Chapter 17

Information extraction

Computers offer powerful capabilities for searching and reasoning about structured records and relational data. Some have argued that the most important limitation of artificial intelligence is not inference or learning, but simply having too little knowledge (Lenat et al., 1990). Natural language processing provides an appealing solution: automatically construct a structured **knowledge base** by reading natural language text.

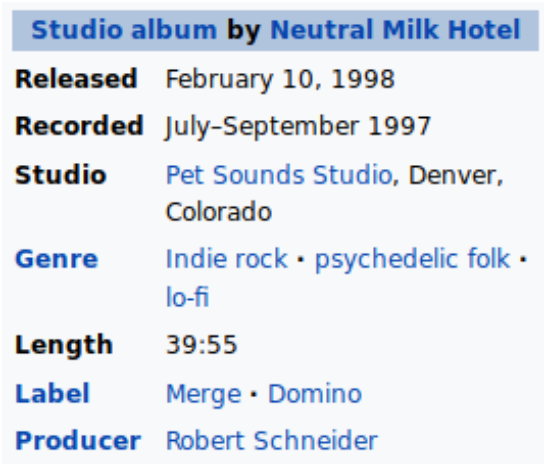
For example, many Wikipedia pages have an "infobox" that provides structured information about an entity or event. An example is shown in Figure 17.1a: each row represents one or more properties of the entity IN THE AEROPLANE OVER THE SEA, a record album. The set of properties is determined by a predefined **schema**, which applies to all record albums in Wikipedia. As shown in Figure 17.1b, the values for many of these fields are indicated directly in the first few sentences of text on the same Wikipedia page.

The task of automatically constructing (or "populating") an infobox from text is an example of **information extraction**. Much of information extraction can be described in terms of **entities**, **relations**, and **events**.

- **Entities** are uniquely specified objects in the world, such as people (JEFF MANGUM), places (ATHENS, GEORGIA), organizations (MERGE RECORDS), and times (FEBRUARY 10, 1998). Chapter 8 described the task of **named entity recognition**, which labels tokens as parts of entity spans. Now we will see how to go further, **linking** each entity **mention** to an element in a **knowledge base**.

- **Relations** include a **predicate** and two **arguments**: for example, CAPITAL(GEORGIA, ATLANTA).

- **Events** involve multiple typed arguments. For example, the production and release



Studio album by Neutral Milk Hotel	
Released	February 10, 1998
Recorded	July–September 1997
Studio	Pet Sounds Studio, Denver, Colorado
Genre	Indie rock • psychedelic folk • lo-fi
Length	39:55
Label	Merge • Domino
Producer	Robert Schneider

(a) A Wikipedia infobox

(17.1) In the Aeroplane Over the Sea is the second and final studio album by the American indie rock band Neutral Milk Hotel.

(17.2) It was released in the United States on February 10, 1998 on Merge Records and May 1998 on Blue Rose Records in the United Kingdom.

(17.3) Jeff Mangum moved from Athens, Georgia to Denver, Colorado to prepare the bulk of the album's material with producer Robert Schneider, this time at Schneider's newly created Pet Sounds Studio at the home of Jim McIntyre.

(b) The first few sentences of text. Strings that match fields or field names in the infobox are underlined; strings that mention other entities are wavy underlined.

Figure 17.1: From the Wikipedia page for the album "In the Aeroplane Over the Sea", retrieved October 26, 2017.

of the album described in Figure 17.1 is described by the event,

$$\langle \text{TITLE} : \text{IN THE AEROPLANE OVER THE SEA}, \\ \text{ARTIST} : \text{NEUTRAL MILK HOTEL}, \\ \text{RELEASE-DATE} : \text{1998-FEB-10}, \ldots \rangle$$

The set of arguments for an event type is defined by a **schema**. Events often refer to time-delimited occurrences: weddings, protests, purchases, terrorist attacks.

Information extraction is similar to semantic role labeling (chapter 13): we may think of predicates as corresponding to events, and the arguments as defining slots in the event representation. However, the goals of information extraction are different. Rather than accurately parsing every sentence, information extraction systems often focus on recognizing a few key relation or event types, or on the task of identifying all properties of a given entity. Information extraction is often evaluated by the correctness of the resulting knowledge base, and not by how many sentences were accurately parsed. The goal is sometimes described as **macro-reading**, as opposed to **micro-reading**, in which each sentence must be analyzed correctly. Macro-reading systems are not penalized for ignoring difficult sentences, as long as they can recover the same information from other, easier-to-read sources. However, macro-reading systems must resolve apparent inconsistencies

(was the album released on MERGE RECORDS or BLUE ROSE RECORDS?), requiring reasoning across the entire dataset.

In addition to the basic tasks of recognizing entities, relations, and events, information extraction systems must handle negation, and must be able to distinguish statements of fact from hopes, fears, hunches, and hypotheticals. Finally, information extraction is often paired with the problem of **question answering**, which requires accurately parsing a query, and then selecting or generating a textual answer. Question answering systems can be built on knowledge bases that are extracted from large text corpora, or may attempt to identify answers directly from the source texts.

17.1 Entities

The starting point for information extraction is to identify mentions of entities in text. Consider the following example:

(17.4) *The United States Army captured a hill overlooking Atlanta on May 14, 1864.*

For this sentence, there are two goals:

1. *Identify* the spans *United States Army*, *Atlanta*, and *May 14, 1864* as entity mentions. (The hill is not uniquely identified, so it is not a *named* entity.) We may also want to recognize the **named entity types**: organization, location, and date. This is **named entity recognition**, and is described in chapter 8.
2. *Link* these spans to entities in a knowledge base: U.S. ARMY, ATLANTA, and 1864-MAY-14. This task is known as **entity linking**.

The strings to be linked to entities are **mentions** — similar to the use of this term in coreference resolution. In some formulations of the entity linking task, only named entities are candidates for linking. This is sometimes called **named entity linking** (Ling et al., 2015). In other formulations, such as **Wikification** (Milne and Witten, 2008), any string can be a mention. The set of target entities often corresponds to Wikipedia pages, and Wikipedia is the basis for more comprehensive knowledge bases such as YAGO (Suchanek et al., 2007), DBPedia (Auer et al., 2007), and Freebase (Bollacker et al., 2008). Entity linking may also be performed in more "closed" settings, where a much smaller list of targets is provided in advance. The system must also determine if a mention does not refer to any entity in the knowledge base, sometimes called a **NIL entity** (McNamee and Dang, 2009).

Returning to (17.4), the three entity mentions may seem unambiguous. But the Wikipedia disambiguation page for the string *Atlanta* says otherwise:[1] there are more than twenty

[1] `https://en.wikipedia.org/wiki/Atlanta_(disambiguation)`, retrieved November 1, 2017.

different towns and cities, five United States Navy vessels, a magazine, a television show, a band, and a singer — each prominent enough to have its own Wikipedia page. We now consider how to choose among these dozens of possibilities. In this chapter we will focus on supervised approaches. Unsupervised entity linking is closely related to the problem of **cross-document coreference resolution**, where the task is to identify pairs of mentions that corefer, across document boundaries (Bagga and Baldwin, 1998b; Singh et al., 2011).

17.1.1 Entity linking by learning to rank

Entity linking is often formulated as a **ranking** problem,

$$\hat{y} = \underset{y \in \mathcal{Y}(\boldsymbol{x})}{\text{argmax}} \, \Psi(y, \boldsymbol{x}, \boldsymbol{c}), \tag{17.1}$$

where y is a target entity, $\boldsymbol{x}$ is a description of the mention, $\mathcal{Y}(\boldsymbol{x})$ is a set of candidate entities, and $\boldsymbol{c}$ is a description of the context — such as the other text in the document, or its metadata. The function Ψ is a scoring function, which could be a linear model, $\Psi(y, \boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{\theta} \cdot \boldsymbol{f}(y, \boldsymbol{x}, \boldsymbol{c})$, or a more complex function such as a neural network. In either case, the scoring function can be learned by minimizing a margin-based **ranking loss**,

$$\ell(\hat{y}, y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{c}^{(i)}) = \Big(\Psi(\hat{y}, \boldsymbol{x}^{(i)}, \boldsymbol{c}^{(i)}) - \Psi(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{c}^{(i)}) + 1\Big)_+, \tag{17.2}$$

where $y^{(i)}$ is the ground truth and $\hat{y} \neq y^{(i)}$ is the predicted target for mention $\boldsymbol{x}^{(i)}$ in context $\boldsymbol{c}^{(i)}$ (Joachims, 2002; Dredze et al., 2010).

Candidate identification For computational tractability, it is helpful to restrict the set of candidates, $\mathcal{Y}(\boldsymbol{x})$. One approach is to use a **name dictionary**, which maps from strings to the entities that they might mention. This mapping is many-to-many: a string such as *Atlanta* can refer to multiple entities, and conversely, an entity such as ATLANTA can be referenced by multiple strings. A name dictionary can be extracted from Wikipedia, with links between each Wikipedia entity page and the anchor text of all hyperlinks that point to the page (Bunescu and Pasca, 2006; Ratinov et al., 2011). To improve recall, the name dictionary can be augmented by partial and approximate matching (Dredze et al., 2010), but as the set of candidates grows, the risk of false positives increases. For example, the string *Atlanta* is a partial match to *the Atlanta Fed* (a name for the FEDERAL RESERVE BANK OF ATLANTA), and a noisy match (edit distance of one) from *Atalanta* (a heroine in Greek mythology and an Italian soccer team).

Features Feature-based approaches to entity ranking rely on three main types of local information (Dredze et al., 2010):

- The similarity of the mention string to the canonical entity name, as quantified by string similarity. This feature would elevate the city ATLANTA over the basketball team ATLANTA HAWKS for the string *Atlanta*.
- The popularity of the entity, which can be measured by Wikipedia page views or PageRank in the Wikipedia link graph. This feature would elevate ATLANTA, GEORGIA over the unincorporated community of ATLANTA, OHIO.
- The entity type, as output by the named entity recognition system. This feature would elevate the city of ATLANTA over the magazine ATLANTA in contexts where the mention is tagged as a location.

In addition to these local features, the document context can also help. If *Jamaica* is mentioned in a document about the Caribbean, it is likely to refer to the island nation; in the context of New York, it is likely to refer to the neighborhood in Queens; in the context of a menu, it might refer to a hibiscus tea beverage. Such hints can be formalized by computing the similarity between the Wikipedia page describing each candidate entity and the mention context $\boldsymbol{c}^{(i)}$, which may include the bag-of-words representing the document (Dredze et al., 2010; Hoffart et al., 2011) or a smaller window of text around the mention (Ratinov et al., 2011). For example, we can compute the cosine similarity between bag-of-words vectors for the context and entity description, typically weighted using **inverse document frequency** to emphasize rare words.[2]

Neural entity linking An alternative approach is to compute the score for each entity candidate using distributed vector representations of the entities, mentions, and context. For example, for the task of entity linking in Twitter, Yang et al. (2016) employ the bilinear scoring function,

$$\Psi(y, \boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{v}_y^\top \boldsymbol{\Theta}^{(y,x)} \boldsymbol{x} + \boldsymbol{v}_y^\top \boldsymbol{\Theta}^{(y,c)} \boldsymbol{c}, \quad [17.3]$$

with $\boldsymbol{v}_y \in \mathbb{R}^{K_y}$ as the vector embedding of entity y, $\boldsymbol{x} \in \mathbb{R}^{K_x}$ as the embedding of the mention, $\boldsymbol{c} \in \mathbb{R}^{K_c}$ as the embedding of the context, and the matrices $\boldsymbol{\Theta}^{(y,x)}$ and $\boldsymbol{\Theta}^{(y,c)}$ as parameters that score the compatibility of each entity with respect to the mention and context. Each of the vector embeddings can be learned from an end-to-end objective, or pre-trained on unlabeled data.

- Pretrained **entity embeddings** can be obtained from an existing knowledge base (Bordes et al., 2011, 2013), or by running a word embedding algorithm such as WORD2VEC

[2]The **document frequency** of word j is $\text{DF}(j) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(x_j^{(i)} > 0\right)$, equal to the number of documents in which the word appears. The contribution of each word to the cosine similarity of two bag-of-words vectors can be weighted by the **inverse document frequency** $\frac{1}{\text{DF}(j)}$ or $\log \frac{1}{\text{DF}(j)}$, to emphasize rare words (Spärck Jones, 1972).

on the text of Wikipedia, with hyperlinks substituted for the anchor text.[3]

- The embedding of the mention $\boldsymbol{x}$ can be computed by averaging the embeddings of the words in the mention (Yang et al., 2016), or by the compositional techniques described in § 14.8.
- The embedding of the context $\boldsymbol{c}$ can also be computed from the embeddings of the words in the context. A **denoising autoencoder** learns a function from raw text to dense K-dimensional vector encodings by minimizing a reconstruction loss (Vincent et al., 2010),

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_h} \sum_{i=1}^{N} ||\boldsymbol{x}^{(i)} - g(h(\tilde{\boldsymbol{x}}^{(i)}; \boldsymbol{\theta}_h); \boldsymbol{\theta}_g)||^2, \quad [17.4]$$

where $\tilde{\boldsymbol{x}}^{(i)}$ is a noisy version of the bag-of-words counts $\boldsymbol{x}^{(i)}$, which is produced by randomly setting some counts to zero; $h : \mathbb{R}^V \rightarrow \mathbb{R}^K$ is an encoder with parameters $\boldsymbol{\theta}_h$; and $g : \mathbb{R}^K \rightarrow \mathbb{R}^V$, with parameters $\boldsymbol{\theta}_g$. The encoder and decoder functions are typically implemented as feedforward neural networks. To apply this model to entity linking, each entity and context are initially represented by the encoding of their bag-of-words vectors, $h(\boldsymbol{e})$ and $g(\boldsymbol{c})$, and these encodings are then fine-tuned from labeled data (He et al., 2013). The context vector $\boldsymbol{c}$ can also be obtained by convolution (§ 3.4) on the embeddings of words in the document (Sun et al., 2015), or by examining metadata such as the author's social network (Yang et al., 2016).

The remaining parameters $\boldsymbol{\Theta}^{(y,x)}$ and $\boldsymbol{\Theta}^{(y,c)}$ can be trained by backpropagation from the margin loss in Equation 17.2.

17.1.2 Collective entity linking

Entity linking can be more accurate when it is performed jointly across a document. To see why, consider the following lists:

(17.5)
a. California, Oregon, Washington
b. Baltimore, Washington, Philadelphia
c. Washington, Adams, Jefferson

In each case, the term *Washington* refers to a different entity, and this reference is strongly suggested by the other entries on the list. In the last list, all three names are highly ambiguous — there are dozens of other *Adams* and *Jefferson* entities in Wikipedia. But a

[3] Pre-trained entity embeddings can be downloaded from `https://code.google.com/archive/p/word2vec/`.

preference for coherence motivates **collectively** linking these references to the first three U.S. presidents.

A general approach to collective entity linking is to introduce a compatibility score $\psi_c(\boldsymbol{y})$. Collective entity linking is then performed by optimizing the global objective,

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in \mathbb{Y}(\boldsymbol{x})}{\text{argmax}} \Psi_c(\boldsymbol{y}) + \sum_{i=1}^{N} \Psi_\ell(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{c}^{(i)}), \quad [17.5]$$

where $\mathbb{Y}(\boldsymbol{x})$ is the set of all possible collective entity assignments for the mentions in $\boldsymbol{x}$, and ψ_ℓ is the local scoring function for each entity i. The compatibility function is typically decomposed into a sum of pairwise scores, $\Psi_c(\boldsymbol{y}) = \sum_{i=1}^{N} \sum_{j \neq i}^{N} \Psi_c(y^{(i)}, y^{(j)})$. These scores can be computed in a number of different ways:

- Wikipedia defines high-level categories for entities (e.g., *living people*, *Presidents of the United States*, *States of the United States*), and Ψ_c can reward entity pairs for the number of categories that they have in common (Cucerzan, 2007).
- Compatibility can be measured by the number of incoming hyperlinks shared by the Wikipedia pages for the two entities (Milne and Witten, 2008).
- In a neural architecture, the compatibility of two entities can be set equal to the inner product of their embeddings, $\Psi_c(y^{(i)}, y^{(j)}) = \boldsymbol{v}_{y^{(i)}} \cdot \boldsymbol{v}_{y^{(j)}}$.
- A non-pairwise compatibility score can be defined using a type of latent variable model known as a **probabilistic topic model** (Blei et al., 2003; Blei, 2012). In this framework, each latent topic is a probability distribution over entities, and each document has a probability distribution over topics. Each entity helps to determine the document's distribution over topics, and in turn these topics help to resolve ambiguous entity mentions (Newman et al., 2006). Inference can be performed using the sampling techniques described in chapter 5.

Unfortunately, collective entity linking is **NP-hard** even for pairwise compatibility functions, so exact optimization is almost certainly intractable. Various approximate inference techniques have been proposed, including **integer linear programming** (Cheng and Roth, 2013), **Gibbs sampling** (Han and Sun, 2012), and graph-based algorithms (Hoffart et al., 2011; Han et al., 2011).

17.1.3 *Pairwise ranking loss functions

The loss function defined in Equation 17.2 considers only the highest-scoring prediction $\hat{y}$, but in fact, the true entity $y^{(i)}$ should outscore *all* other entities. A loss function based on this idea would give a gradient against the features or representations of several entities,

Algorithm 18 WARP approximate ranking loss

procedure WARP($y^{(i)}, \boldsymbol{x}^{(i)}$)
  $N \leftarrow 0$
  repeat
    Randomly sample $y \sim \mathcal{Y}(\boldsymbol{x}^{(i)})$
    $N \leftarrow N + 1$
    if $\psi(y, \boldsymbol{x}^{(i)}) + 1 > \psi(y^{(i)}, \boldsymbol{x}^{(i)})$ **then** ▷ check for margin violation
      $r \leftarrow \left\lfloor |\mathcal{Y}(\boldsymbol{x}^{(i)})|/N \right\rfloor$ ▷ compute approximate rank
      return $L_{\text{rank}}(r) \times (\psi(y, \boldsymbol{x}^{(i)}) + 1 - \psi(y^{(i)}, \boldsymbol{x}^{(i)}))$
  until $N \geq |\mathcal{Y}(\boldsymbol{x}^{(i)})| - 1$ ▷ no violation found
  return 0 ▷ return zero loss

not just the top-scoring prediction. Usunier et al. (2009) define a general ranking error function,

$$L_{\text{rank}}(k) = \sum_{j=1}^{k} \alpha_j, \quad \text{with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0, \tag{17.6}$$

where k is equal to the number of labels ranked higher than the correct label $y^{(i)}$. This function defines a class of ranking errors: if $\alpha_j = 1$ for all j, then the ranking error is equal to the rank of the correct entity; if $\alpha_1 = 1$ and $\alpha_{j>1} = 0$, then the ranking error is one whenever the correct entity is not ranked first; if α_j decreases smoothly with j, as in $\alpha_j = \frac{1}{j}$, then the error is between these two extremes.

This ranking error can be integrated into a margin objective. Remember that large margin classification requires not only the correct label, but also that the correct label outscores other labels by a substantial margin. A similar principle applies to ranking: we want a high rank for the correct entity, and we want it to be separated from other entities by a substantial margin. We therefore define the margin-augmented rank,

$$r(y^{(i)}, \boldsymbol{x}^{(i)}) \triangleq \sum_{y \in \mathcal{Y}(\boldsymbol{x}^{(i)}) \setminus y^{(i)}} \delta\left(1 + \psi(y, \boldsymbol{x}^{(i)}) \geq \psi(y^{(i)}, \boldsymbol{x}^{(i)})\right), \tag{17.7}$$

where $\delta(\cdot)$ is a delta function, and $\mathcal{Y}(\boldsymbol{x}^{(i)}) \setminus y^{(i)}$ is the set of all entity candidates minus the true entity $y^{(i)}$. The margin-augmented rank is the rank of the true entity, after augmenting every other candidate with a margin of one, under the current scoring function ψ. (The context $\boldsymbol{c}$ is omitted for clarity, and can be considered part of $\boldsymbol{x}$.)

For each instance, a hinge loss is computed from the ranking error associated with this

margin-augmented rank, and the violation of the margin constraint,

$$\ell(y^{(i)}, \boldsymbol{x}^{(i)}) = \frac{L_{\text{rank}}(r(y^{(i)}, \boldsymbol{x}^{(i)}))}{r(y^{(i)}, \boldsymbol{x}^{(i)})} \sum_{y \in \mathcal{Y}(\boldsymbol{x}) \setminus y^{(i)}} \Big(\psi(y, \boldsymbol{x}^{(i)}) - \psi(y^{(i)}, \boldsymbol{x}^{(i)}) + 1\Big)_+, \quad [17.8]$$

The sum in Equation 17.8 includes non-zero values for every label that is ranked at least as high as the true entity, after applying the margin augmentation. Dividing by the margin-augmented rank of the true entity thus gives the average violation.

The objective in Equation 17.8 is expensive to optimize when the label space is large, as is usually the case for entity linking against large knowledge bases. This motivates a randomized approximation called **WARP** (Weston et al., 2011), shown in Algorithm 18. In this procedure, we sample random entities until one violates the pairwise margin constraint, $\psi(y, \boldsymbol{x}^{(i)}) + 1 \geq \psi(y^{(i)}, \boldsymbol{x}^{(i)})$. The number of samples N required to find such a violation yields an approximation of the margin-augmented rank of the true entity, $r(y^{(i)}, \boldsymbol{x}^{(i)}) \approx \left\lfloor \frac{|\mathcal{Y}(\boldsymbol{x})|}{N} \right\rfloor$. If a violation is found immediately, $N = 1$, the correct entity probably ranks below many others, $r \approx |\mathcal{Y}(\boldsymbol{x})|$. If many samples are required before a violation is found, $N \rightarrow |\mathcal{Y}(\boldsymbol{x})|$, then the correct entity is probably highly ranked, $r \rightarrow 1$. A computational advantage of WARP is that it is not necessary to find the highest-scoring label, which can impose a non-trivial computational cost when $\mathcal{Y}(\boldsymbol{x}^{(i)})$ is large. The objective is conceptually similar to the **negative sampling** objective in WORD2VEC (chapter 14), which compares the observed word against randomly sampled alternatives.

17.2 Relations

After identifying the entities that are mentioned in a text, the next step is to determine how they are related. Consider the following example:

(17.6) George Bush traveled to France on Thursday for a summit.

This sentence introduces a relation between the entities referenced by *George Bush* and *France*. In the Automatic Content Extraction (ACE) ontology (Linguistic Data Consortium, 2005), the type of this relation is PHYSICAL, and the subtype is LOCATED. This relation would be written,

$$\text{PHYSICAL.LOCATED}(\text{GEORGE BUSH}, \text{FRANCE}). \quad [17.9]$$

Relations take exactly two arguments, and the order of the arguments matters.

In the ACE datasets, relations are annotated between entity mentions, as in the example above. Relations can also hold between nominals, as in the following example from the SemEval-2010 shared task (Hendrickx et al., 2009):

CAUSE-EFFECT	*those cancers were caused by radiation exposures*
INSTRUMENT-AGENCY	*phone operator*
PRODUCT-PRODUCER	*a factory manufactures suits*
CONTENT-CONTAINER	*a bottle of honey was weighed*
ENTITY-ORIGIN	*letters from foreign countries*
ENTITY-DESTINATION	*the boy went to bed*
COMPONENT-WHOLE	*my apartment has a large kitchen*
MEMBER-COLLECTION	*there are many trees in the forest*
COMMUNICATION-TOPIC	*the lecture was about semantics*

Table 17.1: Relations and example sentences from the SemEval-2010 dataset (Hendrickx et al., 2009)

(17.7) The cup contained tea from dried ginseng.

This sentence describes a relation of type ENTITY-ORIGIN between *tea* and *ginseng*. Nominal relation extraction is closely related to **semantic role labeling** (chapter 13). The main difference is that relation extraction is restricted to a relatively small number of relation types; for example, Table 17.1 shows the ten relation types from SemEval-2010.

17.2.1 Pattern-based relation extraction

Early work on relation extraction focused on hand-crafted patterns (Hearst, 1992). For example, the appositive *Starbuck, a native of Nantucket* signals the relation ENTITY-ORIGIN between *Starbuck* and *Nantucket*. This pattern can be written as,

$$\text{PERSON , } \textit{a native of} \text{ LOCATION} \Rightarrow \text{ENTITY-ORIGIN}(\text{PERSON}, \text{LOCATION}). \quad [17.10]$$

This pattern will be "triggered" whenever the literal string *, a native of* occurs between an entity of type PERSON and an entity of type LOCATION. Such patterns can be generalized beyond literal matches using techniques such as lemmatization, which would enable the words (*buy*, *buys*, *buying*) to trigger the same patterns (see § 4.3.1). A more aggressive strategy would be to group all words in a WordNet synset (§ 4.2), so that, e.g., *buy* and *purchase* trigger the same patterns.

Relation extraction patterns can be implemented in finite-state automata (§ 9.1). If the named entity recognizer is also a finite-state machine, then the systems can be combined by finite-state transduction (Hobbs et al., 1997). This makes it possible to propagate uncertainty through the finite-state cascade, and disambiguate from higher-level context. For example, suppose the entity recognizer cannot decide whether *Starbuck* refers to either a PERSON or a LOCATION; in the composed transducer, the relation extractor would be free to select the PERSON annotation when it appears in the context of an appropriate pattern.

17.2.2 Relation extraction as a classification task

Relation extraction can be formulated as a classification problem,

$$\hat{r}_{(i,j),(m,n)} = \underset{r \in \mathcal{R}}{\text{argmax}}\, \Psi(r, (i,j), (m,n), \boldsymbol{w}), \quad [17.11]$$

where $r \in \mathcal{R}$ is a relation type (possibly NIL), $\boldsymbol{w}_{i+1:j}$ is the span of the first argument, and $\boldsymbol{w}_{m+1:n}$ is the span of the second argument. The argument $\boldsymbol{w}_{m+1:n}$ may appear before or after $\boldsymbol{w}_{i+1:j}$ in the text, or they may overlap; we stipulate only that $\boldsymbol{w}_{i+1:j}$ is the first argument of the relation. We now consider three alternatives for computing the scoring function.

Feature-based classification

In a feature-based classifier, the scoring function is defined as,

$$\Psi(r, (i,j), (m,n), \boldsymbol{w}) = \boldsymbol{\theta} \cdot \boldsymbol{f}(r, (i,j), (m,n), \boldsymbol{w}), \quad [17.12]$$

with $\boldsymbol{\theta}$ representing a vector of weights, and $\boldsymbol{f}(\cdot)$ a vector of features. The pattern-based methods described in § 17.2.1 suggest several features:

- Local features of $\boldsymbol{w}_{i+1:j}$ and $\boldsymbol{w}_{m+1:n}$, including: the strings themselves; whether they are recognized as entities, and if so, which type; whether the strings are present in a **gazetteer** of entity names; each string's syntactic head (§ 9.2.2).
- Features of the span between the two arguments, $\boldsymbol{w}_{j+1:m}$ or $\boldsymbol{w}_{n+1:i}$ (depending on which argument appears first): the length of the span; the specific words that appear in the span, either as a literal sequence or a bag-of-words; the wordnet synsets (§ 4.2) that appear in the span between the arguments.
- Features of the syntactic relationship between the two arguments, typically the **dependency path** between the arguments (§ 13.2.1). Example dependency paths are shown in Table 17.2.

Kernels

Suppose that the first line of Table 17.2 is a labeled example, and the remaining lines are instances to be classified. A feature-based approach would have to decompose the dependency paths into features that capture individual edges, with or without their labels, and then learn weights for each of these features: for example, the second line contains identical dependencies, but different arguments; the third line contains a different inflection of the word *travel*; the fourth and fifth lines each contain an additional edge on the dependency path; and the sixth example uses an entirely different path. Rather than attempting to create local features that capture all of the ways in which these dependencies paths

1.	***George Bush*** *traveled to* ***France***	*George Bush* $\underset{\text{NSUBJ}}{\leftarrow}$ *traveled* $\underset{\text{OBL}}{\rightarrow}$ *France*
2.	***Ahab*** *traveled to* ***Nantucket***	*Ahab* $\underset{\text{NSUBJ}}{\leftarrow}$ *traveled* $\underset{\text{OBL}}{\rightarrow}$ *Nantucket*
3.	***George Bush*** *will travel to* ***France***	*George Bush* $\underset{\text{NSUBJ}}{\leftarrow}$ *travel* $\underset{\text{OBL}}{\rightarrow}$ *France*
4.	***George Bush*** *wants to travel to* ***France***	*George Bush* $\underset{\text{NSUBJ}}{\leftarrow}$ *wants* $\underset{\text{XCOMP}}{\rightarrow}$ *travel* $\underset{\text{OBL}}{\rightarrow}$ *France*
5.	***Ahab*** *traveled to a city in* ***France***	*Ahab* $\underset{\text{NSUBJ}}{\leftarrow}$ *traveled* $\underset{\text{OBL}}{\rightarrow}$ *city* $\underset{\text{NMOD}}{\rightarrow}$ *France*
6.	*We await* ***Ahab*** *'s visit to France*	*Ahab* $\underset{\text{NMOD:POSS}}{\leftarrow}$ *visit* $\underset{\text{NMOD}}{\rightarrow}$ *France*

Table 17.2: Candidates instances for the PHYSICAL.LOCATED relation, and their dependency paths

are similar and different, we can instead define a similarity function κ, which computes a score for any pair of instances, $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$. The score for any pair of instances (i, j) is $\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) \geq 0$, with $\kappa(i, j)$ being large when instances $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$ are similar. If the function κ obeys a few key properties it is a valid **kernel function**.[4]

Given a valid kernel function, we can build a non-linear classifier without explicitly defining a feature vector or neural network architecture. For a binary classification problem $y \in \{-1, 1\}$, we have the decision function,

$$\hat{y} = \text{Sign}(b + \sum_{i=1}^{N} y^{(i)} \alpha^{(i)} \kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x})) \quad [17.13]$$

where b and $\{\alpha^{(i)}\}_{i=1}^{N}$ are parameters that must be learned from the training set, under the constraint $\forall_i, \alpha^{(i)} \geq 0$. Intuitively, each α_i specifies the importance of the instance $\boldsymbol{x}^{(i)}$ towards the classification rule. Kernel-based classification can be viewed as a weighted form of the **nearest-neighbor** classifier (Hastie et al., 2009), in which test instances are assigned the most common label among their near neighbors in the training set. This results in a non-linear classification boundary. The parameters are typically learned from a margin-based objective (see § 2.4), leading to the **kernel support vector machine**. To generalize to multi-class classification, we can train separate binary classifiers for each label (sometimes called **one-versus-all**), or train binary classifiers for each pair of possible labels (**one-versus-one**).

Dependency kernels are particularly effective for relation extraction, due to their ability to capture syntactic properties of the path between the two candidate arguments. One class of dependency tree kernels is defined recursively, with the score for a pair of trees

[4] The **Gram matrix** $\mathbf{K}$ arises from computing the kernel function between all pairs in a set of instances. For a valid kernel, the Gram matrix must be symmetric ($\mathbf{K} = \mathbf{K}^\top$) and positive semi-definite ($\forall \boldsymbol{a}, \boldsymbol{a}^\top \mathbf{K} \boldsymbol{a} \geq 0$). For more on kernel-based classification, see chapter 14 of Murphy (2012).

equal to the similarity of the root nodes and the sum of similarities of matched pairs of child subtrees (Zelenko et al., 2003; Culotta and Sorensen, 2004). Alternatively, Bunescu and Mooney (2005) define a kernel function over sequences of unlabeled dependency edges, in which the score is computed as a product of scores for each pair of words in the sequence: identical words receive a high score, words that share a synset or part-of-speech receive a small non-zero score (e.g., *travel / visit*), and unrelated words receive a score of zero.

Neural relation extraction

Convolutional neural networks (§ 3.4) were an early neural architecture for relation extraction (Zeng et al., 2014; dos Santos et al., 2015). For the sentence $(w_1, w_2, \ldots, w_M)$, obtain a matrix of word embeddings $\mathbf{X}$, where $\boldsymbol{x}_m \in \mathbb{R}^K$ is the embedding of w_m. Now, suppose the candidate arguments appear at positions a_1 and a_2; then for each word in the sentence, its position with respect to each argument is $m - a_1$ and $m - a_2$. (Following Zeng et al. (2014), this is a restricted version of the relation extraction task in which the arguments are single tokens.) To capture any information conveyed by these positions, the word embeddings are concatenated with vector encodings of the positional offsets, $\boldsymbol{x}^{(p)}_{m-a_1}$ and $\boldsymbol{x}^{(p)}_{m-a_2}$. (For more on **positional encodings**, see § 18.3.2.) The complete base representation of the sentence is,

$$\mathbf{X}(a_1, a_2) = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_M \\ \boldsymbol{x}^{(p)}_{1-a_1} & \boldsymbol{x}^{(p)}_{2-a_1} & \cdots & \boldsymbol{x}^{(p)}_{M-a_1} \\ \boldsymbol{x}^{(p)}_{1-a_2} & \boldsymbol{x}^{(p)}_{2-a_2} & \cdots & \boldsymbol{x}^{(p)}_{M-a_2} \end{pmatrix}, \tag{17.14}$$

where each column is a vertical concatenation of a word embedding, represented by the column vector $\boldsymbol{x}_m$, and two positional encodings, specifying the position with respect to a_1 and a_2. The matrix $\mathbf{X}(a_1, a_2)$ is then taken as input to a convolutional layer (see § 3.4), and max-pooling is applied to obtain a vector. The final scoring function is then,

$$\Psi(r, i, j, \mathbf{X}) = \boldsymbol{\theta}_r \cdot \text{MaxPool}(\text{ConvNet}(\mathbf{X}(i, j); \boldsymbol{\phi})), \tag{17.15}$$

where $\boldsymbol{\phi}$ defines the parameters of the convolutional operator, and the $\boldsymbol{\theta}_r$ defines a set of weights for relation r. The model can be trained using a margin objective,

$$\hat{r} = \underset{r}{\text{argmax}}\, \Psi(r, i, j, \mathbf{X}) \tag{17.16}$$

$$\ell = (1 + \psi(\hat{r}, i, j, \mathbf{X}) - \psi(r, i, j, \mathbf{X}))_+. \tag{17.17}$$

Recurrent neural networks (§ 6.3) have also been applied to relation extraction, using a network such as a bidirectional LSTM to encode the words or dependency path between the two arguments. Xu et al. (2015) segment each dependency path into left and

right subpaths: the path *George Bush* $\underset{\text{NSUBJ}}{\leftarrow}$ *wants* $\underset{\text{XCOMP}}{\rightarrow}$ *travel* $\underset{\text{OBL}}{\rightarrow}$ *France* is segmented into the subpaths, *George Bush* $\underset{\text{NSUBJ}}{\leftarrow}$ *wants* and *wants* $\underset{\text{XCOMP}}{\rightarrow}$ *travel* $\underset{\text{OBL}}{\rightarrow}$ *France*. In each path, a recurrent neural network is run from the argument to the root word (in this case, *wants*). The final representation by max pooling (§ 3.4) across all the recurrent states along each path. This process can be applied across separate "channels", in which the inputs consist of embeddings for the words, parts-of-speech, dependency relations, and WordNet hypernyms (e.g., *France-nation*; see § 4.2). To define the model formally, let $s(m)$ define the successor of word m in either the left or right subpath (in a dependency path, each word can have a successor in at most one subpath). Let $\boldsymbol{x}_m^{(c)}$ indicate the embedding of word (or relation) m in channel c, and let $\overleftarrow{\boldsymbol{h}}_m^{(c)}$ and $\overrightarrow{\boldsymbol{h}}_m^{(c)}$ indicate the associated recurrent states in the left and right subtrees respectively. Then the complete model is specified as follows,

$$\boldsymbol{h}_{s(m)}^{(c)} = \text{RNN}(\boldsymbol{x}_{s(m)}^{(c)}, \boldsymbol{h}_m^{(c)}) \quad [17.18]$$

$$\boldsymbol{z}^{(c)} = \text{MaxPool}\left(\overleftarrow{\boldsymbol{h}}_i^{(c)}, \overleftarrow{\boldsymbol{h}}_{s(i)}^{(c)}, \ldots, \overleftarrow{\boldsymbol{h}}_{\text{root}}^{(c)}, \overrightarrow{\boldsymbol{h}}_j^{(c)}, \overrightarrow{\boldsymbol{h}}_{s(j)}^{(c)}, \ldots, \overrightarrow{\boldsymbol{h}}_{\text{root}}^{(c)}\right) \quad [17.19]$$

$$\Psi(r, i, j) = \boldsymbol{\theta} \cdot \left[\boldsymbol{z}^{(\text{word})}; \boldsymbol{z}^{(\text{POS})}; \boldsymbol{z}^{(\text{dependency})}; \boldsymbol{z}^{(\text{hypernym})}\right]. \quad [17.20]$$

Note that $\boldsymbol{z}$ is computed by applying max-pooling to the *matrix* of horizontally concatenated vectors $\boldsymbol{h}$, while Ψ is computed from the *vector* of vertically concatenated vectors $\boldsymbol{z}$. Xu et al. (2015) pass the score Ψ through a **softmax** layer to obtain a probability $\text{p}(r \mid i, j, \boldsymbol{w})$, and train the model by regularized **cross-entropy**. Miwa and Bansal (2016) show that a related model can solve the more challenging "end-to-end" relation extraction task, in which the model must simultaneously detect entities and then extract their relations.

17.2.3 Knowledge base population

In many applications, what matters is not what fraction of sentences are analyzed correctly, but how much accurate knowledge can be extracted. **Knowledge base population (KBP)** refers to the task of filling in Wikipedia-style infoboxes, as shown in Figure 17.1a. Knowledge base population can be decomposed into two subtasks: **entity linking** (described in § 17.1), and **slot filling** (Ji and Grishman, 2011). Slot filling has two key differences from the formulation of relation extraction presented above: the relations hold between entities rather than spans of text, and the performance is evaluated at the *type level* (on entity pairs), rather than on the *token level* (on individual sentences).

From a practical standpoint, there are three other important differences between slot filling and per-sentence relation extraction.

- KBP tasks are often formulated from the perspective of identifying attributes of a few "query" entities. As a result, these systems often start with an **information**

retrieval phase, in which relevant passages of text are obtained by search.

- For many entity pairs, there will be multiple passages of text that provide evidence. Slot filling systems must aggregate this evidence to predict a single relation type (or set of relations).
- Labeled data is usually available in the form of pairs of related entities, rather than annotated passages of text. Training from such type-level annotations is a challenge: two entities may be linked by several relations, or they may appear together in a passage of text that nonetheless does not describe their relation to each other.

Information retrieval is beyond the scope of this text (see Manning et al., 2008). The remainder of this section describes approaches to information fusion and learning from type-level annotations.

Information fusion

In knowledge base population, there will often be multiple pieces of evidence for (and sometimes against) a single relation. For example, a search for the entity MAYNARD JACKSON, JR. may return several passages that reference the entity ATLANTA:[5]

(17.8) a. Elected mayor of **Atlanta** in 1973, **Maynard Jackson** was the first African American to serve as mayor of a major southern city.

b. **Atlanta**'s airport will be renamed to honor **Maynard Jackson**, the city's first Black mayor.

c. Born in Dallas, Texas in 1938, **Maynard Holbrook Jackson, Jr.** moved to **Atlanta** when he was 8.

d. **Maynard Jackson** has gone from one of the worst high schools in **Atlanta** to one of the best.

The first and second examples provide evidence for the relation MAYOR holding between the entities ATLANTA and MAYNARD JACKSON, JR.. The third example provides evidence for a different relation between these same entities, LIVED-IN. The fourth example poses an entity linking problem, referring to MAYNARD JACKSON HIGH SCHOOL. Knowledge base population requires aggregating this sort of textual evidence, and predicting the relations that are most likely to hold.

One approach is to run a single-document relation extraction system (using the techniques described in § 17.2.2), and then aggregate the results (Li et al., 2011). Relations

[5] First three examples from: http://www.georgiaencyclopedia.org/articles/government-politics/maynard-jackson-1938-2003; JET magazine, November 10, 2003; www.todayingeorgiahistory.org/content/maynard-jackson-elected

that are detected with high confidence in multiple documents are more likely to be valid, motivating the heuristic,

$$\psi(r, e_1, e_2) = \sum_{i=1}^{N} (\text{p}(r(e_1, e_2) \mid \boldsymbol{w}^{(i)}))^{\alpha}, \quad [17.21]$$

where $\text{p}(r(e_1, e_2) \mid \boldsymbol{w}^{(i)})$ is the probability of relation r between entities e_1 and e_2 conditioned on the text $\boldsymbol{w}^{(i)}$, and $\alpha \gg 1$ is a tunable hyperparameter. Using this heuristic, it is possible to rank all candidate relations, and trace out a **precision-recall curve** as more relations are extracted.[6] Alternatively, features can be aggregated across multiple passages of text, feeding a single type-level relation extraction system (Wolfe et al., 2017).

Precision can be improved by introducing constraints across multiple relations. For example, if we are certain of the relation $\text{PARENT}(e_1, e_2)$, then it cannot also be the case that $\text{PARENT}(e_2, e_1)$. Integer linear programming makes it possible to incorporate such constraints into a global optimization (Li et al., 2011). Other pairs of relations have positive correlations, such $\text{MAYOR}(e_1, e_2)$ and $\text{LIVED-IN}(e_1, e_2)$. Compatibility across relation types can be incorporated into probabilistic graphical models (e.g., Riedel et al., 2010).

Distant supervision

Relation extraction is "annotation hungry," because each relation requires its own labeled data. Rather than relying on annotations of individual documents, it would be preferable to use existing knowledge resources — such as the many facts that are already captured in knowledge bases like DBPedia. However such annotations raise the inverse of the information fusion problem considered above: the existence of the relation $\text{MAYOR}(\text{MAYNARD JACKSON JR.}, \text{ATLANTA})$ provides only **distant supervision** for the example texts in which this entity pair is mentioned.

One approach is to treat the entity pair as the instance, rather than the text itself (Mintz et al., 2009). Features are then aggregated across all sentences in which both entities are mentioned, and labels correspond to the relation (if any) between the entities in a knowledge base, such as FreeBase. Negative instances are constructed from entity pairs that are not related in the knowledge base. In some cases, two entities are related, but the knowledge base is missing the relation; however, because the number of possible entity pairs is huge, these missing relations are presumed to be relatively rare. This approach is shown in Figure 17.2.

In **multiple instance learning**, labels are assigned to *sets* of instances, of which only an unknown subset are actually relevant (Dietterich et al., 1997; Maron and Lozano-Pérez, 1998). This formalizes the framework of distant supervision: the relation $\text{REL}(\text{A}, \text{B})$ acts

[6] The precision-recall curve is similar to the ROC curve shown in Figure 4.4, but it includes the precision $\frac{\text{TP}}{\text{TP+FP}}$ rather than the false positive rate $\frac{\text{FP}}{\text{FP+TN}}$.

- **Label** : MAYOR(ATLANTA, MAYNARD JACKSON)
  - Elected mayor of **Atlanta** in 1973, **Maynard Jackson** ...
  - **Atlanta**'s airport will be renamed to honor **Maynard Jackson**, the city's first Black mayor
  - Born in Dallas, Texas in 1938, **Maynard Holbrook Jackson, Jr.** moved to **Atlanta** when he was 8.
- **Label** : MAYOR(NEW YORK, FIORELLO LA GUARDIA)
  - **Fiorello La Guardia** was Mayor of **New York** for three terms ...
  - **Fiorello La Guardia**, then serving on the **New York** City Board of Aldermen...
- **Label** : BORN-IN(DALLAS, MAYNARD JACKSON)
  - Born in **Dallas**, Texas in 1938, **Maynard Holbrook Jackson, Jr.** moved to Atlanta when he was 8.
  - **Maynard Jackson** was raised in **Dallas** ...
- **Label** : NIL(NEW YORK, MAYNARD JACKSON)
  - **Jackson** married Valerie Richardson, whom he had met in **New York**...
  - **Jackson** was a member of the Georgia and **New York** bars ...

Figure 17.2: Four training instances for relation classification using **distant supervision** Mintz et al. (2009). The first two instances are positive for the MAYOR relation, and the third instance is positive for the BORN-IN relation. The fourth instance is a negative example, constructed from a pair of entities (NEW YORK, MAYNARD JACKSON) that do not appear in any Freebase relation. Each instance's features are computed by aggregating across all sentences in which the two entities are mentioned.

as a label for the entire set of sentences mentioning entities A and B, even when only a subset of these sentences actually describes the relation. One approach to multi-instance learning is to introduce a binary **latent variable** for each sentence, indicating whether the sentence expresses the labeled relation (Riedel et al., 2010). A variety of inference techniques have been employed for this probabilistic model of relation extraction: Surdeanu et al. (2012) use expectation maximization, Riedel et al. (2010) use sampling, and Hoffmann et al. (2011) use a custom graph-based algorithm. Expectation maximization and sampling are surveyed in chapter 5, and are covered in more detail by Murphy (2012); graph-based methods are surveyed by Mihalcea and Radev (2011).

17.2.4 Open information extraction

In classical relation extraction, the set of relations is defined in advance, using a **schema**. The relation for any pair of entities can then be predicted using multi-class classification. In **open information extraction** (OpenIE), a relation can be any triple of text. The example

Task	Relation ontology	Supervision
PropBank semantic role labeling	VerbNet	sentence
FrameNet semantic role labeling	FrameNet	sentence
Relation extraction	ACE, TAC, SemEval, etc	sentence
Slot filling	ACE, TAC, SemEval, etc	relation
Open Information Extraction	open	seed relations or patterns

Table 17.3: Various relation extraction tasks and their properties. VerbNet and FrameNet are described in chapter 13. ACE (Linguistic Data Consortium, 2005), TAC (McNamee and Dang, 2009), and SemEval (Hendrickx et al., 2009) refer to shared tasks, each of which involves an ontology of relation types.

sentence (17.8a) instantiates several "relations" of this sort, e.g.,

- (*mayor of*, *Maynard Jackson*, *Atlanta*),
- (*elected*, *Maynard Jackson*, *mayor of Atlanta*),
- (*elected in*, *Maynard Jackson*, *1973*).

Extracting such tuples can be viewed as a lightweight version of **semantic role labeling** (chapter 13), with only two argument types: first slot and second slot. The task is generally evaluated on the relation level, rather than on the level of sentences: precision is measured by the number of extracted relations that are accurate, and recall is measured by the number of true relations that were successfully extracted. OpenIE systems are trained from distant supervision or bootstrapping, rather than from labeled sentences.

An early example is the TEXTRUNNER system (Banko et al., 2007), which identifies relations with a set of handcrafted syntactic rules. The examples that are acquired from the handcrafted rules are then used to train a classification model that uses part-of-speech patterns as features. Finally, the relations that are extracted by the classifier are aggregated, removing redundant relations and computing the number of times that each relation is mentioned in the corpus. TEXTRUNNER was the first in a series of systems that performed increasingly accurate open relation extraction by incorporating more precise linguistic features (Etzioni et al., 2011), distant supervision from Wikipedia infoboxes (Wu and Weld, 2010), and better learning algorithms (Zhu et al., 2009).

17.3 Events

Relations link pairs of entities, but many real-world situations involve more than two entities. Consider again the example sentence (17.8a), which describes the event of an election,

with four properties: the office (MAYOR), the district (ATLANTA), the date (1973), and the person elected (MAYNARD JACKSON, JR.). In **event detection**, a schema is provided for each event type (e.g., an election, a terrorist attack, or a chemical reaction), indicating all the possible properties of the event. The system is then required to fill in as many of these properties as possible (Doddington et al., 2004).

Event detection systems generally involve a retrieval component (finding relevant documents and passages of text) and an extraction component (determining the properties of the event based on the retrieved texts). Early approaches focused on finite-state patterns for identify event properties (Hobbs et al., 1997); such patterns can be automatically induced by searching for patterns that are especially likely to appear in documents that match the event query (Riloff, 1996). Contemporary approaches employ techniques that are similar to FrameNet semantic role labeling (§ 13.2), such as structured prediction over local and global features (Li et al., 2013) and bidirectional recurrent neural networks (Feng et al., 2016). These methods detect whether an event is described in a sentence, and if so, what are its properties.

Event coreference Because multiple sentences may describe unique properties of a single event, **event coreference** is required to link event mentions across a single passage of text, or between passages (Humphreys et al., 1997). Bejan and Harabagiu (2014) define event coreference as the task of identifying event mentions that share the same event participants (i.e., the slot-filling entities) and the same event properties (e.g., the time and location), within or across documents. Event coreference resolution can be performed using supervised learning techniques in a similar way to entity coreference, as described in chapter 15: move left-to-right through the document, and use a classifier to decide whether to link each event reference to an existing cluster of coreferent events, or to create a new cluster (Ahn, 2006). Each clustering decision is based on the compatibility of features describing the participants and properties of the event. Due to the difficulty of annotating large amounts of data for entity coreference, unsupervised approaches are especially desirable (Chen and Ji, 2009; Bejan and Harabagiu, 2014).

Relations between events Just as entities are related to other entities, events may be related to other events: for example, the event of winning an election both *precedes* and *causes* the event of serving as mayor; moving to Atlanta *precedes* and *enables* the event of becoming mayor of Atlanta; moving from Dallas to Atlanta *prevents* the event of later becoming mayor of Dallas. As these examples show, events may be related both temporally and causally. The **TimeML** annotation scheme specifies a set of six temporal relations between events (Pustejovsky et al., 2005), derived in part from **interval algebra** (Allen, 1984). The TimeBank corpus provides TimeML annotations for 186 documents (Pustejovsky et al., 2003). Methods for detecting these temporal relations combine supervised

	Positive (+)	Negative (-)	Underspecified (u)
Certain (CT)	Fact: CT+	Counterfact: CT-	Certain, but unknown: CTU
Probable (PR)	Probable: PR+	Not probable: PR-	(NA)
Possible (PS)	Possible: PS+	Not possible: PS-	(NA)
Underspecified (U)	(NA)	(NA)	Unknown or uncommitted: UU

Table 17.4: Table of factuality values from the FactBank corpus (Saurí and Pustejovsky, 2009). The entry (NA) indicates that this combination is not annotated.

machine learning with temporal constraints, such as transitivity (e.g. Mani et al., 2006; Chambers and Jurafsky, 2008).

More recent annotation schemes and datasets combine temporal and causal relations (Mirza et al., 2014; Dunietz et al., 2017): for example, the CaTeRS dataset includes annotations of 320 five-sentence short stories (Mostafazadeh et al., 2016). Abstracting still further, **processes** are networks of causal relations between multiple events. A small dataset of biological processes is annotated in the ProcessBank dataset (Berant et al., 2014), with the goal of supporting automatic question answering on scientific textbooks.

17.4 Hedges, denials, and hypotheticals

The methods described thus far apply to **propositions** about the way things are in the real world. But natural language can also describe events and relations that are likely or unlikely, possible or impossible, desired or feared. The following examples hint at the scope of the problem (Prabhakaran et al., 2010):

(17.9) a. GM will lay off workers.
b. A spokesman for GM said GM will lay off workers.
c. GM may lay off workers.
d. The politician claimed that GM will lay off workers.
e. Some wish GM would lay off workers.
f. Will GM lay off workers?
g. Many wonder whether GM will lay off workers.

Accurate information extraction requires handling these **extra-propositional** aspects of meaning, which are sometimes summarized under the terms **modality** and **negation**.[7]

[7]The classification of negation as extra-propositional is controversial: Packard et al. (2014) argue that negation is a "core part of compositionally constructed logical-form representations." Negation is an element of the semantic parsing tasks discussed in chapter 12 and chapter 13 — for example, negation markers are

Modality refers to expressions of the speaker's attitude towards her own statements, including "degree of certainty, reliability, subjectivity, sources of information, and perspective" (Morante and Sporleder, 2012). Various systematizations of modality have been proposed (e.g., Palmer, 2001), including categories such as future, interrogative, imperative, conditional, and subjective. Information extraction is particularly concerned with negation and certainty. For example, Saurí and Pustejovsky (2009) link negation with a modal calculus of certainty, likelihood, and possibility, creating the two-dimensional schema shown in Table 17.4. This is the basis for the FactBank corpus, with annotations of the **factuality** of all sentences in 208 documents of news text.

A related concept is **hedging**, in which speakers limit their commitment to a proposition (Lakoff, 1973):

(17.10) a. These results **suggest** that expression of c-jun, jun B and jun D genes **might** be involved in terminal granulocyte differentiation... (Morante and Daelemans, 2009)

b. A whale is **technically** a mammal (Lakoff, 1973)

In the first example, the hedges *suggest* and *might* communicate uncertainty; in the second example, there is no uncertainty, but the hedge *technically* indicates that the evidence for the proposition will not fully meet the reader's expectations. Hedging has been studied extensively in scientific texts (Medlock and Briscoe, 2007; Morante and Daelemans, 2009), where the goal of large-scale extraction of scientific facts is obstructed by hedges and speculation. Still another related aspect of modality is **evidentiality**, in which speakers mark the source of their information. In many languages, it is obligatory to mark evidentiality through affixes or particles (Aikhenvald, 2004); while evidentiality is not grammaticalized in English, authors are expected to express this information in contexts such as journalism (Kovach and Rosenstiel, 2014) and Wikipedia.[8]

Methods for handling negation and modality generally include two phases:

1. detecting negated or uncertain events;

2. identifying **scope** of the negation or modal operator.

A considerable body of work on negation has employed rule-based techniques such as regular expressions (Chapman et al., 2001) to detect negated events. Such techniques

treated as adjuncts in PropBank semantic role labeling. However, many of the relation extraction methods mentioned in this chapter do not handle negation directly. A further consideration is that negation interacts closely with aspects of modality that are generally not considered in propositional semantics, such as certainty and subjectivity.

[8]https://en.wikipedia.org/wiki/Wikipedia:Verifiability

match lexical cues (e.g., *Norwood was **not** elected Mayor*), while avoiding "double negatives" (e.g., *surely all this is **not without** meaning*). Supervised techniques involve classifiers over lexical and syntactic features (Uzuner et al., 2009) and sequence labeling (Prabhakaran et al., 2010).

The scope refers to the elements of the text whose propositional meaning is negated or modulated (Huddleston and Pullum, 2005), as elucidated in the following example from Morante and Sporleder (2012):

(17.11) [After his habit he said] **nothing**, and after mine I asked no questions.
After his habit he said nothing, and [after mine I asked] **no** [questions].

In this sentence, there are two negation cues (*nothing* and *no*). Each negates an event, indicated by the underlined verbs *said* and *asked*, and each occurs within a scope: *after his habit he said* and *after mine I asked _____ questions*. Scope identification is typically formalized as sequence labeling problems, with each word token labeled as beginning, inside, or outside of a cue, focus, or scope span (see § 8.3). Conventional sequence labeling approaches can then be applied, using surface features as well as syntax (Velldal et al., 2012) and semantic analysis (Packard et al., 2014). Labeled datasets include the BioScope corpus of biomedical texts (Vincze et al., 2008) and a shared task dataset of detective stories by Arthur Conan Doyle (Morante and Blanco, 2012).

17.5 Question answering and machine reading

The victory of the Watson question-answering system against three top human players on the game show *Jeopardy!* was a landmark moment for natural language processing (Ferrucci et al., 2010). Game show questions are usually answered by **factoids**: entity names and short phrases.[9] The task of factoid question answering is therefore closely related to information extraction, with the additional problem of accurately parsing the question.

17.5.1 Formal semantics

Semantic parsing is an effective method for question-answering in restricted domains such as questions about geography and airline reservations (Zettlemoyer and Collins, 2005), and has also been applied in "open-domain" settings such as question answering on Freebase (Berant et al., 2013) and biomedical research abstracts (Poon and Domingos, 2009). One approach is to convert the question into a lambda calculus expression that returns a boolean value: for example, the question *who is the mayor of the capital of Georgia?*

[9] The broader landscape of question answering includes "why" questions (*Why did Ahab continue to pursue the white whale?*), "how questions" (*How did Queequeg die?*), and requests for summaries (*What was Ishmael's attitude towards organized religion?*). For more, see Hirschman and Gaizauskas (2001).

would be converted to,

$$\lambda x.\exists y \text{ CAPITAL}(\text{GEORGIA}, y) \wedge \text{MAYOR}(y, x). \quad [17.22]$$

This lambda expression can then be used to query an existing knowledge base, returning "true" for all entities that satisfy it.

17.5.2 Machine reading

Recent work has focused on answering questions about specific textual passages, similar to the reading comprehension examinations for young students (Hirschman et al., 1999). This task has come to be known as **machine reading**.

Datasets

The machine reading problem can be formulated in a number of different ways. The most important distinction is what form the answer should take.

- **Multiple-choice question answering**, as in the MCTest dataset of stories (Richardson et al., 2013) and the New York Regents Science Exams (Clark, 2015). In MCTest, the answer is deducible from the text alone, while in the science exams, the system must make inferences using an existing model of the underlying scientific phenomena. Here is an example from MCTest:

  (17.12) James the turtle was always getting into trouble. Sometimes he'd reach into the freezer and empty out all the food ...

  Q: What is the name of the trouble making turtle?
  (a) Fries
  (b) Pudding
  (c) James
  (d) Jane

- **Cloze**-style "fill in the blank" questions, as in the CNN/Daily Mail comprehension task (Hermann et al., 2015), the Children's Book Test (Hill et al., 2016), and the Who-did-What dataset (Onishi et al., 2016). In these tasks, the system must guess which word or entity completes a sentence, based on reading a passage of text. Here is an example from Who-did-What:

  (17.13) Q: Tottenham manager Juande Ramos has hinted he will allow _____ to leave if the Bulgaria striker makes it clear he is unhappy. (Onishi et al., 2016)

  The query sentence may be selected either from the story itself, or from an external summary. In either case, datasets can be created automatically by processing large

quantities existing documents. An additional constraint is that that missing element from the cloze must appear in the main passage of text: for example, in Who-did-What, the candidates include all entities mentioned in the main passage. In the CNN/Daily Mail dataset, each entity name is replaced by a unique identifier, e.g., ENTITY37. This ensures that correct answers can only be obtained by accurately reading the text, and not from external knowledge about the entities.

- **Extractive** question answering, in which the answer is drawn from the original text. In WikiQA, answers are sentences (Yang et al., 2015). In the Stanford Question Answering Dataset (SQuAD), answers are words or short phrases (Rajpurkar et al., 2016):

  (17.14) In metereology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
  Q: What causes precipitation to fall? A: gravity

  In both WikiQA and SQuAD, the original texts are Wikipedia articles, and the questions are generated by crowdworkers.

Methods

A baseline method is to search the text for sentences or short passages that overlap with both the query and the candidate answer (Richardson et al., 2013). In example (17.12), this baseline would select the correct answer, since *James* appears in a sentence that includes the query terms *trouble* and *turtle*.

This baseline can be implemented as a neural architecture, using an **attention mechanism** (see § 18.3.1), which scores the similarity of the query to each part of the source text (Chen et al., 2016). The first step is to encode the passage $\boldsymbol{w}^{(p)}$ and the query $\boldsymbol{w}^{(q)}$, using two bidirectional LSTMs (§ 7.6).

$$\boldsymbol{h}^{(q)} = \text{BiLSTM}(\boldsymbol{w}^{(q)}; \Theta^{(q)}) \quad [17.23]$$

$$\boldsymbol{h}^{(p)} = \text{BiLSTM}(\boldsymbol{w}^{(p)}; \Theta^{(p)}). \quad [17.24]$$

The query is represented by vertically concatenating the final states of the left-to-right and right-to-left passes:

$$\boldsymbol{u} = [\overrightarrow{\boldsymbol{h}^{(q)}}_{M_q}; \overleftarrow{\boldsymbol{h}^{(q)}}_{0}]. \quad [17.25]$$

The attention vector is computed as a softmax over a vector of bilinear products, and the expected representation is computed by summing over attention values,

$$\tilde{\alpha}_m = (\boldsymbol{u}^{(q)})^\top \mathbf{W}_a \boldsymbol{h}_m^{(p)} \quad [17.26]$$

$$\boldsymbol{\alpha} = \text{SoftMax}(\tilde{\boldsymbol{\alpha}}) \quad [17.27]$$

$$\boldsymbol{o} = \sum_{m=1}^{M} \alpha_m \boldsymbol{h}_m^{(p)}. \quad [17.28]$$

Each candidate answer c is represented by a vector $\boldsymbol{x}_c$. Assuming the candidate answers are spans from the original text, these vectors can be set equal to the corresponding element in $\boldsymbol{h}^{(p)}$. The score for each candidate answer a is computed by the inner product,

$$\hat{c} = \underset{c}{\text{argmax}}\, \boldsymbol{o} \cdot \boldsymbol{x}_c. \quad [17.29]$$

This architecture can be trained end-to-end from a loss based on the log-likelihood of the correct answer. A number of related architectures have been proposed (e.g., Hermann et al., 2015; Kadlec et al., 2016; Dhingra et al., 2017; Cui et al., 2017), and these methods are surveyed by Wang et al. (2017).

Additional resources

The field of information extraction is surveyed in course notes by Grishman (2012), and more recently in a short survey paper (Grishman, 2015). Shen et al. (2015) survey the task of entity linking, and Ji and Grishman (2011) survey work on knowledge base population. This chapter's discussion of non-propositional meaning was strongly influenced by Morante and Sporleder (2012), who introduced a special issue of the journal *Computational Linguistics* dedicated to recent work on modality and negation.

Exercises

1. Go to the Wikipedia page for your favorite movie. For each record in the info box (e.g., *Screenplay by: Stanley Kubrick*), report whether there is a sentence in the article containing both the field and value (e.g., *The screenplay was written by Stanley Kubrick*). If not, is there is a sentence in the article containing just the value? (For records with more than one value, just use the first value.)

2. Building on your answer in the previous question, report the dependency path between the head words of the field and value for at least three records.

3. Consider the following heuristic for entity linking:

- Among all entities that have the same type as the mention (e.g., LOC, PER), choose the one whose name has the lowest edit distance from the mention.
- If more than one entity has the right type and the lowest edit distance from the mention, choose the most popular one.
- If no candidate entity has the right type, choose NIL.

Now suppose you have the following feature function:

$$\boldsymbol{f}(y, \boldsymbol{x}) = [\text{edit-dist}(\text{name}(y), \boldsymbol{x}), \text{same-type}(y, \boldsymbol{x}), \text{popularity}(y), \delta(y = \text{NIL})]$$

Design a set of ranking weights $\boldsymbol{\theta}$ that match the heuristic. You may assume that edit distance and popularity are always in the range $[0, 100]$, and that the NIL entity has values of zero for all features except $\delta(y = \text{NIL})$.

4. Now consider another heuristic:

   - Among all candidate entities that have edit distance zero from the mention, and are the right type, choose the most popular one.
   - If no entity has edit distance zero from the mention, choose the one with the right type that is most popular, regardless of edit distance.
   - If no entity has the right type, choose NIL.

   Using the same features and assumptions from the previous problem, prove that there is no set of weights that could implement this heuristic. Then show that the heuristic can be implemented by adding a single feature. Your new feature should consider only the edit distance.

5. Download the Reuters corpus in NLTK, and iterate over the tokens in the corpus:

```
import nltk
nltk.corpus.download('reuters')
from nltk.corpus import reuters
for word in reuters.words():
 #your code here
```

   a) Apply the pattern _____, *such as* _____ to obtain candidates for the IS-A relation, e.g. IS-A(ROMANIA, COUNTRY). What are three pairs that this method identifies correctly? What are three different pairs that it gets wrong?

   b) Design a pattern for the PRESIDENT relation, e.g. PRESIDENT(PHILIPPINES, CORAZON AQUINO In this case, you may want to augment your pattern matcher with the ability to match multiple token wildcards, perhaps using case information to detect proper names. Again, list three correct

c) Preprocess the Reuters data by running a named entity recognizer, replacing tokens with named entity spans when applicable — e.g., your pattern can now match on *the United States* if the NER system tags it. Apply your PRESIDENT matcher to this preprocessed data. Does the accuracy improve? Compare 20 randomly-selected pairs from this pattern and the one you designed in the previous part.

6. Using the same NLTK Reuters corpus, apply distant supervision to build a training set for detecting the relation between nations and their capitals. Start with the following known relations: (JAPAN, TOKYO), (FRANCE, PARIS), (ITALY, ROME). How many positive and negative examples are you able to extract?

7. Represent the dependency path $\boldsymbol{x}^{(i)}$ as a sequence of words and dependency arcs of length M_i, ignoring the endpoints of the path. In example 1 of Table 17.2, the dependency path is,

$$\boldsymbol{x}^{(1)} = (\underset{\text{NSUBJ}}{\leftarrow}, \textit{traveled}, \underset{\text{OBL}}{\rightarrow}) \quad [17.30]$$

If $x_m^{(i)}$ is a word, then let $\text{pos}(x_m^{(i)})$ be its part-of-speech, using the tagset defined in chapter 8.

We can define the following kernel function over pairs of dependency paths (Bunescu and Mooney, 2005):

$$\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \begin{cases} 0, & M_i \neq M_j \\ \prod_{m=1}^{M_i} c(x_m^{(i)}, x_m^{(j)}), & M_i = M_j \end{cases}$$

$$c(x_m^{(i)}, x_m^{(j)}) = \begin{cases} 2, & x_m^{(i)} = x_m^{(j)} \\ 1, & x_m^{(i)} \neq x_m^{(j)} \text{ and } \text{pos}(x_m^{(i)}) = \text{pos}(x_m^{(j)}) \\ 0, & \text{otherwise.} \end{cases}$$

Using this kernel function, compute the kernel similarities of example 1 from Table 17.2 with the other five examples.

8. Continuing from the previous problem, suppose that the instances have the following labels:

$$y_2 = 1, y_3 = -1, y_4 = -1, y_5 = 1, y_6 = 1 \quad [17.31]$$

Equation 17.13 defines a kernel-based classification in terms of parameters α and b. Using the above labels for $y_2, \ldots, y_6$, identify the values of α and b under which $\hat{y}_1 = 1$. Remember the constraint that $\alpha_i \geq 0$ for all i.

9. Consider the neural QA system described in § 17.5.2, but restrict the set of candidate answers to words in the passage, and set each candidate answer embedding $\boldsymbol{x}$ equal to the vector $\boldsymbol{h}_m^{(p)}$, representing token m in the passage, so that $\hat{m} = \text{argmax}_m \, \boldsymbol{o} \cdot \boldsymbol{h}_m^{(p)}$. Suppose the system selects answer $\hat{m}$, but the correct answer is m^*. Consider the gradient of the margin loss with respect to the attention:

   a) Prove that $\frac{\partial \ell}{\partial \alpha_{\hat{m}}} \geq \frac{\partial \ell}{\partial \alpha_{m*}}$.

   b) Assuming that $||\boldsymbol{h}_{\hat{m}}|| = ||\boldsymbol{h}_{m^*}||$, prove that $\frac{\partial \ell}{\partial \alpha_{\hat{m}}} \geq 0$ and $\frac{\partial \ell}{\partial \alpha_{m^*}} \leq 0$. Explain in words what this means about how the attention is expected to change after a gradient-based update.