional entity linking, the generated annotations comprises a set of (semantically compatible) mention-entity pairs: $\mathcal{A} = \{(m_1,e_1),\dots,(m_k,e_k)\}$, where e_i is the entity corresponding to mention m_i, and mention offsets must not overlap. In the context of queries, we shall refer to one such possible entity annotation as *interpretation*. Due to ambiguity, a query might have more than a single interpretation. Therefore, the objective of this component is to produce a set of query interpretations, $\mathcal{I} = \{\mathcal{A}_1,\dots,\mathcal{A}_n\}$, where $\mathcal{A}_i$ is an interpretation.

We shall assume that all entity mentions in the query have been recognized and scored in a prior step (cf. Sect. 7.3.3). We shall refer to these as *candidate annotations*, and denote them as the list $C = \langle(m_1,e_1,s_1),\dots,(m_k,e_k,s_k)\rangle$, where each annotation is a triple consisting of mention m_i, entity e_i, and score s_i. The list

Algorithm 7.1: Greedy interpretation finding [32]

```
Input: Candidate annotations $C$, score threshold $\tau$
Output: Interpretations $\mathcal{I}$
$C' \leftarrow prune(C, \tau)$
$C' \leftarrow pruneContainmentMentions(C')$
$\mathcal{I} \leftarrow createInterpretations(C')$
return $\mathcal{I}$

Function createInterpretations($C$):
    $\mathcal{I} \leftarrow \emptyset$
    for $(m, e, s) \in C$ ordered by $s$ do
        $h \leftarrow False$
        for $\mathcal{A}_i \in \mathcal{I}$ do
            if $\neg$ $hasOverlap(m, \mathcal{A}_i)$ then /* Add to existing interpretation */
                $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{(m, e)\}$
                $h \leftarrow True$
            end
        end
        if $\neg h$ then                          /* Create new interpretation */
            $\mathcal{I} \leftarrow \mathcal{I} \cup \{(m, e)\}$
        end
    end
    return $\mathcal{I}$
```

C is ordered by decreasing score. Further, C may be truncated to a certain number of elements (top-k) or to annotations above a minimum score threshold. The task, then, is to form the set of interpretations $\mathcal{I}$, given the candidate annotations C as input.

We note that if one is to find only a single most likely interpretation, that can be done using existing entity linking methods (esp. using those that have been developed for annotating short text, such as TAGME [23] or WAT [60]). Even studies that address entity linking in queries often resort to the simpler problem of finding a single interpretation [12, 19]. Specifically, the top interpretation may be created greedily, by adding candidate annotations in decreasing order of score, as long as (1) they do not overlap and (2) the scores are above a given threshold (SMASH-S [19]). Alternatively, entity annotations may be selected using dynamic programming, such that they maximize the overall query likelihood [12]. In both cases, entity linking decisions are made individually, independently of each other (other than the constraint of non-overlapping). Our interest here, nevertheless, is focused on finding multiple interpretations, and we present both unsupervised (Sect. 7.3.4.1) and supervised (7.3.4.2) approaches for that.

7.3.4.1 Unsupervised Approach

Hasibi et al. [32] present the *greedy interpretation finding* (GIF) algorithm, shown in Algorithm 7.1, which consists of three steps: (1) pruning, (2) containment mention filtering, and (3) set generation. In the first step, the algorithm takes all candidate annotations and discards those with a score below the threshold τ. The threshold parameter is used to control the balance between precision and recall. In the second step, containment mentions are filtered out, by keeping only the highest scoring one. For instance, "*kansas city mo*," "*kansas city*, " and "*kansas*" are containment mentions; only a single one of these three is kept. Finally, interpretations are built iteratively, by processing the filtered candidate annotations C' in decreasing order of score. A given mention-entity pair (m,e) is added to an existing interpretation $\mathcal{A}_i$, if m does not overlap with the mentions already in $\mathcal{A}_i$; in the case multiple such interpretations exist, the pair (m,e) will be added to all of them. Otherwise, a new interpretation is created from the mention-entity pair. The GIF algorithm, despite its simplicity, is shown to be very effective and is on par with considerably more complex systems [34]. Its performance, however, crucially depends on that of the preceding candidate entity ranking step.

7.3.4.2 Supervised Approach

The main idea behind the *collective disambiguation* approach is to carry out a joint analysis of groups of mention-entity pairs, rather than greedily selecting the highest scoring entity for each mention. This idea can be realized by considering multiple possible candidate interpretations of the query and then applying supervised learning to select the most likely one(s). If the goal is to find multiple interpretations, then it is cast as a binary classification problem. If the objective is to find only the single most likely interpretation, then it may be approached as a ranking (regression) task.

As we have pointed out in Sect. 5.6.2.3, the joint optimization of entity annotations in text is an NP-hard problem. However, since queries are typically short, it is possible to enumerate all sensible combinations of entities. Specifically, Hasibi et al. [34] consider only the top-k candidate annotations for forming interpretations. The value of k can be chosen empirically, based on the effectiveness of the underlying candidate entity ranking component. In [34] $k = 5$ is used; less effective candidate entity ranking approaches may be compensated for by choosing a larger k value. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ denote the candidate interpretations, which are generated by enumerating all possible valid (non-overlapping) combinations of mention-entity pairs from the candidate annotations C. For a given interpretation $\mathcal{A}_i$, we let $\mathcal{E}_i$ be the set of linked entities in that interpretation: $\mathcal{E}_i = \{e : (m,e) \in \mathcal{A}_i\}$.

The key challenge, then, is to design features that can capture the coherence of multiple entity annotations in the query. Two main groups of features are employed in [19, 34]:

- *Entity features* express the binding between the query and the entity and depend only on the individual entity (and not on other entities mentioned in the query). The top block of Table 7.7 lists entity features that are used in [34]. We note that other features that have been introduced for the candidate entity ranking step may also be used here (see Table 7.5). These features are computed for each entity that

Table 7.7 Features for producing interpretations

Group	Feature	Description
Entity		
	$links(e)$	Number of out-links of the entity in the knowledge graph
	$P(e\|m)$	Commonness (the probability of e being the link target of m)
	$score(e;q)$	Score from the candidate entity ranking step
	$iRank(e,q)$	Inverse rank from the candidate entity ranking step ($1/rank(e,q)$)
	$sim(q,e)$	Similarity between the query and entity (cf. Sect. 3.3)
	$contextSim(q,e)$	Contextual similarity between the query and entity, where context is the "rest" of the query, without the entity mention
Interpretation		
	$\min(\mathcal{R}_i)$	Minimum relatedness among entities in $\mathcal{E}_i$
	$\max(\mathcal{R}_i)$	Maximum relatedness among entities in $\mathcal{E}_i$
	$P_{co}(\mathcal{E}_i)$	Co-occurrence probability of entities in a Web corpus (Eq. (7.6))
	$H(\mathcal{E}_i)$	Entropy of $\mathcal{E}_i$ (Eq. (7.7))
	$sim(q,\mathcal{E}_i)$	Similarity between the query and $\mathcal{E}_i$ (Eq. (7.8))
	$coverage(\mathcal{A}_i,q)$	Mention coverage (Eq. (7.9))

is part of a given candidate interpretation ($e \in \mathcal{E}_i$), then aggregated by taking the minimum, maximum, or average of the individual values. Thus, each of these features is computed three times, using the three different aggregators.

- *Interpretation features* aim to capture the coherence of the set of linked entities in a given (candidate) interpretation ($\mathcal{A}_i$). These are computed collectively for all linked entities in the interpretation ($\mathcal{E}_i$). A selection of interpretation features are listed in the bottom block of Table 7.7. We briefly explain them below.
  - For relatedness features, we let $\mathcal{R}_i$ be the set of all pairwise relatedness scores between all entities in $\mathcal{E}_i$:

$$\mathcal{R}_i = \{rel(e_k,e_l) : e_k,e_l \in \mathcal{E}_i, e_k \neq e_l\} ,$$

    where $rel(e_k,e_l)$ is a measure of entity-relatedness; commonly WLM relatedness or Jaccard similarity is used (see Sect. 5.6.1.3 for other options). We take the minimum and the maximum of the values in $\mathcal{R}_i$ as two features.
  - The co-occurrence probability of all the entities in $\mathcal{E}_i$ may be estimated using a large web corpus:

$$P_{co}(\mathcal{E}_i) = \frac{|\bigcap_{e\in\mathcal{E}_i} \mathcal{D}_e|}{|\mathcal{D}|} , \qquad (7.6)$$

    where $\mathcal{D}_e$ is the set of documents in which e occurs, and $|\mathcal{D}|$ is the total number of documents in the corpus. With the help of this probability, the *entropy* of $\mathcal{E}_i$ may be calculated as:

$$H(\mathcal{E}_i) = -P_{co}(\mathcal{E}_i)\log P_{co}(\mathcal{E}_i) - (1 - P_{co}(\mathcal{E}_i))\log(1 - P_{co}(\mathcal{E}_i)) . \qquad (7.7)$$

- The similarity between $\mathcal{E}_i$ and the query is calculated similarly to Eq. (7.5), but using a language model of all entities in the interpretation, instead of that of a single entity:

$$P(q|\mathcal{E}_i) = \frac{\prod_{t \in q} P(t|\theta_{\mathcal{E}_i})^{P(t|q)}}{\prod_{t \in q} P(t|\mathcal{E})^{P(t|q)}} , \tag{7.8}$$

  where the interpretation language model is estimated according to:

$$P(t|\theta_{\mathcal{E}_i}) = \frac{1}{|\mathcal{E}_i|} \sum_{e \in \mathcal{E}_i} P(t|\theta_e) .$$

- Mention coverage is the ratio of the query that is annotated, i.e., the length of all entity mentions over the length of the query:

$$coverage(\mathcal{A}_i, q) = \frac{\sum_{(m,e) \in \mathcal{A}_i} l_m}{l_q} , \tag{7.9}$$

  where l_m and l_q denote mention and query length, respectively.

7.3.4.3 Evaluation Measures

Let $\hat{\mathcal{I}} = \{\hat{\mathcal{A}}_1, \ldots, \hat{\mathcal{A}}_m\}$ denote the set of interpretations of query q according to the ground truth, and let $\mathcal{I} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ denote the system-generated interpretations. For comparing these two sets, Carmel et al. [17] define *precision* and *recall* as:

$$\mathrm{P} = \frac{|\mathcal{I} \cap \hat{\mathcal{I}}|}{|\mathcal{I}|} , \quad \mathrm{R} = \frac{|\mathcal{I} \cap \hat{\mathcal{I}}|}{|\hat{\mathcal{I}}|} .$$

Hasibi et al. [32] point out that "according to this definition, if the query does not have any interpretations in the ground truth ($\hat{\mathcal{I}} = \emptyset$) then recall is undefined; similarly, if the system does not return any interpretations ($\mathcal{I} = \emptyset$), then precision is undefined." Therefore, they define precision and recall for *interpretation-based evaluation* as follows:

$$\mathrm{P}_{int} = \begin{cases} |\mathcal{I} \cap \hat{\mathcal{I}}| / |\mathcal{I}|, & \mathcal{I} \neq \emptyset \\ 1, & \mathcal{I} = \emptyset, \hat{\mathcal{I}} = \emptyset \\ 0, & \mathcal{I} = \emptyset, \hat{\mathcal{I}} \neq \emptyset . \end{cases}$$

$$\mathrm{R}_{int} = \begin{cases} |\mathcal{I} \cap \hat{\mathcal{I}}| / |\hat{\mathcal{I}}|, & \hat{\mathcal{I}} \neq \emptyset \\ 1, & \hat{\mathcal{I}} = \emptyset, \mathcal{I} = \emptyset \\ 0, & \hat{\mathcal{I}} = \emptyset, \mathcal{I} \neq \emptyset . \end{cases}$$

An interpretation is taken to be correct only if it matches all the entities of an interpretation in the ground truth exactly. For simplicity, the correctness of the mention offsets are not considered. Formally:

$$|\mathcal{I} \cap \hat{\mathcal{I}}| = \sum_{\mathcal{A}_i \in \mathcal{I}, \hat{\mathcal{A}}_j \in \hat{\mathcal{I}}} match(\mathcal{A}_i, \hat{\mathcal{A}}_j) \,,$$

where

$$match(\mathcal{A}_i, \hat{\mathcal{A}}_j) = \begin{cases} 1, & \{e : e \in \mathcal{A}_i\} = \{e : e \in \hat{\mathcal{A}}_j\} \\ 0, & \text{otherwise} \,. \end{cases}$$

This evaluation is rather strict, as partial matches (for a given interpretation) are not given any credit. Alternatively, Hasibi et al. [32] propose a *lenient evaluation* that rewards partial matches. The idea is to combine interpretation-based evaluations (from above) with conventional entity linking evaluation, referred to as *entity-based evaluation*. Let $\mathcal{E}_{\mathcal{I}}$ denote the set of all entities from all interpretations returned by the entity linking system, $\mathcal{E}_{\mathcal{I}} = \cup_{i \in [1..n]} \{e : e \in \mathcal{A}_i\}$. Similarly, let the set $\hat{\mathcal{E}}_{\mathcal{I}}$ contain all entities from all interpretations in the ground truth, $\hat{\mathcal{E}}_{\mathcal{I}} = \cup_{j \in [1..m]} \{e : e \in \hat{\mathcal{A}}_j\}$. Then, precision and recall are defined as follows:

$$\mathrm{P}_{ent} = \begin{cases} |\mathcal{E}_{\mathcal{I}} \cap \hat{\mathcal{E}}_{\mathcal{I}}| / |\mathcal{E}_{\mathcal{I}}|, & \mathcal{E}_{\mathcal{I}} \neq \emptyset \\ 1, & \mathcal{E}_{\mathcal{I}} = \emptyset, \hat{\mathcal{E}}_{\mathcal{I}} = \emptyset \\ 0, & \mathcal{E}_{\mathcal{I}} = \emptyset, \hat{\mathcal{E}}_{\mathcal{I}} \neq \emptyset \,. \end{cases}$$

$$\mathrm{R}_{ent} = \begin{cases} |\mathcal{E}_{\mathcal{I}} \cap \hat{\mathcal{E}}_{\mathcal{I}}| / |\hat{\mathcal{E}}_{\mathcal{I}}|, & \hat{\mathcal{E}}_{\mathcal{I}} \neq \emptyset \\ 1, & \hat{\mathcal{E}}_{\mathcal{I}} = \emptyset, \mathcal{E}_{\mathcal{I}} = \emptyset \\ 0, & \hat{\mathcal{E}}_{\mathcal{I}} = \emptyset, \mathcal{E}_{\mathcal{I}} \neq \emptyset \,. \end{cases}$$

Finally, the overall precision and recall, in lenient evaluation, are defined as a linear combination of interpretation-based and entity-based precision and recall:

$$\mathrm{P} = \frac{\mathrm{P}_{int} + \mathrm{P}_{ent}}{2} \,, \quad \mathrm{R} = \frac{\mathrm{R}_{int} + \mathrm{R}_{ent}}{2} .$$

For simplicity, precision and recall are averaged with equal weights, but a weight parameter could also be introduced here. The F-measure (for any definition of precision and recall above) is computed according to Eq. (5.10).

So far, we have defined evaluation measures for a single query. For computing precision, recall, and F-measure over a set of queries, the (unweighed) average of the query-level scores is taken (i.e., macro-averaging is used).

Table 7.8 Test collections for interpretation finding

Name	Reference KR	#Queries total	#Queries ≥1 annot.	#Queries ≥1 interp.
ERD-dev [17]	Freebase	91	45	4
Y-ERD [32][a]	Freebase	2398	1256	9

[a] http://bit.ly/ictir2015-elq

7.3.4.4 Test Collections

There are two publicly available test collections for interpretation finding evaluation; see Table 7.8 for a summary.

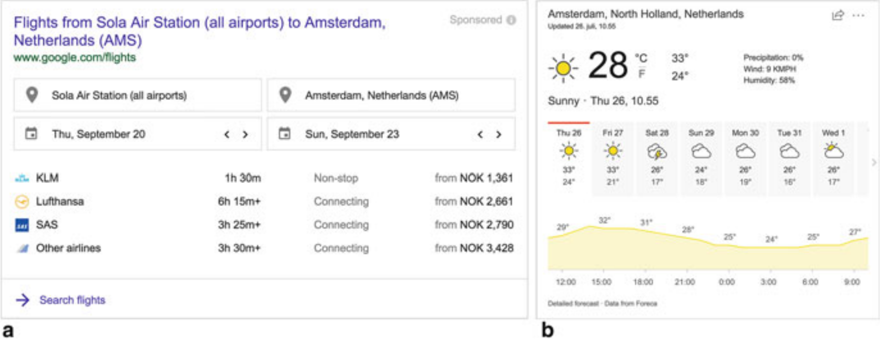
ERD-dev The "short text" track of the Entity Recognition and Disambiguation (ERD) Challenge [17] provided a live evaluation platform for the interpretation finding task. A development set of 91 queries is made publicly available; of these, 45 queries have non-empty entity annotations. The test set comprises 500 queries; because of the live evaluation, the annotations for these queries are not available for traditional offline evaluation (we refer to Sect. 5.8.2.4 for a discussion on the live evaluation platform). The gold standard annotations are created manually, in accordance with the following three rules [17]: (1) for each entity, the longest mention is used, (2) only proper noun entities are annotated, and (3) overlapping mentions are not allowed within a single interpretation.

Y-ERD Hasibi et al. [32] created a larger test set based on the YSQLE collection (cf. Sect. 7.3.3.4). Following a set of guidelines, based on and expanding upon those of the ERD Challenge, they grouped independent entity annotations into semantically compatible sets of entity linking decisions, i.e., interpretations. Queries are annotated on their own, regardless of search sessions.

One observation that can be made from Table 7.8 is that very few queries actually have multiple interpretations. This explains why systems that returned only a single interpretation could end up being the best contenders at the ERD Challenge [17]. It remains an open question whether this is a limitation of currently available test collections, or if it is worth expending algorithmic effort toward finding multiple interpretations.

7.4 Query Templates

A large fraction of queries follow certain patterns. For instance, when people search for jobs, a frequently used query pattern is "*jobs in* ⟨*location*⟩," where ⟨location⟩ is a variable that can be instantiated, e.g., by a city ("*jobs in seattle*"), region ("*jobs in silicon valley*"), or country ("*jobs in the UK*"). From these patterns, templates may be inferred, which may be used for interpreting queries. As defined by Bortnikov et al. [13], "a template is a sequence of terms that are either text tokens or variables that can be substituted from some dictionary or taxonomy." To be consistent with


Flights from Sola Air Station (all airports) to Amsterdam, Netherlands (AMS)
www.google.com/flights
Sponsored

Sola Air Station (all airports) | Amsterdam, Netherlands (AMS)
Thu, September 20 | Sun, September 23

KLM	1h 30m	Non-stop	from NOK 1,361
Lufthansa	6h 15m+	Connecting	from NOK 2,661
SAS	3h 25m+	Connecting	from NOK 2,790
Other airlines	3h 30m+	Connecting	from NOK 3,428

→ Search flights

Amsterdam, North Holland, Netherlands
Updated 26. juli, 10.55
28 °C F 33° 24°
Precipitation: 0%
Wind: 9 KMPH
Humidity: 58%
Sunny · Thu 26, 10.55

Thu 26	Fri 27	Sat 28	Sun 29	Mon 30	Tue 31	Wed 1
33°	33°	26°	24°	26°	26°	26°
24°	21°	17°	18°	19°	16°	17°

Detailed forecast · Data from Foreca

Fig. 7.3 (**a**) Flight search widget shown on Google in response to the query "*flights svg ams*." (**b**) Weather widget on Bing for the query "*weather amsterdam*"

our earlier terminology we will use the expression *token*, which can be either a term (word) or an *attribute* (variable). In our examples, we mark attribute tokens as $\langle \ldots \rangle$.

Templates provide structured interpretations of queries and have numerous advantages:

- Templates allow us not only to identify the target domain (vertical) of the query (such as flights or weather) but also to make parameterized requests to them, by mapping parts of the query to appropriate attributes of the given service. These services can then display various widgets or direct displays on the SERP, such as the ones shown in Fig. 7.3.
- Templates generalize well and can match queries that have not been observed in training data or search logs.
- Templates are very efficient in terms of online performance (only simple dictionary look-ups are required).

Such query templates may be crafted manually, e.g., by using regular expressions. Manual template building, however, has obvious limitations, due to the large variety of possible query formulations. A more scalable approach is to extract templates automatically from search logs. Based on Agarwal et al. [1], we formally introduce some concepts central to this problem in Sect. 7.4.1, followed by an explanation of methods that can be used for mining query templates from query logs in Sect. 7.4.2. Table 7.9 summarizes the notation used in this section.

7.4.1 Concepts and Definitions

Our objective is to direct queries to specific services or verticals, which will be referred to as *domains*. We begin by characterizing the schema of a given domain.

Definition 7.2 (Domain Schema) The schema of a given domain D is a pair $S_D = (\mathcal{A}, \mathcal{W})$, where $\mathcal{A} = \{a_1, \ldots, a_n\}$ is a set of *attributes* and $\mathcal{W} =$

Table 7.9 Notation used in Sect. 7.4

Symbol	Meaning
$\mathcal{A}$	Set of attributes
L	Search log ($L = (\mathcal{Q}, \mathcal{S}, \mathcal{C})$)
D	Domain
q	Query ($q \in \mathcal{Q}$)
$\mathcal{Q}$	Set of queries in the search log
$\mathcal{Q}_0$	Set of seed domain queries
$\mathcal{Q}_D$	Domain queries (queries relevant for D)
$\mathcal{Q}_s$	Set of click-through queries of site s
$\mathcal{Q}_u$	Set of queries instantiated by template u
s	Site ($s \in \mathcal{S}$)
$\mathcal{S}$	Set of sites
S_D	Domain schema ($S_D = (\mathcal{A}, \mathcal{W})$)
u	Query template ($u \in \mathcal{U}$)
$\mathcal{U}$	Template universe
$\mathcal{U}_q$	Set of templates generated by query q
$\mathcal{V}$	Vocabulary of terms
$\mathcal{W}$	Vocabulary of possible attribute values

$\{\mathcal{W}(a_1), \dots, \mathcal{W}(a_n)\}$ is the *vocabulary* of the possible instances (i.e., values) of each of the attributes.

For example, attributes in the *jobs* domain may include $\mathcal{A} = \{company, location, category\}$. The vocabulary of the *company* attribute includes names of all entities that appear as possible values for that attribute: $\mathcal{W}(company) = \{$"Apple", "Microsoft", "Audi",...$\}$. Some attributes, like *category*, may require their own domain-specific dictionary.

Definition 7.3 (Query Template) A query template u is a sequence of tokens $u = \langle u_1, \dots, u_n \rangle$, where each token u_i is either a term or an attribute: $u_i \in \mathcal{V} \cup \mathcal{A}$, where $\mathcal{V}$ is a vocabulary of terms and $\mathcal{A}$ is the set of possible attributes. We further require that at least one of the template tokens is an attribute: $\exists\ u_i \in \mathcal{A}$.

For example, "*jobs in* ⟨*location*⟩," consists of two terms and an attribute. This template can instantiate different queries. The inverse operation is template generation: Given a query, what templates can be generated from it?

Definition 7.4 (Template Instantiation and Generation) Given a query template $u = \langle u_1, \dots, u_n \rangle$ and a query $q = \langle q_1, \dots, q_m \rangle$, the template u *instantiates* q, or, equivalently, query q *generates* u, with respect to the domain schema S_D, if $n = m$ and for each token position $i \in [1, n]$,

- if token i in the template is an attribute, then query term q_i matches one of the possible instances of that attribute: $u_i \in \mathcal{A} \implies q_i \in \mathcal{W}(u_i)$,
- otherwise (if token i in the template is a term), the query and template tokens are equal: $u_i \notin \mathcal{A} \implies q_i = u_i \in \mathcal{V}$.

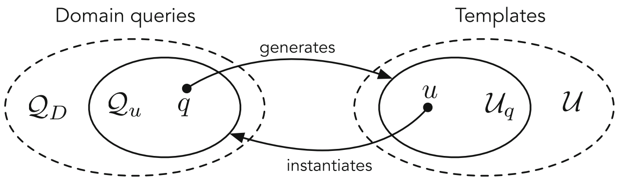


Fig. 7.4 Template generation and instantiation

The set of queries instantiated by template u is denoted by $\mathcal{Q}_u$. The templates generated from query q are denoted by $\mathcal{U}_q$. See Fig. 7.4 for a visual illustration.

Note that the vocabulary of a given attribute, according to the domain schema, may contain not only unigrams but n-grams as well (e.g., "new york"). When such an n-gram attribute instance is matched in the query, we treat it as a single query token.

To see an example of template generation, consider the query q ="*accounting jobs in new york*." According to our jobs domain schema, "*accounting*" is an instance of the *category* attribute and "*new york*" is an instance of the *location* attribute. This query can therefore generate the following three templates:

$$u_a = \text{"}\langle \textit{category} \rangle \textit{ jobs in new york,"}$$

$$u_b = \text{"}\textit{accounting jobs in } \langle \textit{location} \rangle \text{,"}$$

$$u_c = \text{"}\langle \textit{category} \rangle \textit{ jobs in } \langle \textit{location} \rangle \text{."}$$

Thus, $\mathcal{Q}_q = \{u_a, u_b, u_c\}$.

7.4.2 Template Discovery Methods

Template discovery is the task of finding "good" templates (according to some quality measure, such as precision, recall, or F1-score) from a search log L, for a given domain D. Approaching this task as a ranking problem, the output is a ranked list of templates, sorted by a template score.

We shall assume that the search log $L = (\mathcal{Q}, \mathcal{S}, \mathcal{C})$ provides a set $\mathcal{Q}$ of queries with click-throughs $\mathcal{C}$ to a set $\mathcal{S}$ of sites. Specifically, let $\mathcal{Q}_s$ denote the click-through queries of site $s \in \mathcal{S}$, i.e., $\mathcal{Q}_s = \{q : q \in \mathcal{Q}, \mathit{clicks}(q,s) > 0\}$, where $\mathit{clicks}(q,s)$ is the number of times a result originating from site s was clicked in response to q (over some time period). We shall further assume that we are given a seed set $\mathcal{Q}_0$ of domain queries, i.e., queries that resulted in clicks on target pages.

Since our interest is in discovering templates for a given domain D, we shall generate templates that match *domain queries*, i.e., queries that are relevant for that domain. We write $\mathcal{Q}_D$ to denote the set of domain queries ($\mathcal{Q}_D \subseteq \mathcal{Q}$). Let $\mathcal{U}$ denote the template universe, i.e., templates that can be generated by at least one $q \in \mathcal{Q}_D$. Formally: $\mathcal{U} = \{u : \exists\ q \in \mathcal{Q}_D, u \in \mathcal{Q}_q\}$.

Algorithm 7.2: Classify & match [1]

```
Input: $L = (\mathcal{Q},\mathcal{S},\mathcal{C})$, $S_D$, $\mathcal{Q}_0$, $\tau$
Output: Templates $u$, ranked by $score(u)$
classify $\mathcal{Q}$ to $\mathcal{Q}_D$, trained on $\mathcal{Q}_0$, thresholded by $\tau$
$\mathcal{U} \leftarrow \{u : u \in \mathcal{Q}_q, \forall q \in \mathcal{Q}_D\}$
foreach $u \in \mathcal{U}$ do
    compute $score(u)$
end
return $\mathcal{U}$ sorted by $score(u)$                /* $\mathrm{P}_u$, $\mathrm{R}_u$, or $\mathrm{F1}_u$ */
```

7.4.2.1 Classify&Match

Agarwal et al. [1] introduce a natural baseline algorithm, called *Classify&Match*, which operates in two stages. First, domain queries $\mathcal{Q}_D$ are separated from all queries $\mathcal{Q}$ in the query log using automatic classification. Specifically, a query classifier is trained on the seed domain queries $\mathcal{Q}_0$ using the method from [45], with a threshold τ applied on the results. In the second stage, for each template $u \in \mathcal{U}$ is scored against the (estimated) set of domain queries $\mathcal{Q}_D$, using precision, recall, or F-measure as *score*(u). See Algorithm 7.2.

Quality Measures To be able to measure the quality of a given template u, we establish *precision* and *recall*, analogously to the standard retrieval measures. The "target" set is the collection of domain queries, $\mathcal{Q}_D$. We shall assume that this set is clearly identified (e.g., by taking all queries that resulted in clicks on a set of target pages). The "matched" set is $\mathcal{Q}_u$, i.e., queries that are instantiated by the template. The *precision* of template u is the fraction of $\mathcal{Q}_u$ that falls within $\mathcal{Q}_D$:

$$\mathrm{P}_u = \frac{\mathcal{Q}_D \cap \mathcal{Q}_u}{\mathcal{Q}_u} . \tag{7.10}$$

The *recall* of template u is the fraction of $\mathcal{Q}_D$ that is covered by $\mathcal{Q}_u$:

$$\mathrm{R}_u = \frac{\mathcal{Q}_D \cap \mathcal{Q}_u}{\mathcal{Q}_D} . \tag{7.11}$$

Ultimately, the overall quality of a template needs to be measured by a combination of precision and recall, e.g., by using the F-measure ($\mathrm{F1}_u$, cf. Eq. (5.10)).

7.4.2.2 QueST

While simple and intuitive, the above naïve baseline approach suffers from two shortcomings. First, queries are ambiguous and click-throughs are noisy by nature; deterministically separating domain and non-domain queries is problematic. Instead, probabilistic modeling is needed that can encapsulate the fuzziness

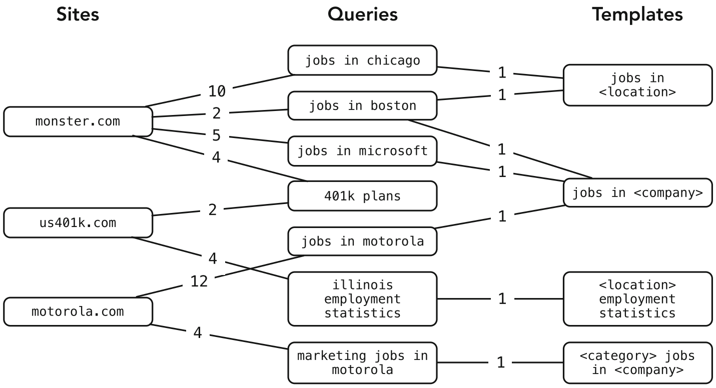


Fig. 7.5 QST graph based on a toy-sized search log. Example is taken from [1]

of domain relevance. Second, search logs are not only noisy but also sparse. By separating template mining into two separate stages, an important (indirect) connection between sites and templates is lost. To overcome these issues, Agarwal et al. [1] present an iterative inferencing framework, called *QueST*, over a tripartite graph of queries, sites, and templates. Precision and recall is defined for each type of node. The process of template discovery can then be seen as propagating precision and recall across the query-site-template graph using random walks. We shall elaborate on the details below.

QST Graph Queries, sites, and templates are represented as a tripartite graph (*QST graph*) $G_{QST} = (V, E)$, where the vertices are $V = \mathcal{Q} \cup \mathcal{S} \cup \mathcal{U}$, and there are two types of edges E, with $w(x, y)$ denoting the edge weight between nodes x and y:

- *Query-site edges*: For each query q that clicks to site s, there is an edge with the click-through frequency as weight: $\forall s, \forall q \in \mathcal{Q}_s : w(q,s) = clicks(q,s)$.
- *Query-template edges*: For each query q that instantiates template u, there is an edge in between with weight 1: $\forall u \in \mathcal{U}, \forall q \in \mathcal{Q}_u : w(q,u) = 1$.

Figure 7.5 displays the QST graph for a toy-sized example.

Probabilistic Modeling In practice, due to query ambiguity and noisy click-throughs, the crisp separation of queries into target domains is rather problematic (e.g., the query "*microsoft*" might be in *job* or in *product*). Therefore, we shall generalize the deterministic notions of precision and recall from the previous subsection to probabilistic measures. Let $match(q,x)$ denote the event that query q and x are semantically matching, where x may be a template u, a site s, or a domain D. Further, we let $P(match(q,x))$ be the probability of *semantic relevance* between q and x. Precision in Eq. (7.10) can then be rewritten in the "match" notation as the

Algorithm 7.3: QueST [1]

```
Input: $L = (\mathcal{Q},\mathcal{S},\mathcal{C})$, $S_D$, $\mathcal{Q}_0$, $P_0$, $R_0$
Output: Templates $u$, ranked by $score(u)$
$\mathcal{U} \leftarrow \{u : u \in \mathcal{Q}_q, \forall q \in \mathcal{Q}_D\}$
construct $G_{QST}$ given $\mathcal{Q}$, $S, \mathcal{U}, \mathcal{C}$
$\mathrm{R}_u, \mathrm{R}_q, \mathrm{R}_s \leftarrow$ QuestR on $G_{QST}$ with $R_0$          /* inference recall */
$\mathrm{P}_u, \mathrm{P}_q, \mathrm{P}_s \leftarrow$ QuestP on $G_{QST}$ with $P_0$       /* inference precision */
return $\mathcal{U}$ sorted by $score(u)$                             /* $\mathrm{P}_u$, $\mathrm{R}_u$, or $\mathrm{F1}_u$ */
```

following conditional probability:

$$\mathrm{P}_u = \frac{P\left(match(q,D), match(q,u)\right)}{P\left(match(q,u)\right)} = P\left(match(q,D)|match(q,u)\right) \ .$$

Similarly, recall in Eq. (7.11) is rewritten as:

$$\mathrm{R}_u = \frac{P\left(match(q,D), match(q,u)\right)}{P\left(match(q,D)\right)} = P\left(match(q,u)|match(q,D)\right) \ .$$

Next, we extend the notions of precision and recall to sites and queries. This is needed for being able to perform integrated inferencing on queries, sites, and templates. The precision and recall of a site is modeled analogously to that of templates. That is, precision measures how likely queries match domain D given that they match site s; for recall, it is the other way around. Formally:

$$\mathrm{P}_s = P\left(match(q,D)|match(q,s)\right) \ ,$$
$$\mathrm{R}_s = P\left(match(q,s)|match(q,D)\right) \ .$$

The precision of a query q is simply its probability of matching the domain. The recall of q is the fraction of domain queries that are actually q.

$$\mathrm{P}_q = P\left(match(q,D)\right) \ ,$$
$$\mathrm{R}_q = P\left(q' = q|match(q',D)\right) \ .$$

Inference Framework Let us remember our ultimate goal, which is to estimate precision P_u and recall R_u for each template $u \in \mathcal{U}$. As part of the integrated inferencing process, we will also estimate precision and recall for other types of vertices in the QST graph, i.e., for queries (P_q, R_q, $\forall q \in \mathcal{Q}$) and for sites ($\mathrm{P}_s$, R_s, $\forall s \in \mathcal{S}$). The QueST algorithm, shown in Algorithm 7.3, infers precision and recall for each vertex, then ranks templates by precision, recall, or the combined F-measure. This process of propagating precision and recall may be interpreted as random walks in opposite directions on the QST-graph. See Fig. 7.6 for an illustration.

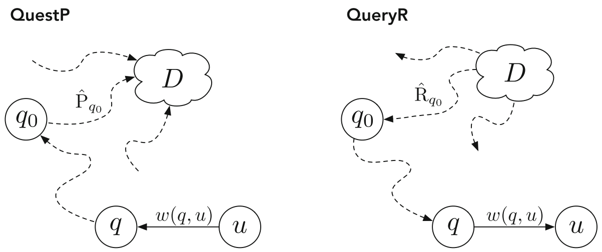


Fig. 7.6 Inferencing precision and recall. QuestP is for precision and propagates backward. QuestR is for recall and propagates forward. Dashed arrows signify random walks

Key to this semi-supervised learning process is to have a small set of seed queries $\mathcal{Q}_0$. Each of these queries $q_0 \in \mathcal{Q}_0$ is labeled with precision $\hat{P}_{q_0}$, indicating how likely it is to fall within the target domain D. As it is infeasible for users to provide seed recall, it is estimated as precision normalized across all seed queries:

$$\hat{R}_{q_0} = \frac{\hat{P}_{q_0}}{\sum_{q' \in \mathcal{Q}_0} \hat{P}_{q'}} .$$

These initial precision and recall values get propagated in the QST-graph through a set of inferencing equations, which we shall present below. For the derivation of these equations, we refer to [1]. The algorithm is reported to converge in four to five iterations [1].

QuestP Estimating P_u may be thought of as the probability of reaching the target domain D in a (backward) random walk starting from u; see Fig. 7.6 (left). We take as input the initial precision estimates for the seed queries, $\hat{P}_{q_0}, q_0 \in \mathcal{Q}_0$. Based on this seed knowledge, the precision P_v of every vertex $v \in V_G$ is determined by propagating precision from its neighboring vertices. The following three equations specify inference for template vertices ($\mathcal{Q} \rightarrow \mathcal{U}$), site vertices ($\mathcal{Q} \rightarrow \mathcal{S}$), and query vertices ($\mathcal{U} \rightarrow \mathcal{Q} \wedge \mathcal{S} \rightarrow \mathcal{Q}$), respectively.

$$P_u = \sum_{q \in \mathcal{Q}_u} \frac{w(q,u)}{\sum_{q'} w(q',u)} P_q ,$$

$$P_s = \sum_{q \in \mathcal{Q}_s} \frac{w(q,s)}{\sum_{q'} w(q',s)} P_q ,$$

$$P_q = \begin{cases} \hat{P}_q, & \text{if } q \in \mathcal{Q}_0 \\ \alpha \sum_{u \in \mathcal{U}} \frac{w(q,u)}{\sum_{u'} w(q,u')} P_u + (1-\alpha) \sum_{s \in \mathcal{S}} \frac{w(q,s)}{\sum_{s'} w(q,s')} P_s, & \text{otherwise} . \end{cases}$$

The parameter α controls the relative importance of templates and sites in inferring the precision of queries ($\alpha = 0.5$ in [1]). Note that for seed queries $q \in \mathcal{Q}_0$, the initial precision $\hat{\mathrm{P}}_q$ is taken as "ground truth" and will not change.

QuestR The inference of probabilistic recall follows a very similar process. We may think of R_u as the probability of arriving at node u in a (forward) random walk starting from given seeds $q_0 \in \mathcal{Q}_0$ originating from a hidden domain D; see Fig. 7.6 (right). Recall is distributed to neighboring nodes using the following inference equations:

$$\mathrm{R}_u = \sum_{q \in \mathcal{Q}_u} \frac{w(q,u)}{\sum_{u'} w(q,u')} \mathrm{R}_q \ ,$$

$$\mathrm{R}_s = \sum_{q \in \mathcal{Q}_s} \frac{w(q,s)}{\sum_{s'} w(q,s')} \mathrm{R}_q \ ,$$

$$\mathrm{R}_q = \beta_1 \hat{\mathrm{R}}_q + \beta_2 \sum_{u \in \mathcal{U}} \frac{w(q,u)}{\sum_{q'} w(q',u)} \mathrm{R}_u + (1 - \beta_1 - \beta_2) \sum_{s \in \mathcal{S}} \frac{w(q,s)}{\sum_{q'} w(q',s)} \mathrm{R}_s \ .$$

Notice that, unlike for precision, the recall of seed queries will also get re-estimated. Parameters β_1 and β_2 specify the relative importance of the different sources, $\hat{\mathrm{R}}_q$, R_u, and R_s, when estimating R_q (with $\beta_1 = 0.1$ and $\beta_2 = 0.45$ in [1]).

7.5 Summary

The question driving this chapter has been how to obtain a semantically enriched representation of the user's information need from a keyword query. We have looked at three specific forms of enrichment, all of which are semantic annotations performed with the help of a knowledge repository. First, we have discussed how to annotate queries with target types from a type taxonomy. Second, we have performed entity linking on queries in a number of flavors, from merely recognizing entity mentions to forming coherent interpretation sets. Third, we have generated query templates, which provide structured interpretations of queries by mapping parts of the query to the specific entity attributes. These structured interpretations can then be used to make parameterized requests to particular search services (e.g., verticals). Having thus enriched the user's query with inferred information about their underlying information need, the response to that query can be made more effectively.

7.6 Further Reading

There is a rich and diverse body of research on understanding search queries that was not possible to compress into this chapter. One important practical issue that we have not discussed is *spell checking*. According to Cucerzan and Brill [20], roughly 10–15% of Web search queries contain spelling errors. It is therefore strongly recommended to perform spelling correction before commencing any of the query analysis steps. For example, Blanco et al. [12] report on a 3% improvement in entity linking performance, ascribed to spelling correction. *Query refinement* (also known as *query modification*) refers to the automated process of changing ill-formed queries submitted by users before scoring results. It includes tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion [29].

Han et al. [31] address the limitations of machine-based methods for query interpretation by utilizing crowdsourcing in a hybrid crowd-machine framework.

The methods we have presented in this chapter do not make use of the searcher's context, such as age, gender, topic, or location. To a large extent, it is because this type of information is unavailable in public test collections. Contemporary web search engines leverage contextual information by serving personalized search results according to the user's profile [69, 72]. This, however, is typically done by designing specific ranking features [2]. For example, Murnane et al. [57] utilize *personal context* for named entity disambiguation by modeling user interests with respect to a personal knowledge context (using Wikipedia).

References

1. Agarwal, G., Kabra, G., Chang, K.C.C.: Towards rich query interpretation: Walking back and forth for mining query templates. In: Proceedings of the 19th international conference on World Wide Web, WWW '10, pp. 1–10. ACM (2010). doi: 10.1145/1772690.1772692
2. Agichtein, E., Brill, E., Dumais, S., Ragno, R.: Learning user interaction models for predicting web search result preferences. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, pp. 3–10. ACM (2006). doi: 10.1145/1148170.1148175
3. Arguello, J., Diaz, F., Callan, J., Crespo, J.F.: Sources of evidence for vertical selection. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, pp. 315–322. ACM (2009). doi: 10.1145/1571941.1571997
4. Ashkan, A., Clarke, C.L.A.: Characterizing commercial intent. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, pp. 67–76. ACM (2009). doi: 10.1145/1645953.1645965
5. Balog, K., Neumayer, R.: Hierarchical target type identification for entity-oriented queries. In: Proceedings of the 21st ACM international conference on Information and knowledge management, CIKM '12, pp. 2391–2394. ACM (2012). doi: 10.1145/2396761.2398648

6. Barr, C., Jones, R., Regelson, M.: The linguistic structure of English web-search queries. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08, pp. 1021–1030 (2008)
7. Bendersky, M., Croft, W.B.: Discovering key concepts in verbose queries. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pp. 491–498. ACM (2008). doi: 10.1145/1390334.1390419
8. Bendersky, M., Croft, W.B., Smith, D.A.: Structural annotation of search queries using pseudo-relevance feedback. In: Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10, pp. 1537–1540. ACM (2010). doi: 10.1145/1871437.1871666
9. Bendersky, M., Croft, W.B., Smith, D.A.: Joint annotation of search queries. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, pp. 102–111. Association for Computational Linguistics (2011)
10. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: Empirical Methods in Natural Language Processing, EMNLP '13, pp. 1533–1544. Association for Computational Linguistics (2013)
11. Bergsma, S., Wang, Q.I.: Learning noun phrase query segmentation. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '07, pp. 819–826. Association for Computational Linguistics (2007)
12. Blanco, R., Ottaviano, G., Meij, E.: Fast and space-efficient entity linking for queries. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15, pp. 179–188. ACM (2015). doi: 10.1145/2684822.2685317
13. Bortnikov, E., Donmez, P., Kagian, A., Lempel, R.: Modeling transactional queries via templates. In: Proceedings of the 34th European Conference on Advances in Information Retrieval, ECIR '12, pp. 13–24. Springer (2012). doi: 10.1007/978-3-642-28997-2_2
14. Brants, T., Franz, A.: Web 1T 5-gram Version 1 LDC2006T13 (2006)
15. Brenes, D.J., Gayo-Avello, D., Garcia, R.: On the fly query entity decomposition using snippets. CoRR **abs/1005.5** (2010)
16. Broder, A.: A taxonomy of web search. SIGIR Forum **36**(2), 3–10 (2002)
17. Carmel, D., Chang, M.W., Gabrilovich, E., Hsu, B.J.P., Wang, K.: ERD'14: Entity recognition and disambiguation challenge. SIGIR Forum **48**(2), 63–77 (2014). doi: 10.1145/2701583.2701591
18. Cornolti, M., Ferragina, P., Ciaramita, M.: A framework for benchmarking entity-annotation systems. In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, pp. 249–260 (2013). doi: 10.1145/2488388.2488411
19. Cornolti, M., Ferragina, P., Ciaramita, M., Rüd, S., Schütze, H.: A piggyback system for joint entity mention detection and linking in web queries. In: Proceedings of the 25th International Conference on World Wide Web, WWW '16, pp. 567–578. International World Wide Web Conferences Steering Committee (2016). doi: 10.1145/2872427.2883061
20. Cucerzan, S., Brill, E.: Spelling correction as an iterative process that exploits the collective knowledge of web users. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP '04 (2004)
21. Dai, H.K., Zhao, L., Nie, Z., Wen, J.R., Wang, L., Li, Y.: Detecting online commercial intention (OCI). In: Proceedings of the 15th International Conference on World Wide Web, WWW '06, pp. 829–837. ACM (2006). doi: 10.1145/1135777.1135902
22. Diaz, F.: Integration of news content into web results. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM '09, pp. 182–191. ACM (2009). doi: 10.1145/1498759.1498825

23. Ferragina, P., Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities). In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 1625–1628. ACM (2010). doi: 10.1145/1871437.1871689
24. Gabrilovich, E., Broder, A., Fontoura, M., Joshi, A., Josifovski, V., Riedel, L., Zhang, T.: Classifying search queries using the web as a source of knowledge. ACM Trans. Web **3**(2), 5:1–5:28 (2009). doi: 10.1145/1513876.1513877
25. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'07, pp. 1606–1611. Morgan Kaufmann Publishers Inc. (2007)
26. Ganti, V., He, Y., Xin, D.: Keyword++: A framework to improve keyword search over entity databases. Proc. VLDB Endow. **3**(1–2), 711–722 (2010). doi: 10.14778/1920841.1920932
27. Garigliotti, D., Hasibi, F., Balog, K.: Target type identification for entity-bearing queries. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17. ACM (2017). doi: 10.1145/3077136.3080659
28. Guo, J., Xu, G., Cheng, X., Li, H.: Named entity recognition in query. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, pp. 267–274. ACM (2009). doi: 10.1145/1571941.1571989
29. Guo, J., Xu, G., Li, H., Cheng, X.: A unified and discriminative model for query refinement. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pp. 379–386 (2008). doi: 10.1145/1390334.1390400
30. Hagen, M., Potthast, M., Stein, B., Bräutigam, C.: Query segmentation revisited. In: Proceedings of the 20th International Conference on World Wide Web, WWW '11, pp. 97–106 (2011). doi: 10.1145/1963405.1963423
31. Han, J., Fan, J., Zhou, L.: Crowdsourcing-assisted query structure interpretation. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, pp. 2092–2098. AAAI Press (2013)
32. Hasibi, F., Balog, K., Bratsberg, S.E.: Entity linking in queries: Tasks and evaluation. In: Proceedings of the 2015 International Conference on The Theory of Information Retrieval, ICTIR '15, pp. 171–180. ACM (2015). doi: 10.1145/2808194.2809473
33. Hasibi, F., Balog, K., Bratsberg, S.E.: Exploiting entity linking in queries for entity retrieval. In: Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR '16, pp. 209–218. ACM (2016). doi: 10.1145/2970398.2970406
34. Hasibi, F., Balog, K., Bratsberg, S.E.: Entity linking in queries: Efficiency vs. effectiveness. In: Proceedings of the 39th European conference on Advances in Information Retrieval, ECIR '17, pp. 40–53. Springer (2017). doi: 10.1007/978-3-319-56608-5_4
35. Hu, J., Wang, G., Lochovsky, F., Sun, J.t., Chen, Z.: Understanding user's query intent with Wikipedia. In: Proceedings of the 18th International Conference on World Wide Web, WWW '09, pp. 471–480. ACM (2009). doi: 10.1145/1526709.1526773
36. Huang, J., Gao, J., Miao, J., Li, X., Wang, K., Behr, F., Giles, C.L.: Exploring web scale language models for search query processing. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 451–460. ACM (2010). doi: 10.1145/1772690.1772737
37. Huston, S., Croft, W.B.: Evaluating verbose query processing techniques. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, pp. 291–298. ACM (2010). doi: 10.1145/1835449.1835499
38. Jansen, B.J., Booth, D.: Classifying web queries by topic and user intent. In: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems, CHI EA '10, pp. 4285–4290. ACM (2010)

39. Jansen, B.J., Booth, D.L., Spink, A.: Determining the informational, navigational, and transactional intent of web queries. Inf. Process. Manage. **44**(3), 1251–1266 (2008). doi: 10.1016/j.ipm.2007.07.015
40. Jones, R., Rey, B., Madani, O., Greiner, W.: Generating query substitutions. In: Proceedings of the 15th International Conference on World Wide Web, WWW '06, pp. 387–396. ACM (2006). doi: 10.1145/1135777.1135835
41. Kaptein, R., Serdyukov, P., De Vries, A., Kamps, J.: Entity ranking using Wikipedia as a pivot. In: Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10, pp. 69–78. ACM (2010). doi: 10.1145/1871437.1871451
42. Kraaij, W., Spitters, M.: Language models for topic tracking. In: Croft, W., Lafferty, J. (eds.) Language Modeling for Information Retrieval, *The Springer International Series on Information Retrieval*, vol. 13, pp. 95–123. Springer (2003)
43. Lee, U., Liu, Z., Cho, J.: Automatic identification of user goals in web search. In: Proceedings of the 14th International Conference on World Wide Web, WWW '05, pp. 391–400. ACM (2005). doi: 10.1145/1060745.1060804
44. Li, X.: Understanding the semantic structure of noun phrase queries. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pp. 1337–1345. Association for Computational Linguistics (2010)
45. Li, X., Wang, Y.Y., Acero, A.: Learning query intent from regularized click graphs. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08, pp. 339–346. ACM (2008). doi: 10.1145/1390334.1390393
46. Li, X., Wang, Y.Y., Acero, A.: Extracting structured information from user queries with semi-supervised conditional random fields. In: Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, pp. 572–579. ACM (2009). doi: 10.1145/1571941.1572039
47. Li, Y., Hsu, B.J.P., Zhai, C., Wang, K.: Unsupervised query segmentation using clickthrough for information retrieval. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, pp. 285–294. ACM (2011). doi: 10.1145/2009916.2009957
48. Li, Y., Zheng, Z., Dai, H.K.: KDD CUP-2005 report: facing a great challenge. SIGKDD Explor. Newsl. **7**(2), 91–99 (2005)
49. Lin, T., Pantel, P., Gamon, M., Kannan, A., Fuxman, A.: Active objects. In: Proceedings of the 21st international conference on World Wide Web, WWW '12, pp. 589–598. ACM (2012). doi: 10.1145/2187836.2187916
50. Liu, X., Zhang, S., Wei, F., Zhou, M.: Recognizing named entities in tweets. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pp. 359–367. Association for Computational Linguistics (2011)
51. Manshadi, M., Li, X.: Semantic tagging of web search queries. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL '09, pp. 861–869. Association for Computational Linguistics (2009)
52. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: The Penn Treebank. Comput. Linguist. **19**(2), 313–330 (1993)
53. Meij, E., Bron, M., Hollink, L., Huurnink, B., de Rijke, M.: Mapping queries to the linking open data cloud: A case study using DBpedia. Web Semant. **9**(4), 418–433 (2011)
54. Meij, E., Weerkamp, W., De Rijke, M.: Adding semantics to microblog posts. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, pp. 563–572. ACM (2012). doi: 10.1145/2124295.2124364

55. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13, pp. 3111–3119. Curran Associates Inc. (2013)
56. Mishra, N., Saha Roy, R., Ganguly, N., Laxman, S., Choudhury, M.: Unsupervised query segmentation using only query logs. In: Proceedings of the 20th International Conference Companion on World Wide Web, WWW '11, pp. 91–92. ACM (2011). doi: 10.1145/1963192.1963239
57. Murnane, E.L., Haslhofer, B., Lagoze, C.: RESLVE: leveraging user interest to improve entity disambiguation on short text. In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13 Companion, pp. 81–82. ACM (2013). doi: 10.1145/2487788.2487823
58. Pantel, P., Lin, T., Gamon, M.: Mining entity types from query logs via user intent modeling. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pp. 563–571. Association for Computational Linguistics (2012)
59. Paparizos, S., Ntoulas, A., Shafer, J., Agrawal, R.: Answering web queries using structured data sources. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, SIGMOD '09, pp. 1127–1130. ACM (2009). doi: 10.1145/1559845.1560000
60. Piccinno, F., Ferragina, P.: From TagME to WAT: A new entity annotator. In: Proceedings of the First International Workshop on Entity Recognition & Disambiguation, ERD '14, pp. 55–62. ACM (2014). doi: 10.1145/2633211.2634350
61. Pound, J., Hudek, A.K., Ilyas, I.F., Weddell, G.: Interpreting keyword queries over web knowledge bases. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, pp. 305–314. ACM (2012). doi: 10.1145/2396761.2396803
62. Risvik, K.M., Mikolajewski, T., Boros, P.: Query segmentation for web search. In: Proceedings of the 12th International Conference on World Wide Web, WWW '03 (2003)
63. Rose, D.E., Levinson, D.: Understanding user goals in web search. In: Proceedings of the 13th International Conference on World Wide Web, WWW '04, pp. 13–19 (2004). doi: 10.1145/988672.988675
64. Rüd, S., Ciaramita, M., Müller, J., Schütze, H.: Piggyback: Using search engines for robust cross-domain named entity recognition. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 965–975 (2011)
65. Saha Roy, R., Ganguly, N., Choudhury, M., Laxman, S.: An IR-based evaluation framework for web search query segmentation. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, pp. 881–890. ACM (2012). doi: 10.1145/2348283.2348401
66. Sarkas, N., Paparizos, S., Tsaparas, P.: Structured annotations of web queries. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pp. 771–782 (2010). doi: 10.1145/1807167.1807251
67. Sawant, U., Chakrabarti, S.: Learning joint query interpretation and response ranking. In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, pp. 1099–1109 (2013). doi: 10.1145/2488388.2488484
68. Shen, D., Sun, J.T., Yang, Q., Chen, Z.: Building bridges for web query classification. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, pp. 131–138. ACM (2006). doi: 10.1145/1148170.1148196
69. Speretta, M., Gauch, S.: Personalized search based on user search histories. In: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI '05, pp. 622–628. IEEE Computer Society (2005). doi: 10.1109/WI.2005.114
70. Srba, I., Bielikova, M.: A comprehensive survey and classification of approaches for community question answering. ACM Trans. Web **10**(3), 18:1–18:63 (2016). doi: 10.1145/2934687

71. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and Wikipedia. In: Proceedings of the 17th international conference on World Wide Web, WWW '08, pp. 347–356. ACM (2008). doi: 10.1145/1367497.1367545
72. Teevan, J., Dumais, S.T., Horvitz, E.: Personalizing search via automated analysis of interests and activities. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '05, pp. 449–456. ACM (2005). doi: 10.1145/1076034.1076111
73. Tonon, A., Catasta, M., Prokofyev, R., Demartini, G., Aberer, K., Cudré-Mauroux, P.: Contextualized ranking of entity types based on knowledge graphs. Web Semant. **37–38**, 170–183 (2016). doi: 10.1016/j.websem.2015.12.005
74. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, NAACL '03, pp. 173–180. Association for Computational Linguistics (2003). doi: 10.3115/1073445.1073478
75. Tsur, G., Pinter, Y., Szpektor, I., Carmel, D.: Identifying web queries with question intent. In: Proceedings of the 25th International Conference on World Wide Web, WWW '16, pp. 783–793. International World Wide Web Conferences Steering Committee (2016). doi: 10.1145/2872427.2883058
76. Ullegaddi, P.V., Varma, V.: Learning to rank categories for web queries. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, pp. 2065–2068. ACM (2011). doi: 10.1145/2063576.2063891
77. Vallet, D., Zaragoza, H.: Inferring the most important types of a query: A semantic approach. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08, pp. 857–858. ACM (2008). doi: 10.1145/1390334.1390541
78. Voskarides, N., Meij, E., Tsagkias, M., de Rijke, M., Weerkamp, W.: Learning to explain entity relationships in knowledge graphs. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 564–574. Association for Computational Linguistics (2015)
79. Wei, X., Peng, F., Dumoulin, B.: Analyzing web text association to disambiguate abbreviation in queries. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pp. 751–752 (2008). doi: 10.1145/1390334.1390485
80. Yih, S.W.t., Chang, M.W., He, X., Gao, J.: Semantic parsing via staged query graph generation: Question answering with knowledge base. In: Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP. ACL - Association for Computational Linguistics (2015)
81. Yin, X., Shah, S.: Building taxonomy of web search intents for name entity queries. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 1001–1010. ACM (2010). doi: 10.1145/1772690.1772792

82. Zhang, S., Balog, K.: Design patterns for fusion-based object retrieval. In: Proceedings of the 39th European conference on Advances in Information Retrieval, ECIR '17. Springer (2017). doi: 10.1007/978-3-319-56608-5_66
83. Zhou, K., Cummins, R., Halvey, M., Lalmas, M., Jose, J.M.: Assessing and predicting vertical intent for web queries. In: Proceedings of the 34th European conference on Advances in Information Retrieval, ECIR'12, pp. 499–502. Springer (2012). doi: 10.1007/978-3-642-28997-2_50