CHAPTER

19 Information Extraction: Relations, Events, and Time

Time will explain.
Jane Austen, *Persuasion*

Imagine that you are an analyst with an investment firm that tracks airline stocks. You're given the task of determining the relationship (if any) between airline announcements of fare increases and the behavior of their stocks the next day. Historical data about stock prices is easy to come by, but what about the airline announcements? You will need to know at least the name of the airline, the nature of the proposed fare hike, the dates of the announcement, and possibly the response of other airlines. Fortunately, these can be all found in news articles like this one:

> Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

information extraction

This chapter presents techniques for extracting limited kinds of semantic content from text. This process of **information extraction** (IE) turns the unstructured information embedded in texts into structured data, for example for populating a relational database to enable further processing.

relation extraction

We begin with the task of **relation extraction**: finding and classifying semantic relations among entities mentioned in a text, like child-of (X is the child-of Y), or part-whole or geospatial relations. Relation extraction has close links to populating a relational database, and **knowledge graphs**, datasets of structured relational knowledge, are a useful way for search engines to present information to users.

knowledge graphs

event extraction

Next, we discuss **event extraction**, the task of finding events in which these entities participate, like, in our sample text, the fare increases by *United* and *American* and the reporting events *said* and *cite*. Events are also situated in **time**, occurring at a particular date or time, and events can be related temporally, happening before or after or simultaneously with each other. We'll need to recognize temporal expressions like *Friday*, *Thursday* or *two days from now* and times such as *3:30 P.M.*, and **normalize** them onto specific calendar dates or times. We'll need to link *Friday* to the time of United's announcement, *Thursday* to the previous day's fare increase, and we'll need to produce a timeline in which United's announcement follows the fare increase and American's announcement follows both of those events.

template filling

The related task of **template filling** is to find recurring stereotypical events or situations in documents and fill in the template slots. These slot-fillers may consist of text segments extracted directly from the text, or concepts like times, amounts, or ontology entities that have been inferred through additional processing. Our airline

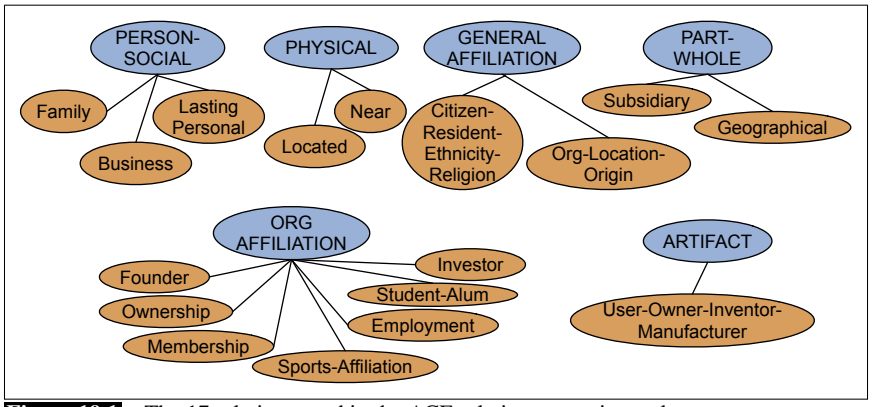

Figure 19.1 The 17 relations used in the ACE relation extraction task.

text presents such a stereotypical situation since airlines often raise fares and then wait to see if competitors follow along. Here we can identify *United* as a lead airline that initially raised its fares, $6 as the amount, *Thursday* as the increase date, and *American* as an airline that followed along, leading to a filled template like the following:

$$\text{FARE-RAISE ATTEMPT:} \begin{bmatrix} \text{LEAD AIRLINE:} & \text{UNITED AIRLINES} \\ \text{AMOUNT:} & \$6 \\ \text{EFFECTIVE DATE:} & 2006\text{-}10\text{-}26 \\ \text{FOLLOWER:} & \text{AMERICAN AIRLINES} \end{bmatrix}$$

19.1 Relation Extraction

Let's assume that we have detected the named entities in our sample text (perhaps using the techniques of Chapter 8), and would like to discern the relationships that exist among the detected entities:

> Citing high fuel prices, [$_{\text{ORG}}$ **United Airlines**] said [$_{\text{TIME}}$ **Friday**] it has increased fares by [$_{\text{MONEY}}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{\text{ORG}}$ **American Airlines**], a unit of [$_{\text{ORG}}$ **AMR Corp.**], immediately matched the move, spokesman [$_{\text{PER}}$ **Tim Wagner**] said. [$_{\text{ORG}}$ **United**], a unit of [$_{\text{ORG}}$ **UAL Corp.**], said the increase took effect [$_{\text{TIME}}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{\text{LOC}}$ **Chicago**] to [$_{\text{LOC}}$ **Dallas**] and [$_{\text{LOC}}$ **Denver**] to [$_{\text{LOC}}$ **San Francisco**].

The text tells us, for example, that *Tim Wagner* is a spokesman for *American Airlines*, that *United* is a unit of *UAL Corp.*, and that *American* is a unit of *AMR*. These binary relations are instances of more generic relations such as **part-of** or **employs** that are fairly frequent in news-style texts. Figure 19.1 lists the 17 relations used in the ACE relation extraction evaluations and Fig. 19.2 shows some sample relations. We might also extract more domain-specific relation such as the notion of an airline route. For example from this text we can conclude that United has routes to Chicago, Dallas, Denver, and San Francisco.

Relations	Types	Examples
Physical-Located	PER-GPE	**He** was in **Tennessee**
Part-Whole-Subsidiary	ORG-ORG	**XYZ**, the parent company of **ABC**
Person-Social-Family	PER-PER	**Yoko**'s husband **John**
Org-AFF-Founder	PER-ORG	**Steve Jobs**, co-founder of **Apple**...

Figure 19.2 Semantic relations with examples and the named entity types they involve.

Sets of relations have been defined for many other domains as well. For example UMLS, the Unified Medical Language System from the US National Library of Medicine has a network that defines 134 broad subject categories, entity types, and 54 relations between the entities, such as the following:

Entity	Relation	Entity
Injury	disrupts	Physiological Function
Bodily Location	location-of	Biologic Function
Anatomical Structure	part-of	Organism
Pharmacologic Substance	causes	Pathological Function
Pharmacologic Substance	treats	Pathologic Function

Given a medical sentence like this one:

(19.1) Doppler echocardiography can be used to diagnose left anterior descending artery stenosis in patients with type 2 diabetes

We could thus extract the UMLS relation:

Echocardiography, Doppler Diagnoses *Acquired stenosis*

infoboxes Wikipedia also offers a large supply of relations, drawn from **infoboxes**, structured tables associated with certain Wikipedia articles. For example, the Wikipedia infobox for **Stanford** includes structured facts like `state = "California"` or `president = "Marc Tessier-Lavigne"`. These facts can be turned into relations like *president-of* or *located-in*. or into relations in a metalanguage called **RDF** (Resource Description Framework). An **RDF triple** is a tuple of entity-relation-entity, called a subject-predicate-object expression. Here's a sample RDF triple:

RDF
RDF triple

subject	predicate	object
Golden Gate Park	location	San Francisco

For example the crowdsourced DBpedia (Bizer et al., 2009) is an ontology derived from Wikipedia containing over 2 billion RDF triples. Another dataset from Wikipedia infoboxes, **Freebase** (Bollacker et al., 2008), now part of Wikidata (Vrandečić and Krötzsch, 2014), has relations between people and their nationality, or locations, and other locations they are contained in.

Freebase

WordNet or other ontologies offer useful ontological relations that express hierarchical relations between words or concepts. For example WordNet has the **is-a** or **hypernym** relation between classes,

is-a
hypernym

Giraffe is-a ruminant is-a ungulate is-a mammal is-a vertebrate ...

WordNet also has *Instance-of* relation between individuals and classes, so that for example *San Francisco* is in the *Instance-of* relation with *city*. Extracting these relations is an important step in extending or building ontologies.

Finally, there are large datasets that contain sentences hand-labeled with their relations, designed for training and testing relation extractors. The TACRED dataset (Zhang et al., 2017) contains 106,264 examples of relation triples about particular people or organizations, labeled in sentences from news and web text drawn from the

annual TAC Knowledge Base Population (TAC KBP) challenges. TACRED contains 41 relation types (like per:city of birth, org:subsidiaries, org:member of, per:spouse), plus a no relation tag; examples are shown in Fig. 19.3. About 80% of all examples are annotated as no relation; having sufficient negative data is important for training supervised classifiers.

Example	Entity Types & Label
Carey will succeed Cathleen P. Black, who held the position for 15 years and will take on a new role as chairwoman of Hearst Magazines, the company said.	PERSON/TITLE Relation: *per:title*
Irene Morgan Kirkaldy, who was born and reared in Baltimore, lived on Long Island and ran a child-care center in Queens with her second husband, Stanley Kirkaldy.	PERSON/CITY Relation: *per:city_of_birth*
Baldwin declined further comment, and said JetBlue chief executive Dave Barger was unavailable.	Types: PERSON/TITLE Relation: *no_relation*

Figure 19.3 Example sentences and labels from the TACRED dataset (Zhang et al., 2017).

A standard dataset was also produced for the SemEval 2010 Task 8, detecting relations between nominals (Hendrickx et al., 2009). The dataset has 10,717 examples, each with a pair of nominals (untyped) hand-labeled with one of 9 directed relations like *product-producer* (a *factory* manufactures *suits*) or *component-whole* (my *apartment* has a large *kitchen*).

19.2 Relation Extraction Algorithms

There are five main classes of algorithms for relation extraction: **handwritten patterns**, **supervised machine learning**, **semi-supervised** (via **bootstrapping** or **distant supervision**), and **unsupervised**. We'll introduce each of these in the next sections.

19.2.1 Using Patterns to Extract Relations

Hearst patterns

The earliest and still common algorithm for relation extraction is lexico-syntactic patterns, first developed by Hearst (1992a), and therefore often called **Hearst patterns**. Consider the following sentence:

> Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use.

Hearst points out that most human readers will not know what *Gelidium* is, but that they can readily infer that it is a kind of (a **hyponym** of) *red algae*, whatever that is. She suggests that the following **lexico-syntactic pattern**

$$NP_0 \text{ such as } NP_1\{,NP_2\ldots,(and|or)NP_i\}, i \geq 1 \tag{19.2}$$

implies the following semantics

$$\forall NP_i, i \geq 1, \text{hyponym}(NP_i, NP_0) \tag{19.3}$$

allowing us to infer

$$\text{hyponym}(\text{Gelidium}, \text{red algae}) \tag{19.4}$$

NP {, NP}* {,} (and|or) other NP_H	temples, treasuries, and other important civic buildings
NP_H such as {NP,}* {(or|and)} NP	red algae such as Gelidium
such NP_H as {NP,}* {(or|and)} NP	such authors as Herrick, Goldsmith, and Shakespeare
NP_H {,} including {NP,}* {(or|and)} NP	common-law countries, including Canada and England
NP_H {,} especially {NP}* {(or|and)} NP	European countries, especially France, England, and Spain

Figure 19.4 Hand-built lexico-syntactic patterns for finding hypernyms, using {} to mark optionality (Hearst 1992a, Hearst 1998).

Figure 19.4 shows five patterns Hearst (1992a, 1998) suggested for inferring the hyponym relation; we've shown NP_H as the parent/hyponym. Modern versions of the pattern-based approach extend it by adding named entity constraints. For example if our goal is to answer questions about "Who holds what office in which organization?", we can use patterns like the following:

PER, POSITION of ORG:
George Marshall, Secretary of State of the United States

PER (named|appointed|chose|etc.) PER Prep? POSITION
Truman appointed Marshall Secretary of State

PER [be]? (named|appointed|etc.) Prep? ORG POSITION
George Marshall was named US Secretary of State

Hand-built patterns have the advantage of high-precision and they can be tailored to specific domains. On the other hand, they are often low-recall, and it's a lot of work to create them for all possible patterns.

19.2.2 Relation Extraction via Supervised Learning

Supervised machine learning approaches to relation extraction follow a scheme that should be familiar by now. A fixed set of relations and entities is chosen, a training corpus is hand-annotated with the relations and entities, and the annotated texts are then used to train classifiers to annotate an unseen test set.

The most straightforward approach, illustrated in Fig. 19.5 is: (1) Find pairs of named entities (usually in the same sentence). (2): Apply a relation-classification on each pair. The classifier can use any supervised technique (logistic regression, RNN, Transformer, random forest, etc.).

An optional intermediate filtering classifier can be used to speed up the processing by making a binary decision on whether a given pair of named entities are related (by any relation). It's trained on positive examples extracted directly from all relations in the annotated corpus, and negative examples generated from within-sentence entity pairs that are not annotated with a relation.

Feature-based supervised relation classifiers. Let's consider sample features for a feature-based classifier (like logistic regression or random forests), classifying the relationship between *American Airlines* (Mention 1, or M1) and *Tim Wagner* (Mention 2, M2) from this sentence:

(19.5) **American Airlines**, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said

These include **word** features (as embeddings, or 1-hot, stemmed or not):

- The headwords of M1 and M2 and their concatenation
  Airlines Wagner Airlines-Wagner

```
function FINDRELATIONS(words) returns relations

    relations ← nil
    entities ← FINDENTITIES(words)
    forall entity pairs ⟨e1, e2⟩ in entities do
        if RELATED?(e1, e2)
            relations ← relations+CLASSIFYRELATION(e1, e2)
```

Figure 19.5 Finding and classifying the relations among entities in a text.

- Bag-of-words and bigrams in M1 and M2
  American, Airlines, Tim, Wagner, American Airlines, Tim Wagner
- Words or bigrams in particular positions
  M2: -1 spokesman
  M2: +1 said
- Bag of words or bigrams between M1 and M2:
  a, AMR, of, immediately, matched, move, spokesman, the, unit

Named entity features:

- Named-entity types and their concatenation
  (M1: ORG, M2: PER, M1M2: ORG-PER)
- Entity Level of M1 and M2 (from the set NAME, NOMINAL, PRONOUN)
  M1: NAME [it or he would be PRONOUN]
  M2: NAME [the company would be NOMINAL]
- Number of entities between the arguments (in this case 1, for AMR)

Syntactic structure is a useful signal, often represented as the dependency or constituency **syntactic path** traversed through the tree between the entities.

- Constituent paths between M1 and M2
  $NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
- Dependency-tree paths
  $Airlines \leftarrow_{subj} matched \leftarrow_{comp} said \rightarrow_{subj} Wagner$

Neural supervised relation classifiers Neural models for relation extraction similarly treat the task as supervised classification. Let's consider a typical system applied to the TACRED relation extraction dataset and task (Zhang et al., 2017). In TACRED we are given a sentence and two spans within it: a subject, which is a person or organization, and an object, which is any other entity. The task is to assign a relation from the 42 TAC relations, or no relation.

A typical Transformer-encoder algorithm, shown in Fig. 19.6, simply takes a pretrained encoder like BERT and adds a linear layer on top of the sentence representation (for example the BERT `[CLS]` token), a linear layer that is finetuned as a 1-of-N classifier to assign one of the 43 labels. The input to the BERT encoder is partially de-lexified; the subject and object entities are replaced in the input by their NER tags. This helps keep the system from overfitting to the individual lexical items (Zhang et al., 2017). When using BERT-type Transformers for relation extraction, it helps to use versions of BERT like RoBERTa (Liu et al., 2019) or spanBERT (Joshi et al., 2020) that don't have two sequences separated by a `[SEP]` token, but instead form the input from a single long sequence of sentences.

In general, if the test set is similar enough to the training set, and if there is enough hand-labeled data, supervised relation extraction systems can get high ac-

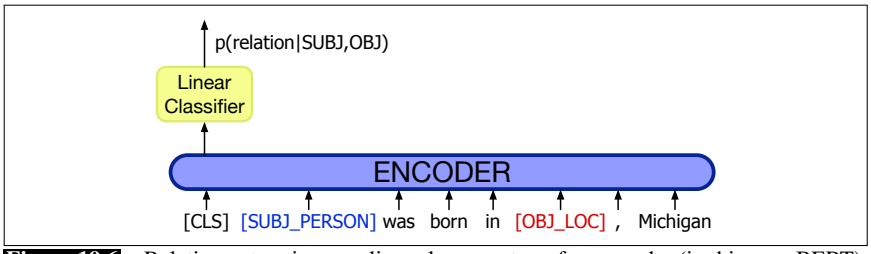

Figure 19.6 Relation extraction as a linear layer on top of an encoder (in this case BERT), with the subject and object entities replaced in the input by their NER tags (Zhang et al. 2017, Joshi et al. 2020).

curacies. But labeling a large training set is extremely expensive and supervised models are brittle: they don't generalize well to different text genres. For this reason, much research in relation extraction has focused on the semi-supervised and unsupervised approaches we turn to next.

19.2.3 Semisupervised Relation Extraction via Bootstrapping

seed patterns
seed tuples
bootstrapping

Supervised machine learning assumes that we have lots of labeled data. Unfortunately, this is expensive. But suppose we just have a few high-precision **seed patterns**, like those in Section 19.2.1, or perhaps a few **seed tuples**. That's enough to bootstrap a classifier! **Bootstrapping** proceeds by taking the entities in the seed pair, and then finding sentences (on the web, or whatever dataset we are using) that contain both entities. From all such sentences, we extract and generalize the context around the entities to learn new patterns. Fig. 19.7 sketches a basic algorithm.

function BOOTSTRAP(*Relation R*) **returns** *new relation tuples*

tuples ← Gather a set of seed tuples that have relation *R*
iterate
  sentences ← find sentences that contain entities in *tuples*
  patterns ← generalize the context between and around entities in *sentences*
  newpairs ← use *patterns* to identify more tuples
  newpairs ← *newpairs* with high confidence
  tuples ← *tuples* + *newpairs*
return tuples

Figure 19.7 Bootstrapping from seed entity pairs to learn relations.

Suppose, for example, that we need to create a list of airline/hub pairs, and we know only that Ryanair has a hub at Charleroi. We can use this seed fact to discover new patterns by finding other mentions of this relation in our corpus. We search for the terms *Ryanair*, *Charleroi* and *hub* in some proximity. Perhaps we find the following set of sentences:

(19.6) Budget airline Ryanair, which uses Charleroi as a hub, scrapped all weekend flights out of the airport.

(19.7) All flights in and out of Ryanair's hub at Charleroi airport were grounded on Friday...

(19.8) A spokesman at Charleroi, a main hub for Ryanair, estimated that 8000 passengers had already been affected.

From these results, we can use the context of words between the entity mentions, the words before mention one, the word after mention two, and the named entity types of the two mentions, and perhaps other features, to extract general patterns such as the following:

```
/ [ORG], which uses [LOC] as a hub /
/ [ORG]'s hub at [LOC] /
/ [LOC], a main hub for [ORG] /
```

These new patterns can then be used to search for additional tuples.

confidence values
semantic drift

Bootstrapping systems also assign **confidence values** to new tuples to avoid **semantic drift**. In semantic drift, an erroneous pattern leads to the introduction of erroneous tuples, which, in turn, lead to the creation of problematic patterns and the meaning of the extracted relations 'drifts'. Consider the following example:

(19.9) Sydney has a ferry hub at Circular Quay.

If accepted as a positive example, this expression could lead to the incorrect introduction of the tuple $\langle Sydney, CircularQuay \rangle$. Patterns based on this tuple could propagate further errors into the database.

Confidence values for patterns are based on balancing two factors: the pattern's performance with respect to the current set of tuples and the pattern's productivity in terms of the number of matches it produces in the document collection. More formally, given a document collection $\mathcal{D}$, a current set of tuples T, and a proposed pattern p, we need to track two factors:

- $hits(p)$: the set of tuples in T that p matches while looking in $\mathcal{D}$
- $finds(p)$: The total set of tuples that p finds in $\mathcal{D}$

The following equation balances these considerations (Riloff and Jones, 1999).

$$Conf_{RlogF}(p) = \frac{|hits(p)|}{|finds(p)|} log(|finds(p)|) \tag{19.10}$$

This metric is generally normalized to produce a probability.

noisy-or

We can assess the confidence in a proposed new tuple by combining the evidence supporting it from all the patterns P' that match that tuple in $\mathcal{D}$ (Agichtein and Gravano, 2000). One way to combine such evidence is the **noisy-or** technique. Assume that a given tuple is supported by a subset of the patterns in P, each with its own confidence assessed as above. In the noisy-or model, we make two basic assumptions. First, that for a proposed tuple to be false, *all* of its supporting patterns must have been in error, and second, that the sources of their individual failures are all independent. If we loosely treat our confidence measures as probabilities, then the probability of any individual pattern p failing is $1 - Conf(p)$; the probability of all of the supporting patterns for a tuple being wrong is the product of their individual failure probabilities, leaving us with the following equation for our confidence in a new tuple.

$$Conf(t) = 1 - \prod_{p \in P'} (1 - Conf(p)) \tag{19.11}$$

Setting conservative confidence thresholds for the acceptance of new patterns and tuples during the bootstrapping process helps prevent the system from drifting away from the targeted relation.

19.2.4 Distant Supervision for Relation Extraction

distant supervision

Although hand-labeling text with relation labels is expensive to produce, there are ways to find indirect sources of training data. The **distant supervision** method (Mintz et al., 2009) combines the advantages of bootstrapping with supervised learning. Instead of just a handful of seeds, distant supervision uses a large database to acquire a huge number of seed examples, creates lots of noisy pattern features from all these examples and then combines them in a supervised classifier.

For example suppose we are trying to learn the *place-of-birth* relationship between people and their birth cities. In the seed-based approach, we might have only 5 examples to start with. But Wikipedia-based databases like DBPedia or Freebase have tens of thousands of examples of many relations; including over 100,000 examples of *place-of-birth*, (<Edwin Hubble, Marshfield>, <Albert Einstein, Ulm>, etc.,). The next step is to run named entity taggers on large amounts of text—Mintz et al. (2009) used 800,000 articles from Wikipedia—and extract all sentences that have two named entities that match the tuple, like the following:

...Hubble was born in Marshfield...
...Einstein, born (1879), Ulm...
...Hubble's birthplace in Marshfield...

Training instances can now be extracted from this data, one training instance for each identical tuple <relation, entity1, entity2>. Thus there will be one training instance for each of:

```
<born-in, Edwin Hubble, Marshfield>
<born-in, Albert Einstein, Ulm>
<born-year, Albert Einstein, 1879>
```

and so on.

We can then apply feature-based or neural classification. For feature-based classification, we can use standard supervised relation extraction features like the named entity labels of the two mentions, the words and dependency paths in between the mentions, and neighboring words. Each tuple will have features collected from many training instances; the feature vector for a single training instance like (<born-in,Albert Einstein, Ulm> will have lexical and syntactic features from many different sentences that mention Einstein and Ulm.

Because distant supervision has very large training sets, it is also able to use very rich features that are conjunctions of these individual features. So we will extract thousands of patterns that conjoin the entity types with the intervening words or dependency paths like these:

PER was born in LOC
PER, born (XXXX), LOC
PER's birthplace in LOC

To return to our running example, for this sentence:

(19.12) **American Airlines**, a unit of AMR, immediately matched the move, spokesman **Tim Wagner** said

we would learn rich conjunction features like this one:

M1 = ORG & M2 = PER & nextword="said"& path= $NP \uparrow NP \uparrow S \uparrow S \downarrow NP$

The result is a supervised classifier that has a huge rich set of features to use in detecting relations. Since not every test sentence will have one of the training

relations, the classifier will also need to be able to label an example as *no-relation*. This label is trained by randomly selecting entity pairs that do not appear in any Freebase relation, extracting features for them, and building a feature vector for each such tuple. The final algorithm is sketched in Fig. 19.8.

```
function DISTANT SUPERVISION(Database D, Text T) returns relation classifier C

  foreach relation R
    foreach tuple (e1,e2) of entities with relation R in D
      sentences ← Sentences in T that contain e1 and e2
      f ← Frequent features in sentences
      observations ← observations + new training tuple (e1, e2, f, R)
  C ← Train supervised classifier on observations
  return C
```

Figure 19.8 The distant supervision algorithm for relation extraction. A neural classifier would skip the feature set f.

Distant supervision shares advantages with each of the methods we've examined. Like supervised classification, distant supervision uses a classifier with lots of features, and supervised by detailed hand-created knowledge. Like pattern-based classifiers, it can make use of high-precision evidence for the relation between entities. Indeed, distance supervision systems learn patterns just like the hand-built patterns of early relation extractors. For example the *is-a* or *hypernym* extraction system of Snow et al. (2005) used hypernym/hyponym NP pairs from WordNet as distant supervision, and then learned new patterns from large amounts of text. Their system induced exactly the original 5 template patterns of Hearst (1992a), but also 70,000 additional patterns including these four:

NP_H like NP	*Many hormones like leptin...*
NP_H called NP	*...using a markup language called XHTML*
NP is a NP_H	Ruby is a programming language...
NP, a NP_H	*IBM, a company with a long...*

This ability to use a large number of features simultaneously means that, unlike the iterative expansion of patterns in seed-based systems, there's no semantic drift. Like unsupervised classification, it doesn't use a labeled training corpus of texts, so it isn't sensitive to genre issues in the training corpus, and relies on very large amounts of unlabeled data. Distant supervision also has the advantage that it can create training tuples to be used with neural classifiers, where features are not required.

The main problem with distant supervision is that it tends to produce low-precision results, and so current research focuses on ways to improve precision. Furthermore, distant supervision can only help in extracting relations for which a large enough database already exists. To extract new relations without datasets, or relations for new domains, purely unsupervised methods must be used.

19.2.5 Unsupervised Relation Extraction

open information extraction

The goal of unsupervised relation extraction is to extract relations from the web when we have no labeled training data, and not even any list of relations. This task is often called **open information extraction** or **Open IE**. In Open IE, the relations

are simply strings of words (usually beginning with a verb).

For example, the **ReVerb** system (Fader et al., 2011) extracts a relation from a sentence s in 4 steps:

1. Run a part-of-speech tagger and entity chunker over s
2. For each verb in s, find the longest sequence of words w that start with a verb and satisfy syntactic and lexical constraints, merging adjacent matches.
3. For each phrase w, find the nearest noun phrase x to the left which is not a relative pronoun, wh-word or existential "there". Find the nearest noun phrase y to the right.
4. Assign confidence c to the relation $r = (x, w, y)$ using a confidence classifier and return it.

A relation is only accepted if it meets syntactic and lexical constraints. The syntactic constraints ensure that it is a verb-initial sequence that might also include nouns (relations that begin with light verbs like *make*, *have*, or *do* often express the core of the relation with a noun, like *have a hub in*):

V | VP | VW*P
V = verb particle? adv?
W = (noun | adj | adv | pron | det)
P = (prep | particle | infinitive "to")

The lexical constraints are based on a dictionary D that is used to prune very rare, long relation strings. The intuition is to eliminate candidate relations that don't occur with sufficient number of distinct argument types and so are likely to be bad examples. The system first runs the above relation extraction algorithm offline on 500 million web sentences and extracts a list of all the relations that occur after normalizing them (removing inflection, auxiliary verbs, adjectives, and adverbs). Each relation r is added to the dictionary if it occurs with at least 20 different arguments. Fader et al. (2011) used a dictionary of 1.7 million normalized relations.

Finally, a confidence value is computed for each relation using a logistic regression classifier. The classifier is trained by taking 1000 random web sentences, running the extractor, and hand labeling each extracted relation as correct or incorrect. A confidence classifier is then trained on this hand-labeled data, using features of the relation and the surrounding words. Fig. 19.9 shows some sample features used in the classification.

(x,r,y) covers all words in s
the last preposition in r is *for*
the last preposition in r is *on*
len(s) $\leq$ 10
there is a coordinating conjunction to the left of r in s
r matches a lone V in the syntactic constraints
there is preposition to the left of x in s
there is an NP to the right of y in s

Figure 19.9 Features for the classifier that assigns confidence to relations extracted by the Open Information Extraction system REVERB (Fader et al., 2011).

For example the following sentence:

(19.13) United has a hub in Chicago, which is the headquarters of United Continental Holdings.

has the relation phrases *has a hub in* and *is the headquarters of* (it also has *has* and *is*, but longer phrases are preferred). Step 3 finds *United* to the left and *Chicago* to the right of *has a hub in*, and skips over *which* to find Chicago to the left of *is the headquarters of*. The final output is:

```
r1:  <United, has a hub in, Chicago>
r2:  <Chicago, is the headquarters of, United Continental Holdings>
```

The great advantage of unsupervised relation extraction is its ability to handle a huge number of relations without having to specify them in advance. The disadvantage is the need to map all the strings into some canonical form for adding to databases or knowledge graphs. Current methods focus heavily on relations expressed with verbs, and so will miss many relations that are expressed nominally.

19.2.6 Evaluation of Relation Extraction

Supervised relation extraction systems are evaluated by using test sets with human-annotated, gold-standard relations and computing precision, recall, and F-measure. Labeled precision and recall require the system to classify the relation correctly, whereas unlabeled methods simply measure a system's ability to detect entities that are related.

Semi-supervised and **unsupervised** methods are much more difficult to evaluate, since they extract totally new relations from the web or a large text. Because these methods use very large amounts of text, it is generally not possible to run them solely on a small labeled test set, and as a result it's not possible to pre-annotate a gold set of correct instances of relations.

For these methods it's possible to approximate (only) precision by drawing a random sample of relations from the output, and having a human check the accuracy of each of these relations. Usually this approach focuses on the **tuples** to be extracted from a body of text rather than on the relation **mentions**; systems need not detect every mention of a relation to be scored correctly. Instead, the evaluation is based on the set of tuples occupying the database when the system is finished. That is, we want to know if the system can discover that Ryanair has a hub at Charleroi; we don't really care how many times it discovers it. The estimated precision $\hat{P}$ is then

$$\hat{P} = \frac{\text{\# of correctly extracted relation tuples in the sample}}{\text{total \# of extracted relation tuples in the sample.}} \tag{19.14}$$

Another approach that gives us a little bit of information about recall is to compute precision at different levels of recall. Assuming that our system is able to rank the relations it produces (by probability, or confidence) we can separately compute precision for the top 1000 new relations, the top 10,000 new relations, the top 100,000, and so on. In each case we take a random sample of that set. This will show us how the precision curve behaves as we extract more and more tuples. But there is no way to directly evaluate recall.

19.3 Extracting Events

event extraction

The task of **event extraction** is to identify mentions of events in texts. For the purposes of this task, an event mention is any expression denoting an event or state that can be assigned to a particular point, or interval, in time. The following markup of the sample text on page 415 shows all the events in this text.

[EVENT Citing] high fuel prices, United Airlines [EVENT said] Friday it has [EVENT increased] fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately [EVENT matched] [EVENT the move], spokesman Tim Wagner [EVENT said]. United, a unit of UAL Corp., [EVENT said] [EVENT the increase] took effect Thursday and [EVENT applies] to most routes where it [EVENT competes] against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

In English, most event mentions correspond to verbs, and most verbs introduce events. However, as we can see from our example, this is not always the case. Events can be introduced by noun phrases, as in *the move* and *the increase*, and some verbs fail to introduce events, as in the phrasal verb *took effect*, which refers to when the event began rather than to the event itself. Similarly, **light verbs** such as *make*, *take*, and *have* often fail to denote events. A light verb is a verb that has very little meaning itself, and the associated event is instead expressed by its direct object noun. In light verb examples like *took a flight*, it's the word *flight* that defines the event; these light verbs just provide a syntactic structure for the noun's arguments.

light verbs

Various versions of the event extraction task exist, depending on the goal. For example in the TempEval shared tasks (Verhagen et al. 2009) the goal is to extract events and aspects like their aspectual and temporal properties. Events are to be classified as actions, states, **reporting events** (*say, report, tell, explain*), perception events, and so on. The aspect, tense, and modality of each event also needs to be extracted. Thus for example the various *said* events in the sample text would be annotated as (class=REPORTING, tense=PAST, aspect=PERFECTIVE).

reporting events

Event extraction is generally modeled via supervised learning, detecting events via IOB sequence models and assigning event classes and attributes with multi-class classifiers. The input can be neural models starting from encoders; or classic feature-based models using features like those in Fig. 19.10.

Feature	Explanation
Character affixes	Character-level prefixes and suffixes of target word
Nominalization suffix	Character-level suffixes for nominalizations (e.g., *-tion*)
Part of speech	Part of speech of the target word
Light verb	Binary feature indicating that the target is governed by a light verb
Subject syntactic category	Syntactic category of the subject of the sentence
Morphological stem	Stemmed version of the target word
Verb root	Root form of the verb basis for a nominalization
WordNet hypernyms	Hypernym set for the target

Figure 19.10 Features commonly used in classic feature-based approaches to event detection.

19.4 Representing Time

temporal logic

Let's begin by introducing the basics of **temporal logic** and how human languages convey temporal information. The most straightforward theory of time holds that it flows inexorably forward and that events are associated with either points or intervals in time, as on a timeline. We can order distinct events by situating them on the timeline; one event *precedes* another if the flow of time leads from the first event

to the second. Accompanying these notions in most theories is the idea of the current moment in time. Combining this notion with the idea of a temporal ordering relationship yields the familiar notions of past, present, and future.

Various kinds of temporal representation systems can be used to talk about temporal ordering relationship. One of the most commonly used in computational modeling is the **interval algebra** of Allen (1984). Allen models all events and time expressions as intervals there is no representation for points (although intervals can be very short). In order to deal with intervals without points, he identifies 13 primitive relations that can hold between these temporal intervals. Fig. 19.11 shows these 13 **Allen relations**.

interval algebra

Allen relations

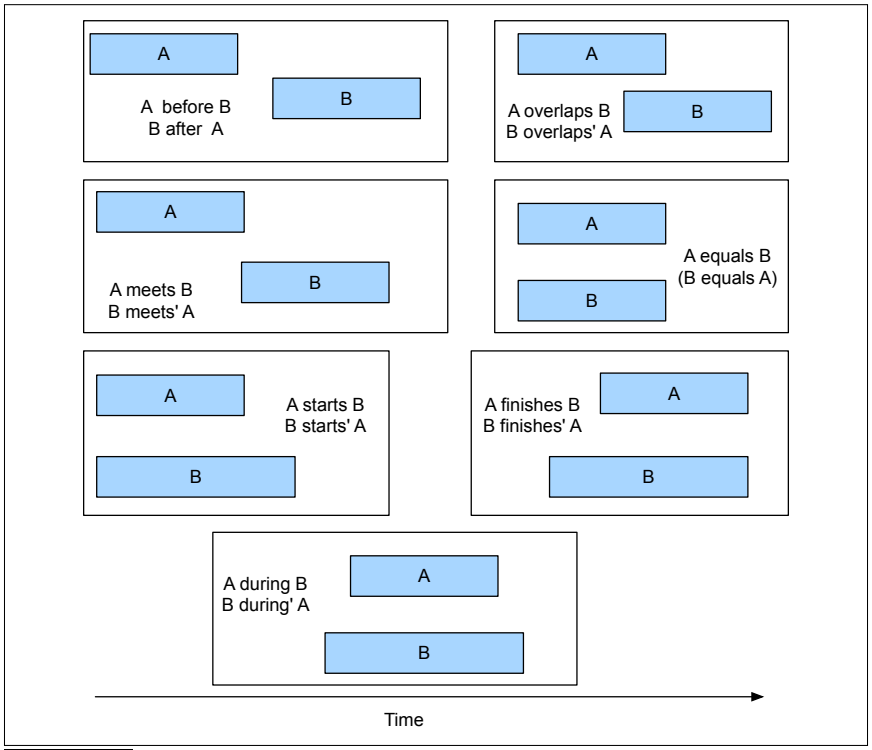

Figure 19.11 The 13 temporal relations from Allen (1984).

19.4.1 Reichenbach's reference point

The relation between simple verb tenses and points in time is by no means straightforward. The present tense can be used to refer to a future event, as in this example:

(19.15) Ok, we fly from San Francisco to Boston at 10.

Or consider the following examples:

(19.16) Flight 1902 arrived late.

(19.17) Flight 1902 had arrived late.

Although both refer to events in the past, representing them in the same way seems wrong. The second example seems to have another unnamed event lurking in the background (e.g., Flight 1902 had already arrived late *when* something else happened).

reference point

To account for this phenomena, Reichenbach (1947) introduced the notion of a **reference point**. In our simple temporal scheme, the current moment in time is equated with the time of the utterance and is used as a reference point for when the event occurred (before, at, or after). In Reichenbach's approach, the notion of the reference point is separated from the utterance time and the event time. The following examples illustrate the basics of this approach:

(19.18) When Mary's flight departed, I ate lunch.

(19.19) When Mary's flight departed, I had eaten lunch.

In both of these examples, the eating event has happened in the past, that is, prior to the utterance. However, the verb tense in the first example indicates that the eating event began when the flight departed, while the second example indicates that the eating was accomplished prior to the flight's departure. Therefore, in Reichenbach's terms the *departure* event specifies the reference point. These facts can be accommodated by additional constraints relating the *eating* and *departure* events. In the first example, the reference point precedes the *eating* event, and in the second example, the eating precedes the reference point. Figure 19.12 illustrates Reichenbach's approach with the primary English tenses. Exercise 19.4 asks you to represent these examples in FOL.

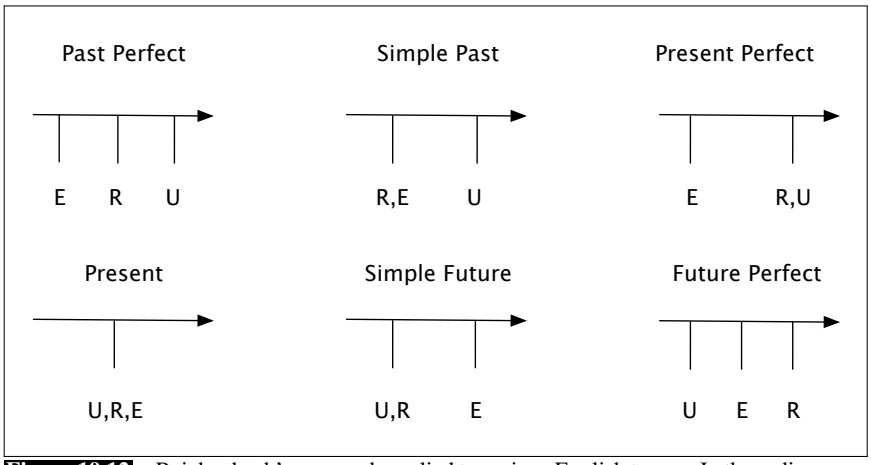

Figure 19.12 Reichenbach's approach applied to various English tenses. In these diagrams, time flows from left to right, **E** denotes the time of the event, **R** denotes the reference time, and **U** denotes the time of the utterance.

Languages have many other ways to convey temporal information besides tense. Most useful for our purposes will be temporal expressions like *in the morning* or *6:45* or *afterwards*.

(19.20) I'd like to go at 6:45 in the morning.

(19.21) Somewhere around noon, please.

(19.22) I want to take the train back afterwards.

Incidentally, temporal expressions display a fascinating metaphorical conceptual organization. Temporal expressions in English are frequently expressed in spatial terms, as is illustrated by the various uses of *at*, *in*, *somewhere*, and *near* in these examples (Lakoff and Johnson 1980, Jackendoff 1983). Metaphorical organizations such as these, in which one domain is systematically expressed in terms of another, are very common in languages of the world.

19.5 Representing Aspect

aspect

A related notion to time is **aspect**, which is what we call the way events can be categorized by their internal temporal structure or temporal contour. By this we mean questions like whether events are ongoing or have ended, or whether they are conceptualized as happening at a point in time or over some interval. Such notions of temporal contour have been used to divide event expressions into classes since Aristotle, although the set of four classes we'll introduce here is due to Vendler (1967) (you may also see the German term **aktionsart** used to refer to these classes).

aktionsart

events

states

stative

The most basic aspectual distinction is between **events** (which involve change) and **states** (which do not involve change). **Stative expressions** represent the notion of an event participant being in a **state**, or having a particular property, at a given point in time. Stative expressions capture aspects of the world at a single point in time, and conceptualize the participant as unchanging and continuous. Consider the following ATIS examples.

(19.23) I like express trains.

(19.24) I need the cheapest fare.

(19.25) I want to go first class.

In examples like these, the event participant denoted by the subject can be seen as experiencing something at a specific point in time, and don't involve any kind of internal change over time (the liking or needing is conceptualized as continuous and unchanging).

activity

Non-states (which we'll refer to as **events**) are divided into subclasses; we'll introduce three here. **Activity expressions** describe events undertaken by a participant that occur over a span of time (rather than being conceptualized as a single point in time like stative expressions), and have no particular end point. Of course in practice all things end, but the meaning of the expression doesn't represent this fact. Consider the following examples:

(19.26) She drove a Mazda.

(19.27) I live in Brooklyn.

These examples both specify that the subject is engaged in, or has engaged in, the activity specified by the verb for some period of time, but doesn't specify when the driving or living might have stopped.

Two more classes of expressions, **achievement** expressions and **accomplishment** expressions, describe events that take place over time, but also conceptualize the event as having a particular kind of endpoint or goal. The Greek word *telos* means 'end' or 'goal' and so the events described by these kinds of expressions are often called **telic** events.

telic

accomplishment expressions

Accomplishment expressions describe events that have a natural end point and result in a particular state. Consider the following examples:

(19.28) He booked me a reservation.

(19.29) The 7:00 train got me to New York City.

In these examples, an event is seen as occurring over some period of time that ends when the intended state is accomplished (i.e., the state of me having a reservation, or me being in New York City).

achievement expressions

The final aspectual class, **achievement expressions**, is only subtly different than accomplishments. Consider the following:

(19.30) She found her gate.

(19.31) I reached New York.

Like accomplishment expressions, achievement expressions result in a state. But unlike accomplishments, achievement events are 'punctual': they are thought of as happening in an instant and the verb doesn't conceptualize the process or activity leading up the state. Thus the events in these examples may in fact have been preceded by extended *searching* or *traveling* events, but the verb doesn't conceptualize these preceding processes, but rather conceptualizes the events corresponding to *finding* and *reaching* as points, not intervals.

In summary, a standard way of categorizing event expressions by their temporal contours is via these four general classes:

Stative: I know my departure gate.

Activity: John is flying.

Accomplishment: Sally booked her flight.

Achievement: She found her gate.

Before moving on, note that event expressions can easily be shifted from one class to another. Consider the following examples:

(19.32) I flew.

(19.33) I flew to New York.

The first example is a simple activity; it has no natural end point. The second example is clearly an accomplishment event since it has an end point, and results in a particular state. Clearly, the classification of an event is not solely governed by the verb, but by the semantics of the entire expression in context.

19.6 Temporally Annotated Datasets: TimeBank

TimeBank

The **TimeBank** corpus consists of American English text annotated with temporal information (Pustejovsky et al., 2003). The annotations use TimeML (Saurí et al., 2006), a markup language for time based on Allen's interval algebra discussed above (Allen, 1984). There are three types of TimeML objects: an EVENT represent events and states, a TIME represents time expressions like dates, and a LINK represents various relationships between events and times (event-event, event-time, and time-time). The links include temporal links (TLINK) for the 13 Allen relations, aspectual links (ALINK) for aspectual relationships between events and subevents, and SLINKS which mark factuality.

Consider the following sample sentence and its corresponding markup shown in Fig. 19.13, selected from one of the TimeBank documents.

(19.34) Delta Air Lines earnings soared 33% to a record in the fiscal first quarter, bucking the industry trend toward declining profits.

This text has three events and two temporal expressions (including the creation time of the article, which serves as the document time), and four temporal links that capture the using the Allen relations:

- Soaring$_{e1}$ is **included** in the fiscal first quarter$_{t58}$
- Soaring$_{e1}$ is **before** 1989-10-26$_{t57}$
- Soaring$_{e1}$ is **simultaneous** with the bucking$_{e3}$

```
<TIMEX3 tid="t57" type="DATE" value="1989-10-26"  functionInDocument="CREATION_TIME">
10/26/89  </TIMEX3>

Delta Air Lines earnings <EVENT eid="e1" class="OCCURRENCE"> soared </EVENT> 33% to a
record in  <TIMEX3 tid="t58" type="DATE" value="1989-Q1" anchorTimeID="t57"> the
fiscal first quarter </TIMEX3>, <EVENT eid="e3"  class="OCCURRENCE">bucking</EVENT>
the industry trend toward <EVENT eid="e4" class="OCCURRENCE">declining</EVENT>
profits.
```

Figure 19.13 Example from the TimeBank corpus.

- Declining$_{e4}$ **includes** soaring$_{e1}$

We can also visualize the links as a graph. The TimeBank snippet in Eq. 19.35 would be represented with a graph like Fig. 19.14.

(19.35) **[DCT:11/02/891]**$_1$: Pacific First Financial Corp. **said**$_2$ shareholders **approved**$_3$ its **acquisition**$_4$ by Royal Trustco Ltd. of Toronto for $27 a share, or $212 million. The thrift holding company **said**$_5$ it **expects**$_6$ to **obtain**$_7$ regulatory **approval**$_8$ and **complete**$_9$ the **transaction**$_{10}$ by **year-end**$_{11}$.

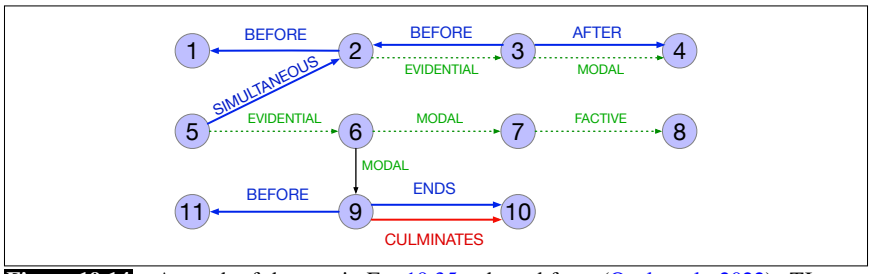

Figure 19.14 A graph of the text in Eq. 19.35, adapted from (Ocal et al., 2022). TLINKS are shown in blue, ALINKS in red, and SLINKS in green.

19.7 Automatic Temporal Analysis

Here we introduce the three common steps used in analyzing time in text:

1. Extracting **temporal expressions**
2. **Normalizing** these expressions, by converting them to a standard format.
3. **Linking** events to times and extracting time graphs and timelines

19.7.1 Extracting Temporal Expressions

absolute **relative** **duration**

Temporal expressions are phrases that refer to absolute points in time, relative times, durations, and sets of these. **Absolute** temporal expressions are those that can be mapped directly to calendar dates, times of day, or both. **Relative** temporal expressions map to particular times through some other reference point (as in *a week from last Tuesday*). Finally, **durations** denote spans of time at varying levels of granularity (seconds, minutes, days, weeks, centuries, etc.). Figure 19.15 lists some sample temporal expressions in each of these categories.

lexical triggers

Temporal expressions are grammatical constructions that often have temporal **lexical triggers** as their heads, making them easy to find. Lexical triggers might be nouns, proper nouns, adjectives, and adverbs; full temporal expressions consist

Absolute	Relative	Durations
April 24, 1916	yesterday	four hours
The summer of '77	next semester	three weeks
10:15 AM	two weeks from yesterday	six days
The 3rd quarter of 2006	last quarter	the last three quarters

Figure 19.15 Examples of absolute, relational and durational temporal expressions.

of their phrasal projections: noun phrases, adjective phrases, and adverbial phrases (Figure 19.16).

Category	Examples
Noun	*morning*, *noon*, *night*, *winter*, *dusk*, *dawn*
Proper Noun	*January, Monday, Ides, Easter, Rosh Hashana, Ramadan, Tet*
Adjective	*recent, past, annual, former*
Adverb	*hourly, daily, monthly, yearly*

Figure 19.16 Examples of temporal lexical triggers.

The task is to detect temporal expressions in running text, like this examples, shown with TIMEX3 tags (Pustejovsky et al. 2005, Ferro et al. 2005).

> A fare increase initiated <TIMEX3>last week</TIMEX3> by UAL Corp's United Airlines was matched by competitors over <TIMEX3>the weekend</TIMEX3>, marking the second successful fare increase in <TIMEX3>two weeks</TIMEX3>.

Rule-based approaches use cascades of regular expressions to recognize larger and larger chunks from previous stages, based on patterns containing parts of speech, trigger words (e.g., *February*) or classes (e.g., *MONTH*) (Chang and Manning, 2012; Strötgen and Gertz, 2013; Chambers, 2013). Here's a rule from SUTime (Chang and Manning, 2012) for detecting expressions like *3 years old*:

```
/(\d+)[-\s]($TEUnits)(s)?([-\s]old)?/
```

Sequence-labeling approaches use the standard IOB scheme, marking words that are either (I)nside, (O)utside or at the (B)eginning of a temporal expression:

A fare increase initiated last week by UAL Corp's...
O O O O B I O O O

A statistical sequence labeler is trained, using either embeddings or a fine-tuned encoder, or classic features extracted from the token and context including words, lexical triggers, and POS.

Temporal expression recognizers are evaluated with the usual recall, precision, and *F*-measures. A major difficulty for all of these very lexicalized approaches is avoiding expressions that trigger false positives:

(19.36) *1984* tells the story of Winston Smith...

(19.37) ...U2's classic *Sunday* Bloody *Sunday*

19.7.2 Temporal Normalization

temporal normalization

Temporal normalization is the task of mapping a temporal expression to a point in time or to a duration. Points in time correspond to calendar dates, to times of day, or both. Durations primarily consist of lengths of time. Normalized times are represented via the ISO 8601 standard for encoding temporal values (ISO8601, 2004). Fig. 19.17 reproduces our earlier example with these value attributes.

```
<TIMEX3 id="t1'' type="DATE" value="2007-07-02" functionInDocument="CREATION_TIME">
    July 2, 2007 </TIMEX3> A fare increase initiated <TIMEX3 id="t2" type="DATE"
    value="2007-W26" anchorTimeID="t1">last week</TIMEX3> by United Airlines was
    matched by competitors over <TIMEX3 id="t3" type="DURATION" value="P1WE"
    anchorTimeID="t1"> the weekend </TIMEX3>, marking the second successful fare
    increase in <TIMEX3 id="t4" type="DURATION" value="P2W" anchorTimeID="t1"> two
    weeks </TIMEX3>.
```

Figure 19.17 TimeML markup including normalized values for temporal expressions.

The dateline, or document date, for this text was *July 2, 2007*. The ISO representation for this kind of expression is YYYY-MM-DD, or in this case, 2007-07-02. The encodings for the temporal expressions in our sample text all follow from this date, and are shown here as values for the VALUE attribute.

The first temporal expression in the text proper refers to a particular week of the year. In the ISO standard, weeks are numbered from 01 to 53, with the first week of the year being the one that has the first Thursday of the year. These weeks are represented with the template YYYY-Wnn. The ISO week for our document date is week 27; thus the value for *last week* is represented as "2007-W26".

The next temporal expression is *the weekend*. ISO weeks begin on Monday; thus, weekends occur at the end of a week and are fully contained within a single week. Weekends are treated as durations, so the value of the VALUE attribute has to be a length. Durations are represented according to the pattern P*nx*, where *n* is an integer denoting the length and *x* represents the unit, as in P3Y for *three years* or P2D for *two days*. In this example, one weekend is captured as P1WE. In this case, there is also sufficient information to anchor this particular weekend as part of a particular week. Such information is encoded in the ANCHORTIMEID attribute. Finally, the phrase *two weeks* also denotes a duration captured as P2W. Figure 19.18 give some more examples, but there is a lot more to the various temporal annotation standards; consult ISO8601 (2004), Ferro et al. (2005), and Pustejovsky et al. (2005) for more details.

Unit	Pattern	Sample Value
Fully specified dates	YYYY-MM-DD	1991-09-28
Weeks	YYYY-Wnn	2007-W27
Weekends	PnWE	P1WE
24-hour clock times	HH:MM:SS	11:13:45
Dates and times	YYYY-MM-DDTHH:MM:SS	1991-09-28T11:00:00
Financial quarters	Qn	1999-Q3

Figure 19.18 Sample ISO patterns for representing various times and durations.

Most current approaches to temporal normalization are rule-based (Chang and Manning 2012, Strötgen and Gertz 2013). Patterns that match temporal expressions are associated with semantic analysis procedures. For example, the pattern above for recognizing phrases like *3 years old* can be associated with the predicate *Duration* that takes two arguments, the length and the unit of time:

```
pattern: /(\d+)[-\s]($TEUnits)(s)?([-\s]old)?/
result: Duration($1, $2)
```

The task is difficult because fully qualified temporal expressions are fairly rare in real texts. Most temporal expressions in news articles are incomplete and are only implicitly anchored, often with respect to the dateline of the article, which we refer to as the document's **temporal anchor**. The values of temporal expressions such as *today*, *yesterday*, or *tomorrow* can all be computed with respect to this temporal

temporal anchor

anchor. The semantic procedure for *today* simply assigns the anchor, and the attachments for *tomorrow* and *yesterday* add a day and subtract a day from the anchor, respectively. Of course, given the cyclic nature of our representations for months, weeks, days, and times of day, our temporal arithmetic procedures must use modulo arithmetic appropriate to the time unit being used.

Unfortunately, even simple expressions such as *the weekend* or *Wednesday* introduce a fair amount of complexity. In our current example, *the weekend* clearly refers to the weekend of the week that immediately precedes the document date. But this won't always be the case, as is illustrated in the following example.

(19.38) Random security checks that began yesterday at Sky Harbor will continue at least through the weekend.

In this case, the expression *the weekend* refers to the weekend of the week that the anchoring date is part of (i.e., the coming weekend). The information that signals this meaning comes from the tense of *continue*, the verb governing *the weekend*.

Relative temporal expressions are handled with temporal arithmetic similar to that used for *today* and *yesterday*. The document date indicates that our example article is ISO week 27, so the expression *last week* normalizes to the current week minus 1. To resolve ambiguous *next* and *last* expressions we consider the distance from the anchoring date to the nearest unit. *Next Friday* can refer either to the immediately next Friday or to the Friday following that, but the closer the document date is to a Friday, the more likely it is that the phrase will skip the nearest one. Such ambiguities are handled by encoding language and domain-specific heuristics into the temporal attachments.

19.7.3 Temporal Ordering of Events

The goal of temporal analysis, is to link times to events and then fit all these events into a complete timeline. This ambitious task is the subject of considerable current research but solving it with a high level of accuracy is beyond the capabilities of current systems. A somewhat simpler, but still useful, task is to impose a partial ordering on the events and temporal expressions mentioned in a text. Such an ordering can provide many of the same benefits as a true timeline. An example of such a partial ordering is the determination that the fare increase by *American Airlines* came *after* the fare increase by *United* in our sample text. Determining such an ordering can be viewed as a binary relation detection and classification task.

Even this partial ordering task assumes that in addition to the detecting and normalizing time expressions steps described above, we have already detected all the events in the text. Indeed, many temporal expressions are anchored to events mentioned in a text and not directly to other temporal expressions. Consider the following example:

(19.39) One week after the storm, JetBlue issued its customer bill of rights.

To determine when JetBlue issued its customer bill of rights we need to determine the time of *the storm* event, and then we need to modify that time by the temporal expression *one week after*.

Thus once the events and times have been detected, our goal next is to assert links between all the times and events: i.e. creating event-event, event-time, time-time, DCT-event, and DCT-time TimeML TLINKS. This can be done by training time relation classifiers to predict the correct T:INK between each pair of times/events, supervised by the gold labels in the TimeBank corpus with features like words/embeddings, parse paths, tense and aspect The sieve-based architecture using precision-

ranked sets of classifiers, which we'll introduce in Chapter 26, is also commonly used.

Systems that perform all 4 tasks (time extraction creation and normalization, event extraction, and time/event linking) include TARSQI (Verhagen et al., 2005) CLEARTK (Bethard, 2013), CAEVO (Chambers et al., 2014), and CATENA (Mirza and Tonelli, 2016).

19.8 Template Filling

Many texts contain reports of events, and possibly sequences of events, that often correspond to fairly common, stereotypical situations in the world. These abstract situations or stories, related to what have been called **scripts** (Schank and Abelson, 1977), consist of prototypical sequences of sub-events, participants, and their roles. The strong expectations provided by these scripts can facilitate the proper classification of entities, the assignment of entities into roles and relations, and most critically, the drawing of inferences that fill in things that have been left unsaid. In their simplest form, such scripts can be represented as **templates** consisting of fixed sets of **slots** that take as values **slot-fillers** belonging to particular classes. The task of **template filling** is to find documents that invoke particular scripts and then fill the slots in the associated templates with fillers extracted from the text. These slot-fillers may consist of text segments extracted directly from the text, or they may consist of concepts that have been inferred from text elements through some additional processing.

scripts

templates

template filling

A filled template from our original airline story might look like the following.

FARE-RAISE ATTEMPT:	LEAD AIRLINE:	UNITED AIRLINES
	AMOUNT:	$6
	EFFECTIVE DATE:	2006-10-26
	FOLLOWER:	AMERICAN AIRLINES

This template has four slots (LEAD AIRLINE, AMOUNT, EFFECTIVE DATE, FOLLOWER). The next section describes a standard sequence-labeling approach to filling slots. Section 19.8.2 then describes an older system based on the use of cascades of finite-state transducers and designed to address a more complex template-filling task that current learning-based systems don't yet address.

19.8.1 Machine Learning Approaches to Template Filling

In the standard paradigm for template filling, we are given training documents with text spans annotated with predefined templates and their slot fillers. Our goal is to create one template for each event in the input, filling in the slots with text spans.

The task is generally modeled by training two separate supervised systems. The first system decides whether the template is present in a particular sentence. This task is called **template recognition** or sometimes, in a perhaps confusing bit of terminology, *event recognition*. Template recognition can be treated as a text classification task, with features extracted from every sequence of words that was labeled in training documents as filling any slot from the template being detected. The usual set of features can be used: tokens, embeddings, word shapes, part-of-speech tags, syntactic chunk tags, and named entity tags.

template recognition

role-filler extraction

The second system has the job of **role-filler extraction**. A separate classifier is trained to detect each role (LEAD-AIRLINE, AMOUNT, and so on). This can be a binary classifier that is run on every noun-phrase in the parsed input sentence, or a sequence model run over sequences of words. Each role classifier is trained on the labeled data in the training set. Again, the usual set of features can be used, but now trained only on an individual noun phrase or the fillers of a single slot.

Multiple non-identical text segments might be labeled with the same slot label. For example in our sample text, the strings *United* or *United Airlines* might be labeled as the LEAD AIRLINE. These are not incompatible choices and the coreference resolution techniques introduced in Chapter 26 can provide a path to a solution.

A variety of annotated collections have been used to evaluate this style of approach to template filling, including sets of job announcements, conference calls for papers, restaurant guides, and biological texts. A key open question is extracting templates in cases where there is no training data or even predefined templates, by inducing templates as sets of linked events (Chambers and Jurafsky, 2011).

19.8.2 Earlier Finite-State Template-Filling Systems

The templates above are relatively simple. But consider the task of producing a template that contained all the information in a text like this one (Grishman and Sundheim, 1995):

> Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan. The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and "metal wood" clubs a month.

The MUC-5 'joint venture' task (the *Message Understanding Conferences* were a series of U.S. government-organized information-extraction evaluations) was to produce hierarchically linked templates describing joint ventures. Figure 19.19 shows a structure produced by the FASTUS system (Hobbs et al., 1997). Note how the filler of the ACTIVITY slot of the TIE-UP template is itself a template with slots.

Tie-up-1		**Activity-1**:	
RELATIONSHIP	tie-up	COMPANY	Bridgestone Sports Taiwan Co.
ENTITIES	Bridgestone Sports Co.	PRODUCT	iron and "metal wood" clubs
	a local concern	START DATE	DURING: January 1990
	a Japanese trading house		
JOINT VENTURE	Bridgestone Sports Taiwan Co.		
ACTIVITY	Activity-1		
AMOUNT	NT$20000000		

Figure 19.19 The templates produced by FASTUS given the input text on page 437.

Early systems for dealing with these complex templates were based on cascades of transducers based on handwritten rules, as sketched in Fig. 19.20.

The first four stages use handwritten regular expression and grammar rules to do basic tokenization, chunking, and parsing. Stage 5 then recognizes entities and events with a recognizer based on finite-state transducers (FSTs), and inserts the recognized objects into the appropriate slots in templates. This FST recognizer is based on hand-built regular expressions like the following (NG indicates Noun-Group and VG Verb-Group), which matches the first sentence of the news story above.

No.	Step	Description
1	**Tokens**	Tokenize input stream of characters
2	**Complex Words**	Multiword phrases, numbers, and proper names.
3	**Basic phrases**	Segment sentences into noun and verb groups
4	**Complex phrases**	Identify complex noun groups and verb groups
5	**Semantic Patterns**	Identify entities and events, insert into templates.
6	**Merging**	Merge references to the same entity or event

Figure 19.20 Levels of processing in FASTUS (Hobbs et al., 1997). Each level extracts a specific type of information which is then passed on to the next higher level.

```
NG(Company/ies) VG(Set-up) NG(Joint-Venture) with NG(Company/ies)
VG(Produce) NG(Product)
```

The result of processing these two sentences is the five draft templates (Fig. 19.21) that must then be merged into the single hierarchical structure shown in Fig. 19.19. The merging algorithm, after performing coreference resolution, merges two activities that are likely to be describing the same events.

#	Template/Slot	Value
1	RELATIONSHIP:	TIE-UP
	ENTITIES:	Bridgestone Co., a local concern, a Japanese trading house
2	ACTIVITY:	PRODUCTION
	PRODUCT:	"golf clubs"
3	RELATIONSHIP:	TIE-UP
	JOINT VENTURE:	"Bridgestone Sports Taiwan Co."
	AMOUNT:	NT$20000000
4	ACTIVITY:	PRODUCTION
	COMPANY:	"Bridgestone Sports Taiwan Co."
	STARTDATE:	DURING: January 1990
5	ACTIVITY:	PRODUCTION
	PRODUCT:	"iron and "metal wood" clubs"

Figure 19.21 The five partial templates produced by stage 5 of FASTUS. These templates are merged in stage 6 to produce the final template shown in Fig. 19.19 on page 437.

19.9 Summary

This chapter has explored techniques for extracting limited forms of semantic content from texts.

- **Relations among entities** can be extracted by pattern-based approaches, supervised learning methods when annotated training data is available, lightly supervised **bootstrapping** methods when small numbers of **seed tuples** or **seed patterns** are available, **distant supervision** when a database of relations is available, and **unsupervised** or **Open IE** methods.
- Reasoning about time can be facilitated by detection and normalization of **temporal expressions**.
- **Events** can be ordered in time using sequence models and classifiers trained on temporally- and event-labeled data like the **TimeBank corpus**.

- **Template-filling** applications can recognize stereotypical situations in texts and assign elements from the text to roles represented as **fixed sets of slots**.

Bibliographical and Historical Notes

The earliest work on information extraction addressed the template-filling task in the context of the Frump system (DeJong, 1982). Later work was stimulated by the U.S. government-sponsored MUC conferences (Sundheim 1991, Sundheim 1992, Sundheim 1993, Sundheim 1995). Early MUC systems like CIRCUS system (Lehnert et al., 1991) and SCISOR (Jacobs and Rau, 1990) were quite influential and inspired later systems like FASTUS (Hobbs et al., 1997). Chinchor et al. (1993) describe the MUC evaluation techniques.

Due to the difficulty of porting systems from one domain to another, attention shifted to machine learning approaches. Early supervised learning approaches to IE (Cardie 1993, Cardie 1994, Riloff 1993, Soderland et al. 1995, Huffman 1996) focused on automating the knowledge acquisition process, mainly for finite-state rule-based systems. Their success, and the earlier success of HMM-based speech recognition, led to the use of sequence labeling (HMMs: Bikel et al. 1997; MEMMs McCallum et al. 2000; CRFs: Lafferty et al. 2001), and a wide exploration of features (Zhou et al., 2005). Neural approaches followed from the pioneering results of Collobert et al. (2011), who applied a CRF on top of a convolutional net.

Progress in this area continues to be stimulated by formal evaluations with shared benchmark datasets, including the Automatic Content Extraction (ACE) evaluations of 2000-2007 on named entity recognition, relation extraction, and temporal expressions[1], the **KBP** (**Knowledge Base Population**) evaluations (Ji et al. 2010, Surdeanu 2013) of relation extraction tasks like **slot filling** (extracting attributes ('slots') like age, birthplace, and spouse for a given entity) and a series of SemEval workshops (Hendrickx et al., 2009).

KBP
slot filling

Semisupervised relation extraction was first proposed by Hearst (1992b), and extended by systems like AutoSlog-TS (Riloff, 1996), DIPRE (Brin, 1998), SNOWBALL (Agichtein and Gravano, 2000), and Jones et al. (1999). The distant supervision algorithm we describe was drawn from Mintz et al. (2009), who first used the term 'distant supervision' (which was suggested to them by Chris Manning) but similar ideas had occurred in earlier systems like Craven and Kumlien (1999) and Morgan et al. (2004) under the name *weakly labeled data*, as well as in Snow et al. (2005) and Wu and Weld (2007). Among the many extensions are Wu and Weld (2010), Riedel et al. (2010), and Ritter et al. (2013). Open IE systems include KNOWITALL Etzioni et al. (2005), TextRunner (Banko et al., 2007), and REVERB (Fader et al., 2011). See Riedel et al. (2013) for a universal schema that combines the advantages of distant supervision and Open IE.

Exercises

19.1 Acronym expansion, the process of associating a phrase with an acronym, can

[1] www.nist.gov/speech/tests/ace/

be accomplished by a simple form of relational analysis. Develop a system based on the relation analysis approaches described in this chapter to populate a database of acronym expansions. If you focus on English **Three Letter Acronyms** (TLAs) you can evaluate your system's performance by comparing it to Wikipedia's TLA page.

19.2 Acquire the CMU seminar corpus and develop a template-filling system by using any of the techniques mentioned in Section 19.8. Analyze how well your system performs as compared with state-of-the-art results on this corpus.

19.3 A useful functionality in newer email and calendar applications is the ability to associate temporal expressions connected with events in email (doctor's appointments, meeting planning, party invitations, etc.) with specific calendar entries. Collect a corpus of email containing temporal expressions related to event planning. How do these expressions compare to the kinds of expressions commonly found in news text that we've been discussing in this chapter?

19.4 For the following sentences, give FOL translations that capture the temporal relationships between the events.

1. When Mary's flight departed, I ate lunch.
2. When Mary's flight departed, I had eaten lunch.