Chapter 6
Populating Knowledge Bases

A knowledge base (KB) contains information about entities and their properties (types, attributes, and relationships). In the case of large KBs, the number of entities is in the millions and the number of facts is in the billions. Commonly, this information is represented in the form of (sets of) subject-predicate-object (SPO) triples, according to the RDF data model (cf. Sect. 2.3). KBs are utilized in a broad variety of information access tasks, including entity retrieval (Chaps. 3 and 4), entity linking (Chap. 5), and semantic search (Chaps. 7–9). Two main challenges associated with knowledge bases are that (1) they are inherently incomplete (and will always remain so, despite any effort), and (2) they need constant updating over time as new facts and discoveries may turn the content outdated, inaccurate, or incomplete.

Knowledge base population (KBP) refers to the task of discovering new facts about entities from a large text corpus, and augmenting a KB with these facts. KBP is a broad problem area, with solutions ranging from fully automated systems to setups with a human content editor in the loop, who is in charge of any changes made to the KB. Our interest in this chapter will be on the latter type of systems, which "merely" provide assistance with the labor-intensive manual process.

Specifically, we will focus on a streaming setting, with the goal to discover and extract new information about entities as it becomes available. This information can then be used to augment an existing KB. This flavor of KBP has been termed *knowledge base acceleration* (KBA) [27]. KBA systems "seek to help humans expand knowledge bases [...] by automatically recommending edits based on incoming content streams" [7].

There is a practical real-world motivation behind this particular problem formulation. Many large knowledge repositories are maintained by a small workforce of content editors. Due to the scarcity of human resources, "most entity profiles lag far behind current events" [26]. For example, Frank et al. [27] show that the median time elapsed between the publication dates of news articles that are cited in Wikipedia articles of living people and the dates of the corresponding edits to

K. Balog, *Entity-Oriented Search*, The Information Retrieval Series 39,
https://doi.org/10.1007/978-3-319-93935-3_6

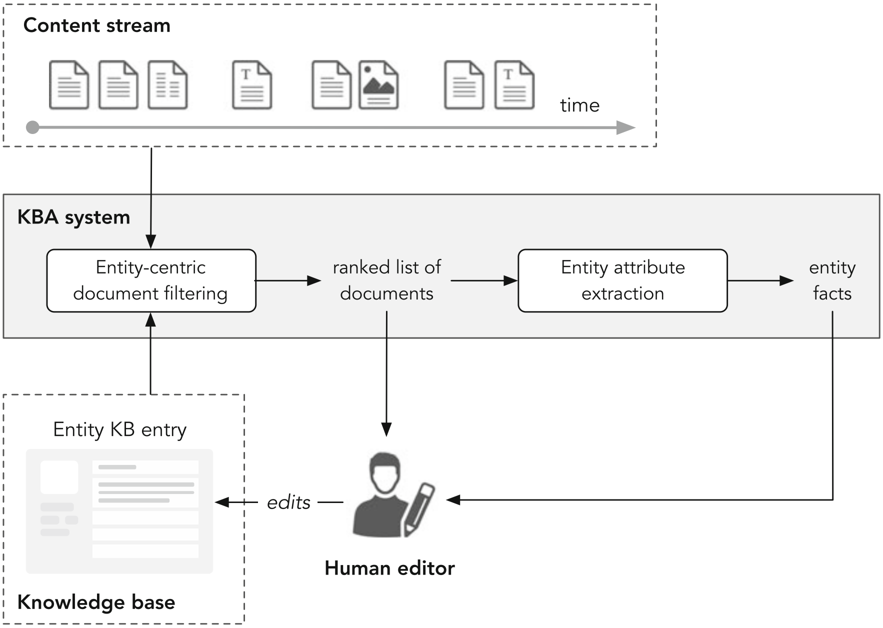


Fig. 6.1 Overview of knowledge base acceleration

Wikipedia is over a year. Thus, we wish to help editors stay on top of changes by automatically identifying content (news articles, blog posts, etc.) that may imply modifications to the KB entries of a certain set of entities of interest (i.e., entities that a given editor is responsible for).

Accelerating the knowledge base entry of a given target entity entails two main steps. The first is about identifying relevant documents that contain new facts about that entity. In a streaming setting, the processing of new documents is performed in batches, as they become available. (We note that the same techniques may be applied in a static, i.e., non-streaming, setting as well.) We address this subtask in Sect. 6.2. The second step concerns the extraction of facts from those documents. This is restricted to a predefined set of entity properties, thus may be seen as the problem of filling slots in the entity's knowledge base entry. We focus on this subtask in Sect. 6.3. Depending on the degree of automation, the human editor may use only the first step and process the documents manually that were filtered by the KBA system. Alternatively, she may operate with the extracted facts and—after reviewing them—apply the recommended changes to the KB by a click of a mouse. Figure 6.1 illustrates the process.

Before discussing the two main components of KBA systems, we shall begin in Sect. 6.1 with giving a brief overview of the broader problem area of extracting structured information from unstructured data.

6.1 Harvesting Knowledge from Text

The broad goal of information extraction (IE) is to automatically extract structured (factual) information from unstructured or semi-structured sources. While IE has its roots in natural language processing, the topic of extracting structured data now engages many different research communities, including those of information retrieval and databases. There is a wide spectrum of potential input sources, including web pages, news articles, social media and user-generated content (blogs, tweets, reviews), corporate and medical reports, web tables, and search queries, just to name a few. The spectrum of methods and techniques is similarly broad, ranging from hand-written regular expressions to probabilistic graphical models. Finally, the type of structure extracted extends from atomic units, such as entities, types, attributes, and relationships to higher-order structures such as lists, tables, and ontologies. For a general overview on IE, the reader is referred to [38, 68].

The area of large-scale knowledge acquisition, i.e., extracting information related to entities from large document corpora with the aim to build or extend a knowledge base, has become a major research avenue over the past decade [14, 87]. There are many dimensions along which approaches may be categorized. We highlight two particular dimensions.

Closed vs. Open Information Extraction Traditional information extraction is *closed*, in a sense that we wish to populate an existing knowledge base with additional facts about entities, where all entities, types, and relationships already have canonicalized (unique) identifiers assigned to them in the KB. That is, we are not aiming to discover new entities, entity types, or relationship types, but only instantiations of those in the form of new facts. This paradigm is also referred to as *ontology-based IE* [71]. *Open* information extraction, on the other hand, aims to discover new entities and new relationship types and extract all instances of those. In other words, there is no pre-specified vocabulary, the elements of facts are represented as textual phrases, which—at this stage—are not linked to a knowledge base. For example, given the sentence "Born in Honolulu, Hawaii, Obama is a US Citizen," a closed IE system would represent the contained information (using the entities and predicates from the DBpedia knowledge base) as:[1]

<dbr:Barack_Obama>	<dbo:nationality>	<dbr:United_States>
<dbr:Barack_Obama>	<dbo:birthPlace>	<dbr:Honolulu>

[1] We know (can infer) from the knowledge base that Honolulu is the state capital of Hawaii, therefore it is not needed to add a third triple asserting the relationship between Obama and Hawaii.

Instead, an open IE system might extract the following two triples:

```
(Obama; is; US citizen)
(Obama; born in; Honolulu, Hawaii)
```

Non-targeted vs. Targeted Extraction *Non-targeted* (or unfocused) extraction processes a data corpus (e.g., a collection of documents) in batch mode and extracts "whatever facts it can find" [88]. In West et al. [88], it is referred to as the "push" model. The Never-Ending Language Learner (NELL) project [49], which performs continuous machine reading of the Web, provides one specific example of non-targeted extraction. *Targeted* (or focused) extraction, on the other hand, refers to extracting facts for specific entities (a.k.a. "slot filling"). In [88], it is referred to as the "pull" paradigm. For example, West et al. [88] leverage question answering techniques, and aim to learn the best set of queries to ask, to find missing entity attributes.

The methods we will be discussing in this chapter fall under closed and targeted information extraction. Specifically, our interest will be in filtering an incoming stream of documents for information pertinent to a pre-defined set of entities, and extracting values of specific properties of entities from those documents in order to extend an existing knowledge base. Below, we briefly look at a number of other tasks that fall within the general problem area of harvesting entity-related information from text. These are loosely organized around entity types (Sect. 6.1.1), attributes (Sect. 6.1.2), and relationships (Sect. 6.1.3). Note that we are not aiming for an in-depth or exhaustive treatment of these topics here. Our goal is merely to give a high-level overview of research in this area.

6.1.1 Class-Instance Acquisition

There is a large body of research on semantic class learning (a.k.a. *hyponym extraction*), which focuses on "is-A" (or "instanceOf") relations between instances and classes (hypernyms). In our case, instances are entities, while classes refer to entity types (e.g., "country") or semantic categories (e.g., "countries with nuclear power plants"). We briefly review related work from two directions: (1) obtaining additional instances (entities) that belong to a given semantic class (entity type) and (2) obtaining (additional) semantic classes (entity types) for a given instance (entity).

6.1.1.1 Obtaining Instances of Semantic Classes

Concept expansion is the problem of expanding a seed set of instances that belong to the same semantic class with additional instances. *Set expansion* is a closely related task, where the underlying semantic class is not defined explicitly via a textual label, only implicitly through the seed set of examples. The inherent ambiguity of the seeds makes set expansion a considerably harder task than concept expansion. Both concept and set expansion are commonly cast as a ranking task: Given a set of seed instances and (optionally) a class label, return a ranked list of instances that belong to the same target class.

One particular application of class-instance acquisition techniques is to extend tail types or categories of knowledge bases. We note that this is similar to the task of *similar entity search* (using the class label as the keyword query), which we have discussed in Sect. 4.5. However, since the overall goal here is KB expansion, the ranking is performed over a different data source than the KB, such as web documents [59] (often accessed via general-purpose web search engines [85, 86]), HTML tables [82], HTML lists [33], search queries [57], or a combination of multiple sources [61]. We note that many of the concept expansion approaches concentrate on noun phrases as instances, which are not necessarily entities (e.g., "red," "green," and "blue" are instances of the concept "colors"). Moreover, even when specifically entities are targeted, they may be identified only via their surface forms (noun phrases), and not as unique identifiers. In that case, an additional linking step is required to anchor them in a knowledge base [52]. When entities are coming from a knowledge base, then the class-instance acquisition process may incorporate additional semantic constraints, by considering entity properties in the KB [77].

Regarding the size of the seed set, Pantel et al. [59] show that "only few seeds (10–20) yield best performance and that adding more seeds beyond this does not on average affect performance in a positive or negative way." Vyas et al. [81] investigate the impact that the composition of seed sets has on the quality of set expansions and show that seed set composition can significantly affect expansion performance. They further show that, in many cases, the seed sets produced by an average (i.e., not expert) editor are worse than randomly chosen seed sets. Finally, they propose algorithms that can improve the quality of set expansion by removing certain entities from the seed set. Inevitably, the lists generated by set expansion systems will contain errors. Vyas and Pantel [80] propose semi-supervised techniques, which require minimal annotation effort, to help editors clean the resulting expanded sets.

6.1.1.2 Obtaining Semantic Classes of Instances

Reversing the concept expansion task, we now take a particular instance as input and seek to obtain semantic classes for that instance. In the context of knowledge bases, this problem is known as *entity typing*: Given an entity as input, automatically assign (additional) types to that entity from a type catalog. This task is addressed in two flavors: when the entity already exists in the KB and when it does not.

We start with the task of assigning types to entities that are already in the knowledge base. It is shown in [60] that, while seemingly a straightforward approach, traditional reasoning cannot be used to tackle this problem, due to the incompleteness and noisy nature of large-scale KBs. Gangemi et al. [30] extract types for DBpedia entities, based on the corresponding natural language definitions of those entities in Wikipedia (i.e., Wikipedia abstracts). Their approach relies on a number of heuristics and makes heavy use of Wikipedia's markup conventions. A more general approach is to frame the type prediction task as a hierarchical multi-label classification problem ("hierarchical because we assume the types to be structured in a hierarchy, and it is a multilabel problem because instances are allowed to have more than one type" [46]). Paulheim and Bizer [60] exploit links between instances in the knowledge base as indicators for types, without resorting to features specific to any particular KB. Melo et al. [46] present a top-down prediction approach and employ local classifiers and local feature selection per node. "The local training sets are created including the instances belonging to the target class as positive examples and the instances belonging to its sibling classes as negative examples" [46]. This method is shown to improve scalability without sacrificing performance.

The other flavor of entity typing is when the entity is not present in the KB. This is closely related to the task of identifying *emerging* or *tail* entities, i.e., entities that are not (yet) prominent enough to be included in a human-curated knowledge base. Often, a tail entity is informally defined as not being present in Wikipedia [43, 50, 52]. Fine-grained typing of entities has been long established as a subtask in named entity recognition [25, 69] (cf. Sect. 5.1.1). These early works, however, are limited to flat sets of several dozen types. It has been only more recently that entity typing is performed against type systems of large-scale knowledge bases, i.e., hierarchical type taxonomies with hundreds of types [43, 52, 92]. Lin et al. [43] introduce the *unlinkable noun phrase* problem: Given a noun phrase that cannot be linked to a knowledge base, determine whether it is an entity, and, if so, return its fine-grained semantic types. To decide whether the noun phrase is an entity, they train a classifier with features primarily derived from mentions of the noun phrase over time in a timestamped corpus (specifically, the Google Books Ngram Corpus [47]). In stage two, the semantic type of a noun phrase is predicted by (1) finding the textual relations it occurs with, (2) finding linked entities that share the same textual relations, and (3) propagating types from those linked entities. HYENA [92] employs a multi-label classification approach, using a rich set of mention, part-of-speech, and gazetteer features. The PEARL system [52] considers typed relational patterns (learned by PATTY [53]) and the type signatures of patterns the new entity occurs with (e.g., "⟨`singer`⟩ `released` ⟨`album`⟩" or "⟨`person`⟩ `nominated for` ⟨`award`⟩"). This is similar in spirit to Lin et al. [43], but the patterns in [43] are not typed. Additionally, PEARL attempts to resolve inconsistencies among the predicted types using integer linear programming. Mohapatra et al. [50] point out the synergy between named entity linking and entity typing: "knowing the types of unfamiliar entities helps disambiguate mentions, and words in mention contexts help assign types to entities." Mohapatra et al. [50] present a bootstrapping system for solving the two tasks jointly.

6.1.2 Class-Attribute Acquisition

Another area of automatic knowledge acquisition concerns the discovery of relevant attributes (and relationship types) of classes. For example, for the class “country,” the set of attributes includes “capital city,” “population,” “currency,” etc. The task is commonly formulated as a ranking problem: Given an input class label, return a ranked list of attributes. Candidate attributes are generally obtained by employing an *instance-driven* extraction strategy, i.e., by “inspecting the attributes of individual instances from that class” [58], thereby enjoying access to substantially more input data. We will look at attribute extraction for specific instances (entities) in Sect. 6.3. Alternatively, attributes may be extracted by using only the class label, without the set of instances, see, e.g., [78, 79].

A broad variety of data sources have been utilized, including web documents [78], web tables [12, 91], web search query logs [56, 58], and Wikipedia category labels [54]. Methods for attribute extraction range from simple extraction patterns [4] and term frequency statistics [58, 78] to probabilistic topic modeling techniques [64].

6.1.3 Relation Extraction

Relation extraction refers to the task of extracting relationships between entities from a data collection (usually, from documents). Relationships are generally defined in the form of tuples, however, most relation extraction systems focus on binary relationships between named entities; these take the form of subject-predicate-object triples (e.g., “`X marriedTo Y`” or “`X locatedIn Y`”). Early work has focused on pre-defined relationships between pairs of entities. Traditional relation extraction benchmarking campaigns include the Message Understanding Conference (MUC-7, *template relations* task [15]) and the Automatic Content Extraction program (ACE, *relation detection and characterization* task [18]). Relation extraction is naturally approached by learning a binary classifier, using feature-based approaches [39, 96] or kernel methods [11, 95]. These *supervised* models can achieve high accuracy, but they require lots of hand-labeled training data. Moreover, they do not generalize to different (new) relationships.

To overcome these limitations, *semi-supervised* approaches, and in particular *boostrapping*, have been proposed and gained attention. The idea behind bootstrapping is to learn new extraction patterns by using previous instances extracted by the system as training data. More specifically, the general bootstrapping algorithm, for a given relationship, is as follows:

1. Take as input a small number of seed tuples (i.e., pairs of entities) that engage in the required relationship.
2. Find occurrences of seed tuples in documents and extract contextual patterns.
3. Identify the best (top-*k*) patterns and add those to the pattern set.

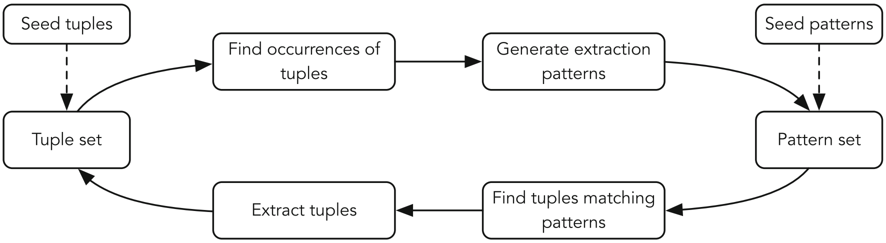

Seed tuples
Find occurrences of tuples
Generate extraction patterns
Seed patterns
Tuple set
Pattern set
Extract tuples
Find tuples matching patterns

Fig. 6.2 Bootstrapping for relation extraction

4. Find new instances of the relationship using the pattern set.
5. Add those instances to the tuple set and repeat from step 2.

The output is a set of tuples along with a set of patterns. Alternatively, the process may also be initialized with a small set of golden seed patterns (extraction rules), instead of the seed tuples [22] (and then start the bootstrapping process with step 4). See Fig. 6.2 for an illustration. Early examples of bootstrapping systems are DIPRE [10], Snowball [2], and KnowItAll [22]. More recent approaches include SOFIE [71] and NELL [14] . All these systems learn extractors for a predefined set of relationship types. While they clearly need less labeled data than supervised approaches, extending them to new relationships still requires seed data.

Distant supervision (also called *weak supervision*) uses a knowledge base to provide a "training set of relations and entity pairs that participate in those relations" [48]. Wu and Weld [89] applied this idea to learn relationships from Wikipedia, by matching attributes from infoboxes to corresponding sentences in the article. Others have used Freebase facts to train relational extractors on Wikipedia [48] or on a news corpus [65]. According to the *distant supervision assumption*, if two entities engage in a certain relationship, then any sentence containing those two entities might express that relationship [48]. Or, in a stronger form, "all sentences that mention these two entities express that relation" [65]. It has been argued that this assumption is too strong, and various refinements have been proposed [65, 76]. Riedel et al. [65] employ the following relaxation: "if two entities participate in a relation, at least one sentence that mentions these two entities might express that relation" [65], thereby casting the problem as a form of multi-instance learning. Surdeanu et al. [76] assume that a mention of two entities together expresses exactly one relationship, but the two entities might be related with different predicates across different mentions. This corresponds to a multi-instance multi-label learning setting. Distant supervision is shown to be scalable [35]. However, the training data is noisy and does not contain explicit negative examples. We further note that training data generation requires high-quality entity linking. State-of-the-art systems, such as Google's Knowledge Vault [19] and DeepDive [70], employ distant supervision and probabilistically combine results from multiple extractors.

Open information extraction is an alternative paradigm, pioneered by the TextRunner system [8], which employs *self-supervision*; training examples are generated heuristically using a small set of extraction patterns. The idea was taken up and developed further in a number of influential systems, including WOE [90], ReVerb [24], R2A2 [23], OLLIE [44], and ClausIE [16]. Open IE methods are domain-independent and very scalable but also highly susceptible to noise. Note that unlike previous approaches, they do not use canonical names for relationships, i.e., are “schema-less.” To be able to use the extracted relationships for KB expansion, the natural language relation phrases need to be aligned with (i.e., mapped to) existing knowledge base predicates; see, e.g., [13].

All the above methods are based on the idea of two named entities appearing in the same sentence. Some systems operate on the level of entity mentions (identified by a named entity recognizer), while others ground entities in a knowledge base (by performing entity linking). For the purpose of KBP, the latter one is more convenient. Finally, while most systems operate on unstructured text, there are also other sources that provide relational information, e.g., web tables [12, 42].

6.2 Entity-Centric Document Filtering

Document filtering is the task of identifying documents from a stream, from which relevant information can be extracted. This problem dates back to the Topic Detection and Tracking (TDT) track of the Text Retrieval Conference (TREC), which focused on finding and following events in broadcast news stories [3]. In this section, we are focusing on an entity-centric variant of the document filtering task. Unlike traditional filter topics, which are defined by a set of keyword queries, entities are described by (semi-)structured entries in a knowledge repository.

Definition 6.1 *Entity-centric document filtering* is the task of analyzing a time-ordered stream of documents and assigning a score to each document based on how relevant it is to a given target entity.

Consider the following scenario. A human content editor is responsible for the maintenance of the entries of one or multiple entities in a knowledge repository, e.g., Wikipedia. The filtering system monitors a stream of documents (news articles, blog posts, tweets, etc.) and identifies those that contain novel information pertinent to the target entities. Each document is assigned a (relevance) score; whenever the editor checks the system, which may be several times a day or once a week, the system presents a relevance-ordered list of documents that have been discovered since the last time the system was checked. The editor reviews some or all of these documents; the stopping criterion may be a score threshold or her time available.

Upon examining a document, the editor may decide to make changes to the KR entry of a given target entity. That document will then serve as provenance, i.e., the change made in the knowledge repository can be tracked back to this document as the original source. For example, in Wikipedia, changes are substantiated by adding a citation; the corresponding source is listed at the bottom of the article under the "References" section.

An abstraction of the above scenario was studied at the Knowledge Base Acceleration track at TREC 2012–2014, termed as *cumulative citation recommendation*: Filter a time-ordered corpus for documents that are highly relevant to a predefined set of entities [27]. One particularly tricky aspect of this task—and many other IR problems for that matter—is how to define relevance. At TREC KBA, it was initially based on the notion of "citation worthiness," i.e., whether a human editor would cite that document in the Wikipedia article of the target entity. This was later refined by requiring the presence of new information that would change an already up-to-date KR entry, rendering that document "vital." See Sect. 6.2.5.2 for details. We note that the need for this type of filtering also arises in other domains besides KBA, for instance, in social media, for business intelligence, crisis management, or celebrity tracking [97].

One of the main challenges involved in the entity-centric filtering task is how to distinguish between documents that are only tangentially relevant and those that are vitally relevant to the entity. Commonly, this distinction is captured by training supervised learning models, leveraging a rich set of signals as features, to replicate human judgments.

The attentive reader might have noticed that we are talking about knowledge repositories here, and not about knowledge bases. This is on purpose. For the document filtering task, we are unconcerned whether knowledge is in semi-structured or in structured format, since it is the human editor that needs to make manual updates to the entity's (KR or KB) entry.

6.2.1 Overview

Formally, the entity-centric filtering task is defined as follows. Let $\mathcal{D}$ be a time-ordered document stream and e be a particular target entity. The entity-centric filtering system is to assign a numerical score, $score(d;e)$, to each document $d \in \mathcal{D}$.

Figure 6.3 shows the canonical architecture for this task. For each document in the stream, the processing starts with an entity detection step: Does the document mention the target entity? (That is, for a single target entity. In the case of multiple

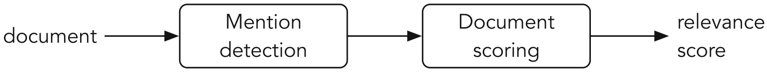


Fig. 6.3 Entity-centric document filtering architecture

Table 6.1 Notation used in Sect. 6.2

Symbol	Meaning
a	Entity aspect ($a \in \mathcal{A}$)
$\mathcal{A}$	Set of entity aspects
d	Document ($d \in \mathcal{D}$)
$\mathcal{D}$	Collection (stream) of documents
e	Entity ($e \in \mathcal{E}$)
h	Time (measured in hours)
$\mathcal{L}_e$	Set of entities e links to
T_d	Publication time of document d
$\mathcal{T}_e$	Type of entity e

target entities, this step has to be performed for each entity.) If the answer is yes, then, as a second step, the document needs to be scored do determine its relevance ("citation-worthiness") with respect to the target entity; otherwise, it is no longer of interest (i.e., gets zero assigned as score).

The mention detection component is discussed in Sect. 6.2.2. For document scoring, we present both unsupervised and supervised approaches in Sect. 6.2.3. State-of-the-art systems employ supervised learning based on a rich array of features; we present a selection of effective features in Sect. 6.2.4. Table 6.1 summarizes the notation used in this section.

6.2.2 Mention Detection

This step is responsible for determining whether a particular document contains a mention of a given target entity. Since this operation needs to be performed for all pairs of streaming documents and target entities, efficiency is paramount here. We wish to obtain high recall, so as not to miss any document that is potentially of interest. At the same time, we would also like to keep the number of false positives low. Therefore, much as it was done for entity linking in Sect. 5.4, mention detection relies on known surface forms of the target entity. We let $\mathcal{S}_e$ denote the set of known surface forms for entity e, where $s \in \mathcal{S}_e$ is a particular surface form. By definition, each entity has at least one canonical name, therefore $\mathcal{S}_e \neq \emptyset$. With $\mathcal{S}_e$ at hand, determining whether the document mentions the entity is conceptually as simple as:

$$mentions(d,e) = \begin{cases} 1, & \exists\, s \in \mathcal{S}_e : contains(d,s) \\ 0, & \text{otherwise}\,, \end{cases}$$

where $contains(d,s)$ denotes a case-insensitive string matching function that searches for string s in d.

The only remaining issue concerns the construction of the set of surface forms. For entities that already have an entry in the knowledge repository, this is readily available. For example, Balog et al. [7] extract surface forms (aliases) from DBpedia. For certain entity types, name variants may be generated automatically by devising simple heuristics, e.g., using only the last names of people [7] or dropping the type labels of business entities ("Inc.," "Co.," "Ltd.," etc.). Finally, considering the KBA problem setting, it is reasonable to assume that a human knowledge engineer would be willing to manually expand the set of surface forms. This was also done at TREC KBA 2013, where the oracle baseline system relied on a hand-crafted list of entity surface forms [26].

6.2.3 Document Scoring

Next, we need to estimate $score(d;e)$ for document d, which, as we know, potentially contains the target entity e. For convenience, we shall assume that this score is between 0 and 1.[2] Further, we assume that, for each target entity, a set of documents and corresponding (manually assigned) relevance labels are made available as training data. We distinguish between non-relevant (R0) and relevant documents, where relevance has two levels. Following the latest TREC KBA terminology (cf. Sect. 6.2.5.2), we shall refer to the lower relevance level (R1) as *useful* and the higher relevance level (R2) as *vital*.

It is worth pointing out that none of the document scoring techniques we will discuss attempt to explicitly disambiguate the mentioned entities (i.e., no entity linking is performed). The reason is that entity disambiguation requires the presence of either a textual description or relationship information about the given entity (cf. Sect. 5.6). In the KBA context, where target entities are often long-tail entities, this information is not necessarily available (yet) in the knowledge repository.

6.2.3.1 Mention-Based Scoring

A simple baseline approach, which was employed at TREC KBA, is to assign a score "based on the number of matches of tokens in the name" [28]. That is, the longest observed mention of e in the document is considered, normalized by the longest known surface form of e, thereby producing a score in $(0, 1]$. Formally:

$$score(d;e) = \frac{\max(\{l_s : s \in \mathcal{S}_e, contains(d,s)\})}{\max(\{l_{s'} : s' \in \mathcal{S}_e\})} ,$$

where l_s denotes the character length of surface form s.

[2] At TREC KBA, the score needs to be in the range $(0, 1000]$, but mapping to that scale is straightforward.

Other simple mention-based scoring formulas may also be imagined, e.g., based on the total number of occurrences of the entity in the document.

6.2.3.2 Boolean Queries

Efron et al. [21] perform filtering by developing highly accurate Boolean queries, called *sufficient queries*, for each entity. A sufficient query is defined as a "Boolean query with enough breadth and nuance to identify documents relevant to an entity *e* without further analysis or estimation." Essentially, the mention detection and scoring steps are performed jointly, by using a single Boolean query per target entity. Specifically, a sufficient query consists of two parts: (1) a *constraint clause*, which is the surface form of the entity, and (2) a *refinement clause*, which is a set of zero or more n-grams (bigrams in [21]), selected from the set of relevant training documents. An example query, expressed using the Indri query language, is

```
#band(#1(phyllis lambert) #syn(#1(canadian architect) #1(public
art))) ,
```

where `#1` matches as an exact phrase, `#band` is a binary AND operator, and `#syn` enumerates the elements of the refinement clause. That is, the document must contain the name of the entity (constraint clause) and any of the n-grams that are listed in the refinement clause.

The same idea may also be expressed in terms of our framework (cf. Fig. 6.3), by viewing the constraint clause as the mention detection step, and the refinement clause as a Boolean scoring mechanism that assigns either 0 or 1 as score.

6.2.3.3 Supervised Learning

The predominant approach to entity-centric filtering is to employ supervised learning. The notion of citation-worthiness "is not a precise definition that can easily be captured algorithmically" [7]. It is a combination of multiple factors that make a document useful/vital. The idea, therefore, is to focus on capturing and extracting as many of these contributing factors as possible, as features. Then, we let a machine learning algorithm figure out how to best combine these signals based on a manually labeled set of training documents. We shall discuss specific features in Sect. 6.2.4. For now, our concern is the supervised learning part, i.e., what type of model we want to train and how.

The entity-centric filtering task may be approached both as a classification and as a ranking problem. In the former case, a binary decision is made, where the classifier's confidence may be translated into a relevance score. In the latter case, relevance is directly predicted, and thresholding is left to the end user, i.e., she decides when to stop in the ranked list. We look at these two possibilities in more detail below. In terms of performance, ranking-based approaches were found to perform better in [6]. Later studies, however, have shown that it varies whether classification or ranking is more effective, depending on the (sub-)set of features used and on the relevance criteria that are targeted [31, 83].

Classification Balog et al. [7] propose two classification methods, referred to as *2-step* and *3-step classification* approaches. The first step, in both cases, corresponds to the mention detection phase, which may also be seen as a binary classification decision (whether the document contains the target entity or not). This is then followed by one or two additional binary classification steps. The 2-step approach makes a single classification decision to decide whether the document is vital or not. The 3-step approach first tries to separate non-relevant documents from relevant ones (R0 vs. R1 or R2), and in a subsequent step distinguish between useful and vital documents (R1 vs. R2). See Fig. 6.4 for an illustration. Notably, the same set of features are used in all classification steps. The difference lies in how documents get labeled as positive/negative instances during training. In 2-step classification, non-relevant documents constitute the negative class, while vital documents are the positive class; useful documents are not used so as not to "soften the distinction between the two classes that we are trying to separate" [7]. In 3-step classification, the "Relevant?" decision uses non-relevant and relevant documents as negative and positive instances, respectively, while the "Vital?" decision uses useful and vital documents as negatives and positives, respectively.

According to the experiments reported in [7], these two methods deliver very similar performance, making the simpler 2-step approach the recommended choice. We note that this stance may need to be revisited depending on the amount of training data available.

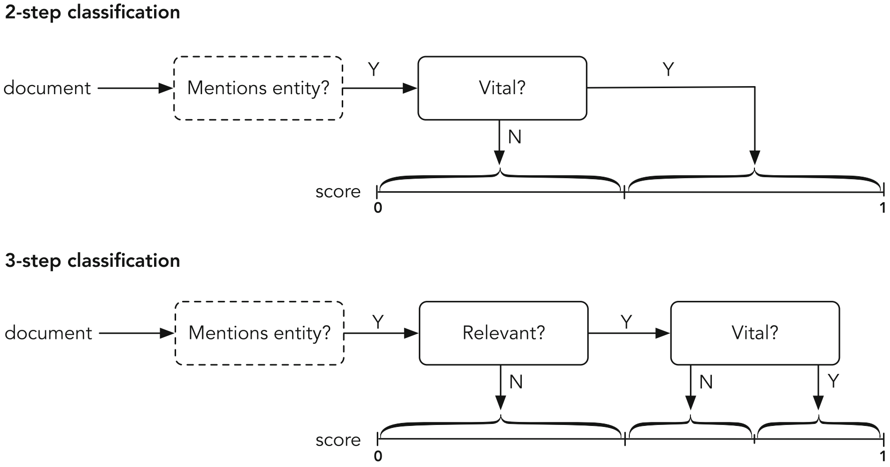


Fig. 6.4 Multi-step classification approaches proposed in [7]. Each box represents a binary classifier. The dashed box corresponds to the mention detection step

Learning-to-Rank Filtering may also be approached as a learning-to-rank problem: Estimate a numerical score for a document-entity pair. We refer back to Sect. 3.3.3 for a crash course on learning-to-rank. It is simpler than classification-based approaches in the sense that there is no additional mapping involved, i.e., the target score directly corresponds to the relevance level. Balog and Ramampiaro [6] compared pointwise, pairwise, and listwise learning-to-rank approaches and found that the pointwise approach yielded the best performance.

Global vs. Entity-Specific Models Another issue is whether to train a single global model, which is used for all target entities, or a separate entity-specific model for each entity. Commonly, the former option is selected [6, 7, 63, 83], as there is limited training data available for any individual entity. Also, the global model generalizes to previously unseen entities. Wang et al. [84] argue that "these models ignore the distinctions between different entities and learn a set of fixed model parameters for all entities, which leads [to] unsatisfactory performance when dealing with a diverse entity set." Therefore, they propose an entity-type-dependent discriminative mixture model, which involves an intermediate latent layer to model each entity's distribution across entity types.

6.2.4 Features

We present a selection of features from [7, 63], organized into the following four main groups:

- *Document features* estimate the "citation worthiness" of the document, independent of the target entity.
- *Entity features* capture characteristics of the target entity, based on its entry in the knowledge repository.
- *Document-entity features* express the relation between a particular document and the target entity.
- *Temporal features* model events happening around the target entity.

We discuss each feature group in turn. Table 6.2 provides a summary. We note that these features may be used both with binary classification and learning-to-rank approaches (cf. Sect. 6.2.3.3).

6.2.4.1 Document Features

Simple document features include various measurements of length, the document's source, and its language [7]. Reinanda et al. [63] consider several intrinsic characteristics of a document to help identify vital documents. In particular: informativeness, entity-saliency, and timeliness.

Table 6.2 Overview of features for the entity-centric document filtering task

Group	Feature	Description	Src.	Val.
Document features				
	l_{f_d}	Length of doc. field f (title, body, anchors)	S	N
	$len(d)$	Length of d in number of chunks or sentences	S	N
	$src(d)$	Document source (news, social, linking, etc.)	S	C
	$lang(d)$	Whether the document's language is English	S	B
	$aspectSim(d,a)$	Similarity between d and aspect a	S, KR	N
	$numEntities(d)$	Number of unique entities mentioned in d	S	N
	$numMentions(d)$	Total number of entity mentions in d	S	N
	$timeMatchY(d)$	Number of temporal expressions matching the document's creation year	S	N
	$timeMatchYM(d)$	Number of temporal expressions matching the document's creation year and month	S	N
	$timeMatchYMD(d)$	Number of temporal expressions matching the document's creation year, month, and day	S	N
Entity features				
	$\mathcal{T}_e$	Type of the entity (PER, ORG, etc.)	KR	C
	l_e	Length of the entity's description	KR	N
	$\vert\mathcal{L}_e\vert$	Number of related entities	KR	N
Document-entity features				
	$n(f_d,e)$	Number of mentions of e in document field f	S	N
	$firstPos(d,e)$	Term position of the first mention of e in d	S	N
	$firstPosNorm(d,e)$	$firstPos(d,e)$ normalized by the document length	S	N
	$lastPos(d,e)$	Term position of the last mention of e in d	S	N
	$lastPosNorm(d,e)$	$lastPos(d,e)$ normalized by the document length	S	N
	$spread(d,e)$	Spread (distance between first and last mentions)	S	N
	$spreadNorm(d,e)$	$spread(d,e)$ normalized by the document length	S	N
	$numSentences(d,e)$	Number of sentences mentioning e	S	N
	$mentionFrac(d,e)$	Mentions of e divided by all entity mentions in d	S	N
	$sim_{Jac}(d,e)$	Jaccard similarity between d and e	S, KR	N
	$sim_{cos}(d,e)$	Cosine similarity between d and e	S, KR	N
	$sim_{KL}(d,e)$	KL divergence between d and e	S, KR	N
	$numRelated(f_d,e)$	Number of unique related entities mentioned in document field f	S	N
Temporal features				
	$sv(e)$	Average hourly stream volume	S	N
	$sv_h(e)$	Stream volume over the past h hours	S	N
	$\Delta sv_h(e)$	Change in stream volume over the past h hours	S	N
	$isBurstSv_h(e)$	Whether there is a burst in stream volume	S	B
	$wpv(e)$	Average hourly Wikipedia page views	U	N

(continued)

Table 6.2 (continued)

Group	Feature	Description	Src.	Val.
	$wpv_h(e)$	Wikipedia page views volume in the past h hours	U	N
	$\Delta wpv_h(e)$	Change in Wikipedia page views volume	U	N
	$isBurstWpv_h(e)$	Whether there is a burst in Wikipedia page views	U	B
	$burstValue(d,e)$	Document's burst value, based on Wikipedia page views	U	N

Source can be stream (S), knowledge repository (KR), or usage data (U). Value can be numerical (N), categorical (C), or Boolean (B)

The intuition behind *informativeness* features is that a document that is rich in facts is more likely to be vital. One way to measure this is to consider the aspects of entities that are mentioned in the document, where *aspects* are defined as "key pieces of information with respect to an entity" [63]. The set of aspects $\mathcal{A}$ is constructed by taking the top-k ($k = 50$ in [63]) most common section headings (with stopwords removed) of Wikipedia articles belonging to a set of categories of interest (here, *Person* or *Location*). Then, a textual representation of each entity aspect $a \in \mathcal{A}$ is constructed by aggregating the contents of those sections. Finally, the cosine similarity between bag-of-words representations of document d and aspect a are computed:

$$aspectSim(d,a) = \cos(\mathbf{d},\mathbf{a}) \ .$$

Reinanda et al. [63] also consider relations from both open and closed information extraction systems, but those are reported to perform worse than Wikipedia aspects.

Entity-saliency refers to the notion of how prominent the target entity is in the document, by considering other entities mentioned in the document. Here, we only consider two simple features of this kind that do not depend on the specific target entity. One is the number of unique entities mentioned in the document ($numEntities(d)$), the other is the total number of entity mentions in d ($numMentions(d)$).

The *timeliness* of a document captures "how timely a piece of information mentioned in the document is" [63]. It is measured by comparing the document's creation/publication time, T_d (which is obtained from document metadata), with the temporal expressions present in the document. Specifically, the following equation counts the occurrences of the year of the document's creation time appearing in d:

$$timeMatchY(d) = count(year(T_d),d) \ .$$

Similarly, (year and month) and (year, month, and date) expressions of T_d are also counted:

$$timeMatchYM(d) = count(yearmonth(T_d),d) \ ,$$

$$timeMatchYMD(d) = count(yearmonthday(T_d),d) \ .$$

6.2.4.2 Entity Features

These features capture our knowledge about a given target entity e, including (1) the type of the entity ($\mathcal{T}_e$)[3]; (2) the length of the entity's description in the knowledge repository (l_e); and (3) the number of related entities $|\mathcal{L}_e|$, where $\mathcal{L}_e$ is defined as the set of entities that e links to [7]. Alternative interpretations of $\mathcal{L}_e$ are also possible.

6.2.4.3 Document-Entity Features

Document-entity features are meant to express how vital a given document d is for the target entity e. We assume a fielded document representation, which would typically include title, body, and anchors fields. Several features are developed in [7] to characterize the occurrences of the target entity in the document, including the total number of mentions in each document field, the first and last position of mentions in the document's body, and the spread between the first and last mentions. To measure entity-salience, Reinanda et al. [63] calculate the number of sentences mentioning the target entity and the fraction of the target entity's mentions with respect to all entity mentions in the document. All these features may be computed for both full and partial entity name matches (i.e., requiring the detection of one of the surface forms for a full match and considering only the last names of people for a partial match) [7].

A second group of features looks at the similarity between the term-based representations of the target entity (i.e., the entity's description in the knowledge repository, e.g., the Wikipedia article corresponding to the entity) and the document, employing various similarity measures (cosine, Jaccard, or KL divergence) [7]. Finally, we may also take into consideration what other entities, related to e, are mentioned in the document [7]. Formally:

$$numRelated(f_d,e) = \sum_{e' \in \mathcal{L}_e} \mathbb{1}(n(f_d,e')) \ ,$$

where $\mathcal{L}_e$ is the set of related entities, and $\mathbb{1}(n(f_d,e'))$ is 1 if entity e' is mentioned in field f of document d, otherwise 0.

6.2.4.4 Temporal Features

Temporal features attempt "to capture if something is happening around the target entity at a given point in time" [7]. The volume of mentions of the entity in the streaming corpus offers one such temporal signal. Another—independent—signal may be obtained from an usage log that indicates how often people searched for

[3] Unlike in other parts of the book, type here is not a set but is assumed to take a single value (PER, ORG, LOC, etc.).

or looked at the given entity. For example, Balog et al. [7] leverage Wikipedia page view statistics for that purpose. In both cases, the past h hours are observed in a sliding windows manner ($h = \{1,2,3,6,12,24\}$), to detect changes or bursts. Formally, let $sv(e)$ denote the *stream volume* of entity e, i.e., the average number of documents mentioning e per hour. This is computed over the training period and serves as a measure of the general popularity of the entity. Further, $sv_h(e)$ is the number of documents mentioning e over the past h hours. The change relative to the normal volume, over the past h hours, is expressed as:

$$\Delta sv_h(e) = \frac{sv_h(e)}{h \times sv(e)} .$$

All these "raw" quantities, i.e., $sv(e)$, $sv_h(e)$, and $\Delta sv_h(e)$, are used as features. In addition, Balog et al. [7] detect bursts by checking if $\Delta sv_h(e)$ is above a given threshold τ, which they set to 2:

$$isBurstSv_h(e) = \begin{cases} 1, & \Delta sv_h(e) \geq \tau \\ 0, & \text{otherwise} . \end{cases}$$

For Wikipedia page views, the definitions follow analogously, using the page view count instead of the number of documents mentioning the entity. Alternatively, Wang et al. [83] define the *burst value* for each document-entity pair as follows:

$$burstValue(d,e) = \frac{N \times wpvd(e,T_d)}{\sum_{i=1}^{N} wpvd(e,i)} , \tag{6.1}$$

where N is the total number of days covered by the stream corpus, $wpvd(e,i)$ is the number of page views of the Wikipedia page of entity e during the ith day of the streaming corpus, and T_d refers to the date when document d was published ($T_d \in [1..N]$). We note that this formulation considers information "from the future" (i.e., beyond the given document's publication date); this may be avoided by replacing N with T_d in Eq. (6.1).

6.2.5 Evaluation

We discuss the evaluation of entity-centric filtering systems as it was carried out at the TREC 2012–2014 Knowledge Base Acceleration track.

6.2.5.1 Test Collections

For each edition of the TREC KBA track, a streaming test collection was developed. Content from the 2012 corpus is included in the 2013 corpus. Similarly, the 2013 corpus is subsumed by the 2014 one. The document collection is composed of

Table 6.3 TREC KBA test collections for entity-centric document filtering

Name	Time period	Size	#Docs	#Target entities
Stream Corpus 2012 [27][a]	Oct 2011–Apr 2012	1.9TB	462.7M	29
Stream Corpus 2013 [26][b]	Oct 2011–Feb 2013	6.5TB	1B	141
Stream Corpus 2014 [28][c]	Oct 2011–Apr 2013	10.9TB	1.2B	109

Size is for compressed data

[a] http://trec-kba.org/kba-stream-corpus-2012.shtml

[b] http://trec-kba.org/kba-stream-corpus-2013.shtml

[c] http://trec-kba.org/kba-stream-corpus-2014.shtml

three main sources: news (global public news wires), social (blogs and forums), and linking (content from an URL-shortening service). The 2013 and 2014 versions include additional substreams of web content. The streaming corpus is divided into hourly batches, allowing entities to evolve over time.

The 2014 corpus adds to the previous two versions not only in terms of duration but also in terms of NLP tagging information, by deploying BBN's Serif NLP system [9]. The official corpus (listed in Table 6.3) is tagged with the recognized named entities. An extended version (16.1 TB in size) also contains within-document coreference resolution and dependency parsing annotations. In both cases, annotations are available only for the (likely) English documents. Furthermore, Google has released Freebase annotations for the TREC KBA 2014 Stream Corpus (FAKBA1).[4] A total of 9.4 billion entity mentions are recognized and disambiguated in 394M documents. The format is identical to that of the ClueWeb Freebase annotations, cf. Sect. 5.9.2.

The set of target entities for 2012 consists of 27 people and 2 organizations from Wikipedia. The 2013 set focuses on 14 inter-related communities of entities, a total of 98 people, 19 organizations, and 24 facilities from Wikipedia and Twitter. The 2014 entities are primarily long-tail entities that lacked Wikipedia entries; 86 people, 16 organizations, and 7 facilities were hand-picked from within a given geographic region (between Seattle and Vancouver).

6.2.5.2 Annotations

The annotation guidelines evolved over the years. Initially, human annotators were given the following instructions: "Use the Wikipedia article to identify (disambiguate) the entity, and then imagine *forgetting* all info in the Wikipedia article and asking whether the text provides any information about the entity" [27].

[4] http://trec-kba.org/data/fakba1/.

Documents were annotated along two dimensions: (1) whether it mentions the target entity explicitly and (2) whether it is relevant. The annotation matrix is shown in Fig. 6.5, where rows correspond to mentions and columns denote the level of relevance. The relevance levels were defined as follows:

- *Garbage*: not relevant, e.g., spam.
- *Neutral*: not relevant, i.e., no information can be learned about the target entity.
- *Relevant*: relates indirectly to the target entity, e.g., discusses topics or events that likely impact the entity.
- *Central*: relates directly to the target entity, i.e., the entity is a central figure in the mentioned topics or events. The document would be cited in the Wikipedia article of the entity.

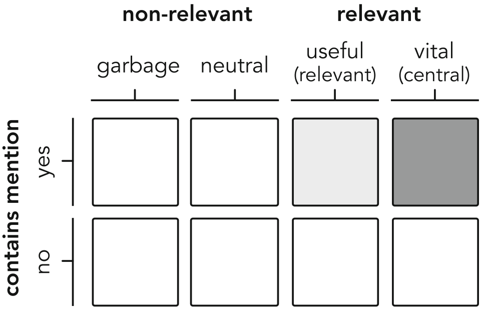
For the 2013 edition, the instructions given to the assessors were revised, in order to make a better distinction between the two highest relevance levels. The "central" judgment was replaced by "vital," reserving this rating to documents that would trigger changes to an already up-to-date KR entry. The second highest rating level was also renamed, from "relevant" to "useful," to include documents that are "citation-worthy as background information that might be used when writing an initial dossier but do not present timely or 'fresh' changes to the entity" [26]. That is:

- *Useful*: contains citation-worthy background information that should be included in the entity's KR entry.
- *Vital*: presents timely information that would imply changes to an already up-to-date KR entry.

According to Frank et al. [26], these changes "removed a large area of subjectivity in the notion of 'citation-worthiness'" (as observed by the increased inter-annotator agreement). However, deciding what to include in the entity's KR entry and what up-to-dateness means still involves subjective judgment. For less noteworthy entities there seems to be increased subjectivity, hence lower inter-annotator agreement [28].

Theoretically speaking, a document may be relevant, even if it does not mention the target directly (e.g., through relations to other entities mentioned in the

Fig. 6.5 Document annotation matrix from the TREC KBA track. The goal is to identify vital documents (top right corner). The labels in parentheses correspond to the 2012 terminology



document). In practice, relevance without an explicit mention of the target entity rarely happens. In 2014, the annotations were simplified by dropping the "mention" dimension; the "useful" and "vital" ratings imply that the entity was mentioned in the document.

Crucially, for all three editions of the TREC KBA track, the judgments were performed *pre-hoc*, i.e., there might be relevant documents (true positives) that were not seen by annotators. The recall of annotations is estimated to be over 90% [27]. Documents that are not primarily English were discarded. Furthermore, there is no novelty requirement, i.e., all documents that report the same event within a given timeframe are deemed citation-worthy. That timeframe is decided subjectively by the assessor; generally, less than a week and more than an hour. Annotations for an early portion of the stream are provided as training data.

6.2.5.3 Evaluation Methodology

Systems are required to process the stream corpus in chronological order, in hourly batches, and assign a confidence score in (0,1000] to each citation-worthy document. At any given point in time, systems may only access information about entities from the past. Evaluation uses set-based measures: F1-measure and scaled utility. See Eq. (5.10) for the definition of the F1-measure. *Scaled utility* is a measure from general information filtering that measures a system's ability to accept relevant and reject non-relevant documents from a stream [66]. Formally,

$$\mathrm{SU} = \frac{\max(\mathrm{T11NU}, \mathrm{MinNU}) - \mathrm{MinNU}}{1 - \mathrm{MinNU}} , \tag{6.2}$$

where T11U is the linear utility, which gives a credit of 2 for a relevant document retrieved (i.e., true positives) and a debit of 1 for a non-relevant document retrieved (i.e., false positives):

$$\mathrm{T11U} = 2 \times \mathrm{TP} - \mathrm{FP} .$$

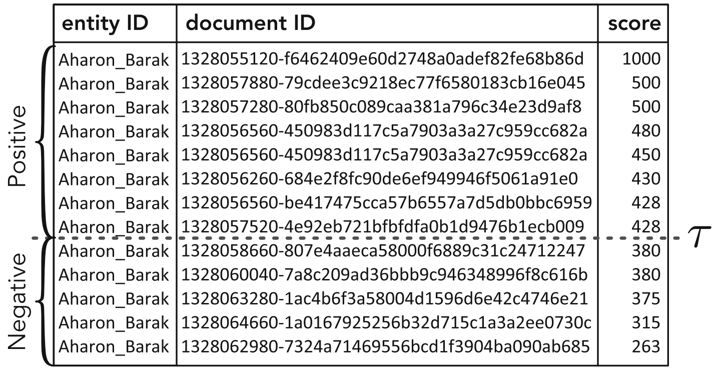
A normalized version is computed using:

$$\mathrm{T11NU} = \frac{\mathrm{T11U}}{\mathrm{MaxU}} ,$$

where MaxU is the maximum possible utility:

$$\mathrm{MaxU} = 2 \times (\mathrm{TP} + \mathrm{FN}) .$$

Finally, MinNU in Eq. (6.2) is a tunable parameter, set to -0.5.


Positive
Negative
τ

entity ID	document ID	score
Aharon_Barak	1328055120-f6462409e60d2748a0adef82fe68b86d	1000
Aharon_Barak	1328057880-79cdee3c9218ec77f6580183cb16e045	500
Aharon_Barak	1328057280-80fb850c089caa381a796c34e23d9af8	500
Aharon_Barak	1328056560-450983d117c5a7903a3a27c959cc682a	480
Aharon_Barak	1328056560-450983d117c5a7903a3a27c959cc682a	450
Aharon_Barak	1328056260-684e2f8fc90de6ef949946f5061a91e0	430
Aharon_Barak	1328056560-be417475cca57b6557a7d5db0bbc6959	428
Aharon_Barak	1328057520-4e92eb721bfbfdfa0b1d9476b1ecb009	428
Aharon_Barak	1328058660-807e4aaeca58000f6889c31c24712247	380
Aharon_Barak	1328060040-7a8c209ad36bbb9c946348996f8c616b	380
Aharon_Barak	1328063280-1ac4b6f3a58004d1596d6e42c4746e21	375
Aharon_Barak	1328064660-1a0167925256b32d715c1a3a2ee0730c	315
Aharon_Barak	1328062980-7324a71469556bcd1f3904ba090ab685	263

Fig. 6.6 Scoring of documents for a given target entity, using a cutoff value of 400. Figure is based on [6]

To be able to apply the above set-based measures, the output of the filtering system needs to be divided into positive and negative sets (which are then compared against the positive/negative classes defined in the ground truth). It is done by employing a confidence threshold τ. Documents with a confidence score $\geq \tau$ are treated as positive instances, while all remaining documents are negatives. This idea is illustrated in Fig. 6.6. At TREC KBA, a scoring tool sweeps the confidence cutoff between 0 and 1000 in steps of 50. For each entity, the highest performing cutoff value is chosen, then F1/SU scores are averaged over the set of all target entities (i.e., macro-averaging is used). Alternatively, a single cutoff value may be applied for all entities [6]. The strictness of evaluation is considered on two levels: (1) only central/vital documents are accepted as positive, and (2) both relevant/useful and central/vital documents are treated as positives.

6.2.5.4 Evaluation Methodology Revisited

The official TREC KBA evaluation methodology is time-agnostic. Even though systems process the stream corpus in hourly batches, evaluation considers all returned documents (above the threshold) as a single set, thereby ignoring the streaming nature of the task. Intuitively, not all *time batches* (e.g., hourly periods) are equally important; e.g., some might be high intensity burst periods, while others hardly contain any relevant documents. Additionally, the cutoff threshold τ is a free parameter; it remains an open issue how to set it in a way that it enables a fair comparison between systems. Furthermore, the actual ranking of documents within a given batch (which are above the cutoff threshold) is not taken into account. Dietz et al. [17] present a time-aware evaluation framework, which is composed of three main elements: (1) slicing the entire evaluation period into time batches, (2) evaluating the performance of each batch using standard rank-based measures, (3) aggregating the batch evaluation results into a single system-level score.

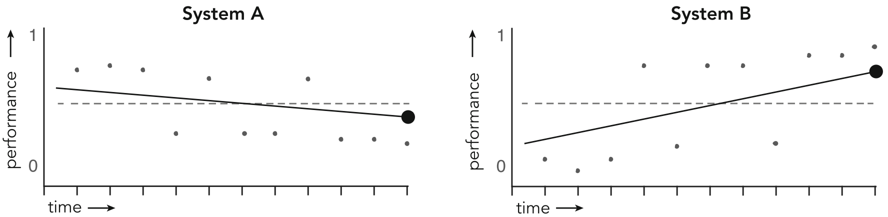


Fig. 6.7 Time-aware evaluation of document filtering systems proposed by Kenter et al. [40]. Systems A and B have the same average performance (dashed line), however, A degrades over time while B improves. The solid straight line is the fitted trend line. Systems are compared in terms of "estimated end-point performance" (the large dots)

Kenter et al. [40] argue that a crucial aspect of the document filtering task, which sets it apart from other document classification problems, is how the performance of a system changes over time. They propose to capture if a system improves, degrades, or remains the same over time, by employing trend analysis. The idea is to measure performance at regular time intervals, using any existing evaluation measure (either set-based or rank-based). Then, by fitting a straight line to these data points, one can estimate the expected performance of the system at the end of the evaluation period. This "estimated end-point performance" can serve as the basis of comparison across systems; see Fig. 6.7.

6.3 Slot Filling

Once documents that potentially contain new information about a given target entity have been identified, the next step is to examine the contents of those documents for new facts. The extracted facts can then be used to populate the knowledge base entry of the target entity. (Notice the shift from a knowledge repository to a knowledge base, as we shall now operate on the level of specific facts.) We look at one particular variant of this task, which assumes that the set of predicates ("slots") that are of interest is provided.

Definition 6.2 *Slot filling* is the task of extracting the corresponding values for a pre-defined set $\mathcal{P}$ of predicates, for a given target entity e, from a previously identified set of documents $\mathcal{D}_e$.

Stating the problem in terms of SPO triples, we are to find triples (e, p, o), where e is the target entity, $p \in \mathcal{P}$ is one of the target predicates (slots), and the object value o is to be extracted from $\mathcal{D}_e$. The resulting triples (facts) can then be used to populate the

Table 6.4 Entity attributes targeted at the TAC 2009 KBP slot filling task [45]

Entity type	Attributes
Person	alternate names; age; date and place of birth; date, place, and cause of death;
	national or ethnic origin; places of residence; spouses, children, parents,
	siblings, and other familial relationships; schools attended; job titles;
	employers; organizational memberships; religion; criminal charges
Organization	alternate names; political or religious affiliation; top members/employees;
	number of employees; members; member of; subsidiaries; parents; founder; date founded;
	date dissolved; location of headquarters; shareholders; website
Geo-political entity	alternate names; capital; subsidiary organizations; top employees;
	active political parties; when established; population; currency

KB. Note that, depending on the object's value, a distinction can be made between attributes (objects with a literal value) and relationships (where the object is another entity). In this section, we shall assume that objects are strings (literals), hence we are always targeting attributes. Consequently, we will refer to the predicates for which values are to be found as *attributes* and to the corresponding object values as *slot values*. These values may subsequently be linked within the knowledge base. We also note that relationship extraction is typically separately addressed through a dedicated set of techniques, which we have already surveyed in Sect. 6.1.3.

As an illustration, Table 6.4 lists the attributes that were targeted by the slot filling task of the Knowledge Base Population (KBP) track at the 2009 Text Analysis Conference (TAC). According to the categorization scheme presented in Sect. 6.1, slot filling is a closed and targeted information extraction task. We are targeting specific entities and are interested in a closed set of attributes.

6.3.1 Approaches

There are two main groups of approaches to slot filling: (1) *pattern-based* methods, which extract and generalize lexical and syntactic patterns (semi-)automatically [41, 72, 93], and (2) *supervised classification* methods, which consider entities and candidate slot values as instances and train a classifier through distant supervision [1, 67, 73]. Here, we will focus exclusively on the latter group, as it is more relevant to our entity-oriented perspective.

Training data for slot filling consists of documents that are explicitly marked up with slot values. The challenge is that such training datasets are restricted in size. Thus, "traditional supervised learning, based directly on the training data, would provide limited coverage" [36]. On the other hand, the facts that are already in the knowledge base can be exploited to produce training examples [48].

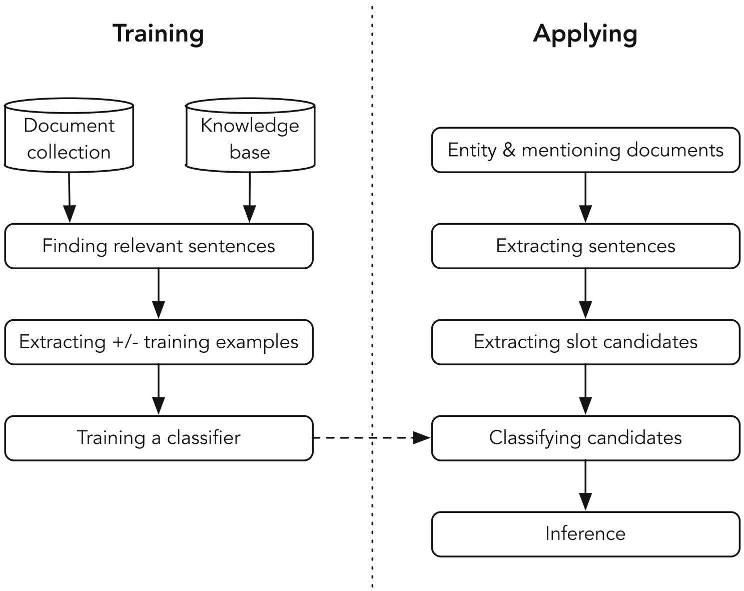


Fig. 6.8 Slot filling architecture using distant supervision

Figure 6.8 shows the architecture of a typical slot filling system that involves distant supervision [73, 75]. The steps of the training phase, for a given attribute $p \in \mathcal{P}$, are the following.

1. Finding sentences in the document collection that mention entities with already existing values for attribute p in the KB. The slot value candidates are identified in these sentences using an *extended named entity recognizer*. This extended recognizer should be able to detect text spans that are possible values of the targeted attribute and label those with a type (e.g., temporal expressions, job titles, religion, etc.).
2. All sentences that contain an entity and its corresponding slot value from the KB, are positive training examples. All other sentences, which contain a candidate slot value with the matching type (e.g., a temporal expression) but not with the correct slot value from the KB, are taken as negative training examples.
3. A classifier is trained on entity and slot value candidate pairs. It can be a global multi-class classifier for all attributes or multiple binary classifiers for each individual attribute. Commonly, three main groups of features are used: (i) information about the entity and the slot value candidate (e.g., terms and the type of the slot value candidate as labeled by the extended NER); (ii) surface features (e.g., terms surrounding the entity and the slot value candidate); (iii) syntactic features (e.g., dependency path between the entity and the slot value candidate) [75].

Applying the learned model involves the following steps:

1. Retrieving sentences that mention the target entity.
2. Extracting candidate slot values using the extended named entity recognizer.
3. Classifying the target entity and candidate slot value pairs according to the learned model.
4. A final inference step decides what slot values to return as output. For example, certain attributes can take only a single value (e.g., age or city of birth) while others can take a list of values (e.g., parents or alternate names). If multiple slot values are extracted for the same attribute, then those decisions need to be merged. One can return the slot value with the highest overall score or the one with the highest combined score. Additionally, a confidence threshold may be applied; slot values below that threshold are discarded.

6.3.2 Evaluation

The slot filling task has been addressed at the Web People Search (WePS) evaluation campaign, targeting only persons [5], and at the Knowledge Base Population track of the Text Analysis Conference (TAC KBP), targeting persons, organizations, and geo-political entities [37, 45, 74]. For details of the test collections, evaluation methodology, and evaluation metrics, we refer to the overview papers of the respective campaigns [5, 74].

Slot filling remains to be an extremely challenging task. According to the 2014 TAC KBP evaluation results, the best performing team achieved an F1-score of 0.36, which is approximately at 52% of human annotator performance [74].

6.4 Summary

This chapter has addressed the topic of populating knowledge bases with new facts about entities. We have approached this task using a two-step pipeline. The first component, *entity-centric document filtering*, aims to identify documents, from an incoming stream, that contain information that would trigger an update to a given entity's entry in the knowledge repository. The second component, *slot filling*, focuses on extracting new facts about an entity from a given input document. More specifically, for a pre-defined set of predicates, this component aims to find the values corresponding to those predicates.

When it comes to making actual updates to a knowledge base, one could argue that manual curation by humans is, and will remain, indispensable. Otherwise, systems could enter into a vicious cycle of feeding themselves false information that they themselves would be generating. The techniques presented in this chapter, however, represent crucial tools to support knowledge engineers and to help them perform their work effectively.

Knowledge bases are (some of) the core building blocks of next generation search engines. As we shall see in Part III, they can be utilized, among other tasks, to help understand users' information needs, improve the ranking of documents, present rich search results, aid users in formulating their queries, and recommend related content.

6.5 Further Reading

The automatic updating and expansion of knowledge bases still presents a number of challenges. A knowledge base can never be truly complete and up-to-date. In reality, some information is changing so rapidly that it would be difficult (or hardly meaningful) to materialize it in a knowledge base. Examples include movie sales, stock prices, or chart positions of songs. Preda et al. [62] refer to it as *active knowledge*, and propose to gather and integrate data from web services on the fly, transparently to the user, as if it was already contained in the knowledge base. In a recent study, Galárraga et al. [29] make the first steps toward predicting the completeness of certain properties of objects in a KB.

Cold start knowledge base population refers to the ambitious goal of constructing a KB from raw input data (e.g., a document collection). Cold start KBP encompasses a number of subtasks, including the discovery of *emerging entities* [32, 34], the clustering of entity mentions, relation extraction, and slot filling. Knowledge can also be inferred from what is already in the knowledge graph, without resorting to external sources, referred to as the problem of *link prediction*. For example, if we know that "`playerA playsFor teamX`" and "`playerB teamMates playerA`," then "`playerB playsFor teamX`" may be inferred. We refer to Nickel et al. [55] for an overview of approaches.

Another active area is concerned with the *quality* of data. Nakashole and Mitchell [51] compute believability, i.e., the likelihood that a given statement is true. Dong et al. [20] evaluate the trustworthiness of sources by the correctness of their factual information. They show that trustworthiness is almost orthogonal to the popularity of the source. For example, many websites with high PageRank scores are "gossip sites," and generally are considered less reliable. Conversely, some less popular websites contain very accurate information. Zaveri et al. [94] present a survey of data quality issues for Linked Data.

References

1. Adel, H., Roth, B., Schütze, H.: Comparing convolutional neural networks to traditional models for slot filling. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL '16, pp. 828–838. Association for Computational Linguistics (2016)

2. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: Proceedings of the Fifth ACM Conference on Digital Libraries, DL '00, pp. 85–94. ACM (2000). doi: 10.1145/336597.336644
3. Allan, J.: Topic detection and tracking: Event-based information organization. Kluwer Academic Publishers (2002)
4. Almuhareb, A., Massimo, P.: Finding attributes in the web using a parser. In: Proceedings of the Corpus Linguistics Conference (2005)
5. Artiles, J., Borthwick, A., Gonzalo, J., Sekine, S., Amigó, E.: Weps-3 evaluation campaign: Overview of the web people search clustering and attribute extraction tasks. In: CLEF 2010 LABs and Workshops, Notebook Papers, 22–23 September 2010, Padua, Italy (2010)
6. Balog, K., Ramampiaro, H.: Cumulative citation recommendation: Classification vs. ranking. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, SIGIR '13, pp. 941–944 (2013). doi: 10.1145/2484028.2484151
7. Balog, K., Ramampiaro, H., Takhirov, N., Nørvåg, K.: Multi-step classification approaches to cumulative citation recommendation. In: Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, OAIR '13, pp. 121–128 (2013)
8. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'07, pp. 2670–2676. Morgan Kaufmann Publishers Inc. (2007)
9. Boschee, E., Weischedel, R., Zamanian, A.: Automatic information extraction. In: Proceedings of the International Conference on Intelligence Analysis (2005)
10. Brin, S.: Extracting patterns and relations from the world wide web. In: Selected Papers from the International Workshop on The World Wide Web and Databases, WebDB '98, pp. 172–183. Springer (1999)
11. Bunescu, R.C., Mooney, R.J.: A shortest path dependency kernel for relation extraction. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pp. 724–731. Association for Computational Linguistics (2005). doi: 10.3115/1220575.1220666
12. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: Exploring the power of tables on the web. Proc. VLDB Endow. **1**(1), 538–549 (2008). doi: 10.14778/1453856.1453916
13. Cai, Q., Yates, A.: Large-scale semantic parsing via schema matching and lexicon extension. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL '13, pp. 423–433. Association for Computational Linguistics (2013)
14. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E.R.H., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010) (2010)
15. Chinchor, N.A.: Overview of MUC-7/MET-2. In: Proceedings of the 7th Message Understanding Conference, MUC-7 (1998)
16. Del Corro, L., Gemulla, R.: Clausie: Clause-based open information extraction. In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, pp. 355–366. ACM (2013). doi: 10.1145/2488388.2488420
17. Dietz, L., Dalton, J., Balog, K.: Time-aware evaluation of cumulative citation recommendation systems. In: SIGIR 2013 Workshop on Time-aware Information Access (TAIA2013) (2013)
18. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R.: The automatic content extraction (ACE) program – tasks, data, and evaluation. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC '04. ELRA (2004)
19. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge Vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 601–610. ACM (2014). doi: 10.1145/2623330.2623623
20. Dong, X.L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., Zhang, W.: Knowledge-based trust: Estimating the trustworthiness of web sources. Proc. VLDB Endow. **8**(9), 938–949 (2015). doi: 10.14778/2777598.2777603

21. Efron, M., Willis, C., Sherman, G.: Learning sufficient queries for entity filtering. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pp. 1091–1094. ACM (2014). doi: 10.1145/2600428.2609517
22. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web-scale information extraction in knowitall: (preliminary results). In: Proceedings of the 13th International Conference on World Wide Web, WWW '04, pp. 100–110. ACM (2004). doi: 10.1145/988672.988687
23. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam, M.: Open information extraction: The second generation. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume One, IJCAI'11, pp. 3–10. AAAI Press (2011). doi: 10.5591/978-1-57735-516-8/IJCAI11-012
24. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pp. 1535–1545. Association for Computational Linguistics (2011)
25. Fleischman, M., Hovy, E.: Fine grained classification of named entities. In: Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02, pp. 1–7. Association for Computational Linguistics (2002). doi: 10.3115/1072228.1072358
26. Frank, J.R., Bauer, S.J., Kleiman-Weiner, M., Roberts, D.A., Tripuraneni, N., Zhang, C., Ré, C., Voorhees, E.M., Soboroff, I.: Evaluating stream filtering for entity profile updates for TREC 2013. In: Proceedings of The Twenty-Second Text REtrieval Conference, TREC '13 (2013)
27. Frank, J.R., Kleiman-Weiner, M., Roberts, D.A., Niu, F., Zhang, C., Ré, C., Soboroff, I.: Building an entity-centric stream filtering test collection for TREC 2012. In: The Twenty-First Text REtrieval Conference Proceedings, TREC '12 (2012)
28. Frank, J.R., Kleiman-Weiner, M., Roberts, D.A., Voorhees, E.M., Soboroff, I.: Evaluating stream filtering for entity profile updates in TREC 2012, 2013, and 2014. In: Proceedings of The Twenty-Third Text REtrieval Conference, TREC '14 (2014)
29. Galárraga, L., Razniewski, S., Amarilli, A., Suchanek, F.M.: Predicting completeness in knowledge bases. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pp. 375–383. ACM (2017). doi: 10.1145/3018661.3018739
30. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of DBpedia entities. In: Proceedings of the 11th International Conference on The Semantic Web, ISWC '12, pp. 65–81. Springer (2012)
31. Gebremeskel, G.G., He, J., de Vries, A.P., Lin, J.J.: Cumulative citation recommendation: A feature-aware comparison of approaches. In: 25th International Workshop on Database and Expert Systems Applications, DEXA '14, pp. 193–197 (2014)
32. Graus, D., Odijk, D., de Rijke, M.: The birth of collective memories: Analyzing emerging entities in text streams. arXiv preprint arXiv:1701.04039 (2017)
33. He, Y., Xin, D.: SEISA: Set expansion by iterative similarity aggregation. In: Proceedings of the 20th International Conference on World Wide Web, WWW '11. ACM (2011). doi: 10.1145/1963405.1963467
34. Hoffart, J., Altun, Y., Weikum, G.: Discovering emerging entities with ambiguous names. In: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pp. 385–396. ACM (2014). doi: 10.1145/2566486.2568003
35. Hoffmann, R., Zhang, C., Weld, D.S.: Learning 5000 relational extractors. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pp. 286–295. Association for Computational Linguistics (2010)
36. Ji, H., Grishman, R.: Knowledge base population: Successful approaches and challenges. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pp. 1148–1158. Association for Computational Linguistics (2011)
37. Ji, H., Grishman, R., Dang, H.T.: Overview of the TAC 2011 Knowledge Base Population track. In: Proceedings of the 2010 Text Analysis Conference, TAC '11. NIST (2011)
38. Jiang, J.: Information Extraction from Text, pp. 11–41. Springer (2012). doi: 10.1007/978-1-4614-3223-4_2

39. Kambhatla, N.: Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In: Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, ACLdemo '04. Association for Computational Linguistics (2004). doi: 10.3115/1219044.1219066
40. Kenter, T., Balog, K., de Rijke, M.: Evaluating document filtering systems over time. Inf. Process. Manage. **51**(6), 791–808 (2015). doi: 10.1016/j.ipm.2015.03.005
41. Li, Y., Chen, S., Zhou, Z., Yin, J., Luo, H., Hong, L., Xu, W., Chen, G., Guo, J.: PRIS at TAC2012 KBP track. In: Text Analysis Conference, TAC '12. NIST (2012)
42. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endow. **3**(1–2), 1338–1347 (2010). doi: 10.14778/1920841.1921005
43. Lin, T., Mausam, Etzioni, O.: No noun phrase left behind: Detecting and typing unlinkable entities. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pp. 893–903. Association for Computational Linguistics (2012)
44. Mausam, Schmitz, M., Bart, R., Soderland, S., Etzioni, O.: Open language learning for information extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pp. 523–534. Association for Computational Linguistics (2012)
45. McNamee, P., Dang, H.T., Simpson, H., Schone, P., Strassel, S.: An evaluation of technologies for knowledge base population. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC '10 (2010)
46. Melo, A., Paulheim, H., Völker, J.: Type prediction in RDF knowledge bases using hierarchical multilabel classification. In: Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics, WIMS '16, pp. 14:1–14:10. ACM (2016). doi: 10.1145/2912845.2912861
47. Michel, J.B., Shen, Y.K., Aiden, A.P., Veres, A., Gray, M.K., Team, T.G.B., Pickett, J.P., Holberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M.A., Aiden, E.L.: Quantitative analysis of culture using millions of digitized books. Science (2010)
48. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. 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. 1003–1011. Association for Computational Linguistics (2009)
49. Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J.: Never-ending learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15) (2015)
50. Mohapatra, H., Jain, S., Chakrabarti, S.: Joint bootstrapping of corpus annotations and entity types. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP '13, pp. 436–446. Association for Computational Linguistics (2013)
51. Nakashole, N., Mitchell, T.M.: Language-aware truth assessment of fact candidates. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL '14, pp. 1009–1019 (2014)
52. Nakashole, N., Tylenda, T., Weikum, G.: Fine-grained semantic typing of emerging entities. In: 51st Annual Meeting of the Association for Computational Linguistics, ACL '13, pp. 1488–1497. ACL (2013)
53. Nakashole, N., Weikum, G., Suchanek, F.: PATTY: a taxonomy of relational patterns with semantic types. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pp. 1135–1145. Association for Computational Linguistics (2012)

54. Nastase, V., Strube, M.: Decoding Wikipedia categories for knowledge acquisition. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2, AAAI'08, pp. 1219–1224. AAAI Press (2008)
55. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2016). doi: 10.1109/JPROC.2015.2483592
56. Paşca, M.: Organizing and searching the World Wide Web of facts – step two: Harnessing the wisdom of the crowds. In: Proceedings of the 16th International Conference on World Wide Web, WWW '07, pp. 101–110. ACM (2007a). doi: 10.1145/1242572.1242587
57. Paşca, M.: Weakly-supervised discovery of named entities using web search queries. In: Proceedings of the 16th ACM conference on Conference on information and knowledge management, CIKM '07, pp. 683–690. ACM (2007b). doi: 10.1145/1321440.1321536
58. Paşca, M., Van Durme, B.: What you seek is what you get: Extraction of class attributes from query logs. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'07, pp. 2832–2837. Morgan Kaufmann Publishers Inc. (2007)
59. Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.M., Vyas, V.: Web-scale distributional similarity and entity set expansion. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2, EMNLP '09, pp. 938–947. Association for Computational Linguistics (2009)
60. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Proceedings of the 12th International Semantic Web Conference - Part I, ISWC '13, pp. 510–525. Springer (2013). doi: 10.1007/978-3-642-41335-3_32
61. Pennacchiotti, M., Pantel, P.: Entity extraction via ensemble semantics. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1, EMNLP '09, pp. 238–247. Association for Computational Linguistics (2009)
62. Preda, N., Kasneci, G., Suchanek, F.M., Neumann, T., Yuan, W., Weikum, G.: Active knowledge: Dynamically enriching RDF knowledge bases by web services. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pp. 399–410. ACM (2010). doi: 10.1145/1807167.1807212
63. Reinanda, R., Meij, E., de Rijke, M.: Document filtering for long-tail entities. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, pp. 771–780. ACM (2016). doi: 10.1145/2983323.2983728
64. Reisinger, J., Paşca, M.: Latent variable models of concept-attribute attachment. 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. 620–628. Association for Computational Linguistics (2009)
65. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III, ECML PKDD'10, pp. 148–163. Springer (2010)
66. Robertson, S., Soboroff, I.: The TREC 2002 Filtering track report. In: Proceedings of the Eleventh Text Retrieval Conference, TREC '02 (2002)
67. Roth, B., Barth, T., Wiegand, M., Singh, M., Klakow, D.: Effective slot filling based on shallow distant supervision methods. In: Text Analysis Conference, TAC '13. NIST (2013)
68. Sarawagi, S.: Information extraction. Foundations and Trends in Databases **1**(3), 261–377 (2007). doi: 10.1561/1900000003
69. Sekine, S., Nobata, C.: Definition, dictionaries and tagger for extended named entity hierarchy. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC '04. ELRA (2004)
70. Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., Ré, C.: Incremental knowledge base construction using deepdive. Proc. VLDB Endow. **8**(11), 1310–1321 (2015). doi: 10.14778/2809974.2809991
71. Suchanek, F.M., Sozio, M., Weikum, G.: SOFIE: a self-organizing framework for information extraction. In: Proceedings of the 18th International Conference on World Wide Web, WWW '09, pp. 631–640. ACM (2009). doi: 10.1145/1526709.1526794

72. Sun, A., Grishman, R., Xu, W., Min, B.: New York University 2011 system for KBP slot filling. In: Text Analysis Conference, TAC '11. NIST (2011)
73. Surdeanu, M., Gupta, S., Bauer, J., McClosky, D., Chang, A.X., Spitkovsky, V.I., Manning, C.D.: Stanford's distantly-supervised slot-filling system. In: Text Analysis Conference, TAC '11. NIST (2011)
74. Surdeanu, M., Ji, H.: Overview of the English Slot Filling track at the TAC2014 Knowledge Base Population evaluation. In: Text Analysis Conference, TAC '14. NIST (2014)
75. Surdeanu, M., McClosky, D., Tibshirani, J., Bauer, J., Chang, A.X., Spitkovsky, V.I., Manning, C.D.: A simple distant supervision approach for the TAC-KBP slot filling task. In: Text Analysis Conference, TAC '10. NIST (2010)
76. Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C.D.: Multi-instance multi-label learning for relation extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pp. 455–465. Association for Computational Linguistics (2012)
77. Talukdar, P.P., Pereira, F.: Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pp. 1473–1481. Association for Computational Linguistics (2010)
78. Tokunaga, K., Kazama, J., Torisawa, K.: Automatic discovery of attribute words from web documents. In: Proceedings of the Second International Joint Conference on Natural Language Processing, IJCNLP '05, pp. 106–118 (2005). doi: 10.1007/11562214_10
79. Van Durme, B., Qian, T., Schubert, L.: Class-driven attribute extraction. In: Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1, COLING '08, pp. 921–928. Association for Computational Linguistics (2008)
80. Vyas, V., Pantel, P.: Semi-automatic entity set refinement. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09, pp. 290–298. Association for Computational Linguistics (2009)
81. Vyas, V., Pantel, P., Crestan, E.: Helping editors choose better seed sets for entity set expansion. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, pp. 225–234. ACM (2009). doi: 10.1145/1645953.1645984
82. Wang, C., Chakrabarti, K., He, Y., Ganjam, K., Chen, Z., Bernstein, P.A.: Concept expansion using web tables. In: Proceedings of the 24th International Conference on World Wide Web, WWW '15, pp. 1198–1208. International World Wide Web Conferences Steering Committee (2015a). doi: 10.1145/2736277.2741644
83. Wang, J., Song, D., Liao, L., Lin, C.Y.: BIT and MSRA at TREC KBA CCR track 2013. In: Proceedings of The Twenty-Second Text REtrieval Conference, TREC '13 (2013)
84. Wang, J., Song, D., Wang, Q., Zhang, Z., Si, L., Liao, L., Lin, C.Y.: An entity class-dependent discriminative mixture model for cumulative citation recommendation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pp. 635–644. ACM (2015b). doi: 10.1145/2766462.2767698
85. Wang, R.C., Cohen, W.W.: Iterative set expansion of named entities using the web. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ICDM '08, pp. 1091–1096. IEEE Computer Society (2008). doi: 10.1109/ICDM.2008.145
86. Wang, R.C., Cohen, W.W.: Automatic set instance extraction using the web. 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 1 - Volume 1, ACL '09, pp. 441–449. Association for Computational Linguistics (2009)
87. Weikum, G., Hoffart, J., Suchanek, F.: Ten years of knowledge harvesting: Lessons and challenges. IEEE Data Eng. Bull. **39**(3), 41–50 (2016)
88. West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., Lin, D.: Knowledge base completion via search-based question answering. In: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pp. 515–526. ACM (2014). doi: 10.1145/2566486.2568032

89. Wu, F., Weld, D.S.: Autonomously semantifying Wikipedia. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pp. 41–50. ACM (2007). doi: 10.1145/1321440.1321449
90. Wu, F., Weld, D.S.: Open information extraction using Wikipedia. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pp. 118–127. Association for Computational Linguistics (2010)
91. Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12, pp. 97–108. ACM (2012). doi: 10.1145/2213836.2213848
92. Yosef, M.A., Bauer, S., Spaniol, J.H.M., Weikum, G.: HYENA: Hierarchical type classification for entity names. In: Proceedings of COLING 2012, pp. 1361–1370 (2012)
93. Yu, D., Li, H., Cassidy, T., Li, Q., Huang, H., Chen, Z., Ji, H., Zhang, Y., Roth, D.: RPI-BLENDER TAC-KBP2013 knowledge base population system. In: Text Analysis Conference, TAC '13. NIST (2013)
94. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: A survey. Semantic Web **7**(1), 63–93 (2016). doi: 10.3233/SW-150175
95. Zhao, S., Grishman, R.: Extracting relations with integrated information using kernel methods. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05, pp. 419–426. Association for Computational Linguistics (2005). doi: 10.3115/1219840.1219892
96. Zhou, G., Su, J., Zhang, J., Zhang, M.: Exploring various knowledge in relation extraction. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05, pp. 427–434. Association for Computational Linguistics (2005). doi: 10.3115/1219840.1219893
97. Zhou, M., Chang, K.C.C.: Entity-centric document filtering: Boosting feature mapping through meta-features. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM '13, pp. 119–128. ACM (2013). doi: 10.1145/2505515.2505683