CHAPTER

17 Information Extraction

I am the very model of a modern Major-General,
I've information vegetable, animal, and mineral,
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical...
Gilbert and Sullivan, *Pirates of Penzance*

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 the text entities. These are often binary relations like child-of, employment, part-whole, and geospatial relations. Relation extraction has close links to populating a relational database. Indeed, **knowledge graphs**, datasets of structured relational knowledge, are a common way that search engines present information to users.

knowledge graphs

event extraction

Next, we discuss three tasks related to *events*. **Event extraction** is 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*. **Event coreference** (Chapter 22) is needed to figure out which event mentions in a text refer to the same event; in our running example the two instances of *increase* and the phrase *the move* all refer to the same event.

temporal expression

To figure out *when* the events in a text happened we extract **temporal expressions** like days of the week (*Friday* and *Thursday*), relative expressions like *two days from now* or *next year* and times such as *3:30 P.M.*. These expressions must be **normalized** onto specific calendar dates or times of day to situate events in time. In

our sample task, this will allow us to link *Friday* to the time of United's announcement, and *Thursday* to the previous day's fare increase, and produce a timeline in which United's announcement follows the fare increase and American's announcement follows both of those events.

template filling

Finally, many texts describe recurring stereotypical events or situations. The task of **template filling** is to find such 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 from text elements through additional processing.

Our airline text is an example of this kind of stereotypical situation since airlines often raise fares and then wait to see if competitors follow along. In this situation, 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.

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

17.1 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, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [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 17.1 lists the 17 relations used in the ACE relation extraction evaluations and Fig. 17.2 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.

These relations correspond nicely to the model-theoretic notions we introduced in Chapter 15 to ground the meanings of the logical forms. That is, a relation consists of a set of ordered tuples over elements of a domain. In most standard information-extraction applications, the domain elements correspond to the named entities that occur in the text, to the underlying entities that result from coreference resolution, or to entities selected from a domain ontology. Figure 17.3 shows a model-based view of the set of entities and relations that can be extracted from our running example.

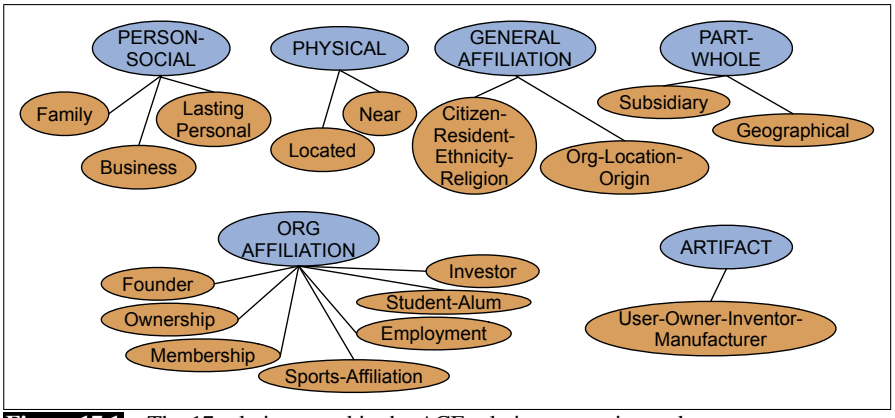

Figure 17.1 The 17 relations used in the ACE relation extraction task.

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

Domain	$\mathcal{D} = \{a,b,c,d,e,f,g,h,i\}$
United, UAL, American Airlines, AMR	a,b,c,d
Tim Wagner	e
Chicago, Dallas, Denver, and San Francisco	f,g,h,i
Classes	
United, UAL, American, and AMR are organizations	$Org = \{a,b,c,d\}$
Tim Wagner is a person	$Pers = \{e\}$
Chicago, Dallas, Denver, and San Francisco are places	$Loc = \{f,g,h,i\}$
Relations	
United is a unit of UAL American is a unit of AMR	$PartOf = \{\langle a,b\rangle, \langle c,d\rangle\}$
Tim Wagner works for American Airlines	$OrgAff = \{\langle c,e\rangle\}$
United serves Chicago, Dallas, Denver, and San Francisco	$Serves = \{\langle a,f\rangle, \langle a,g\rangle, \langle a,h\rangle, \langle a,i\rangle\}$

Figure 17.3 A model-based view of the relations and entities in our sample text.

Notice how this model-theoretic view subsumes the NER task as well; named entity recognition corresponds to the identification of a class of unary relations.

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:

(17.1) 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. 17.4. 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 17.4 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*).

17.2 Relation Extraction Algorithms

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

17.2.1 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**

$$NP_0 \text{ such as } NP_1\{,NP_2 \ldots, (and|or)NP_i\}, i \geq 1 \tag{17.2}$$

implies the following semantics

$$\forall NP_i, i \geq 1, \text{hyponym}(NP_i, NP_0) \tag{17.3}$$

allowing us to infer

$$\text{hyponym}(\text{Gelidium}, \text{red algae}) \tag{17.4}$$

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

Figure 17.5 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.

17.2.2 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. 17.6 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.

```
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 17.6 Finding and classifying the relations among entities in a text.

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:

(17.5) **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
- 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)

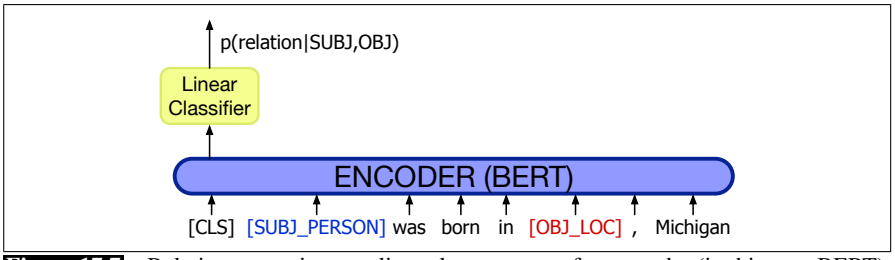

Figure 17.7 Relation extraction as a linear layer on top of an encoder (in this case BERT), with the subject and object entities replaced in the input by their NER tags (Zhang et al. 2017, Joshi et al. 2020).

- 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, showin in Fig. 17.7, 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 accuracies. 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.

17.2.3 Semisupervised Relation Extraction via 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 17.2.1, or perhaps a few **seed tuples**. That's enough

seed patterns
seed tuples

bootstrapping

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. 17.8 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 grep for more tuples
  newpairs ← *newpairs* with high confidence
  tuples ← *tuples* + *newpairs*
return tuples

Figure 17.8 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:

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

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

(17.8) 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:

(17.9) 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).

$$Conf_{RlogF}(p) = \frac{|hits(p)|}{|finds(p)|} log(|finds(p)|) \tag{17.10}$$

This metric is generally normalized to produce a probability.

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.

noisy-or

$$Conf(t) = 1 - \prod_{p \in P'} (1 - Conf(p)) \tag{17.11}$$

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.

17.2.4 Distant Supervision for Relation Extraction

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.

distant supervision

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, 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:

(17.12) **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. 17.9.

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 17.9 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.

17.2.5 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 | inf. marker)

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 oc-

cur 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. 17.10 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) ≤ 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 17.10 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:

(17.13) 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 these large sets of strings into some canonical form for adding to databases or other knowledge sources. Current methods focus heavily on relations expressed with verbs, and so will miss many relations that are expressed nominally.

17.2.6 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

$$\hat{P} = \frac{\text{\# of correctly extracted relation tuples in the sample}}{\text{total \# of extracted relation tuples in the sample.}} \tag{17.14}$$

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.

17.3 Extracting Times

Times and dates are a particularly important kind of named entity that play a role in question answering, in calendar and personal assistant applications. In order to reason about times and dates, after we extract these **temporal expressions** they must be **normalized**—converted to a standard format so we can reason about them. In this section we consider both the extraction and normalization of temporal expressions.

17.3.1 Temporal Expression Extraction

absolute
relative
duration

Temporal expressions are those 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 17.11 lists some sample temporal expressions in each of these categories.

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 17.11 Examples of absolute, relational and durational temporal expressions.

lexical triggers

Temporal expressions are grammatical constructions that have temporal **lexical triggers** as their heads. Lexical triggers might be nouns, proper nouns, adjectives,

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 17.12 Examples of temporal lexical triggers.

and adverbs; full temporal expressions consist of their phrasal projections: noun phrases, adjective phrases, and adverbial phrases. Figure 17.12 provides examples.

Let's look at the TimeML annotation scheme, in which temporal expressions are annotated with an XML tag, TIMEX3, and various attributes to that tag (Pustejovsky et al. 2005, Ferro et al. 2005). The following example illustrates the basic use of this scheme (we defer discussion of the attributes until Section 17.3.2).

> 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>.

The temporal expression recognition task consists of finding the start and end of all of the text spans that correspond to such temporal expressions. **Rule-based approaches** to temporal expression recognition use cascades of automata to recognize patterns at increasing levels of complexity. Tokens are first part-of-speech tagged, and then larger and larger chunks are recognized from the results from previous stages, based on patterns containing trigger words (e.g., *February*) or classes (e.g., *MONTH*). Figure 17.13 gives a fragment from a rule-based system.

```
# yesterday/today/tomorrow
$string =~ s/((($OT+the$CT+\s+)?$OT+day$CT+\s+$OT+(before|after)$CT+\s+)?$OT+$TERelDayExpr$CT+
(\s+$OT+(morning|afternoon|evening|night)$CT+)?)/<TIMEX$tever TYPE=\"DATE\">$1
<\/TIMEX$tever>/gio;

$string =~ s/($OT+\w+$CT+\s+)<TIMEX$tever TYPE=\"DATE\"[^>]*>($OT+(Today|Tonight)$CT+)
<\/TIMEX$tever>/$1$4/gso;

# this (morning/afternoon/evening)
$string =~ s/(($OT+(early|late)$CT+\s+)?$OT+this$CT+\s*$OT+(morning|afternoon|evening)$CT+)/
  <TIMEX$tever TYPE=\"DATE\">$1<\/TIMEX$tever>/gosi;
$string =~ s/(($OT+(early|late)$CT+\s+)?$OT+last$CT+\s*$OT+night$CT+)/<TIMEX$tever
   TYPE=\"DATE\">$1<\/TIMEX$tever>/gsio;
```

Figure 17.13 Perl fragment from the GUTime temporal tagging system in Tarsqi (Verhagen et al., 2005).

Sequence-labeling approaches follow the same IOB scheme used for named-entity tags, marking words that are either inside, outside or at the beginning of a TIMEX3-delimited temporal expression with the I, O, and B tags as follows:

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

Features are extracted from the token and its context, and a statistical sequence labeler is trained (any sequence model can be used). Figure 17.14 lists standard features used in temporal tagging.

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:

(17.15) *1984* tells the story of Winston Smith...

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

Feature	Explanation
Token	The target token to be labeled
Tokens in window	Bag of tokens in the window around a target
Shape	Character shape features
POS	Parts of speech of target and window words
Chunk tags	Base phrase chunk tag for target and words in a window
Lexical triggers	Presence in a list of temporal terms

Figure 17.14 Typical features used to train IOB-style temporal expression taggers.

17.3.2 Temporal Normalization

temporal normalization

Temporal normalization is the process of mapping a temporal expression to either a specific 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 but may also include information about start and end points. Normalized times are represented with the VALUE attribute from the ISO 8601 standard for encoding temporal values (ISO8601, 2004). Fig. 17.15 reproduces our earlier example with the value attributes added in.

```
<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 17.15 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. There is a lot more to the various temporal annotation standards—far too much to cover here. Figure 17.16 describes some of the basic ways that other times and durations are represented. Consult ISO8601 (2004), Ferro et al. (2005), and Pustejovsky et al. (2005) for more details.

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. As in the compositional

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 17.16 Sample ISO patterns for representing various times and durations.

rule-to-rule approach introduced in Chapter 16, the meaning of a constituent is computed from the meaning of its parts using a method specific to the constituent, although here the semantic composition rules involve temporal arithmetic rather than λ-calculus attachments.

fully qualified

Fully qualified date expressions contain a year, month, and day in some conventional form. The units in the expression must be detected and then placed in the correct place in the corresponding ISO pattern. The following pattern normalizes expressions like *April 24, 1916*.

$$FQTE \rightarrow Month\ Date\ ,\ Year \qquad \{Year.val - Month.val - Date.val\}$$

The non-terminals *Month*, *Date*, and *Year* represent constituents that have already been recognized and assigned semantic values, accessed through the **.val* notation. The value of this *FQE* constituent can, in turn, be accessed as *FQTE.val* during further processing.

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

temporal anchor. The values of temporal expressions such as *today*, *yesterday*, or *tomorrow* can all be computed with respect to this temporal 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.

(17.17) 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.

17.4 Extracting Events and their Times

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 14 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; for light verbs the event is often expressed by the nominal direct object (*took a flight*), and these light verbs just provide a syntactic structure for the noun's arguments.

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 sequence models with IOB tagging, and assigning event classes and attributes with multi-class classifiers. Feature-based models use surface information like parts of speech, lexical items, and verb tense information; see Fig. 17.17.

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 17.17 Features commonly used in both rule-based and machine learning approaches to event detection.

17.4.1 Temporal Ordering of Events

With both the events and the temporal expressions in a text having been detected, the next logical task is to use this information to fit the events into a complete timeline. Such a timeline would be useful for applications such as question answering and summarization. This ambitious task is the subject of considerable current research but 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 similar to those described earlier in Section 17.1. The temporal relation between events is classified into one of the standard set of **Allen relations** shown in Fig. 17.18 (Allen, 1984), using feature-based classifiers as in Section 17.1, trained on the TimeBank corpus with features like words/embeddings, parse paths, tense and aspect.

Allen relations

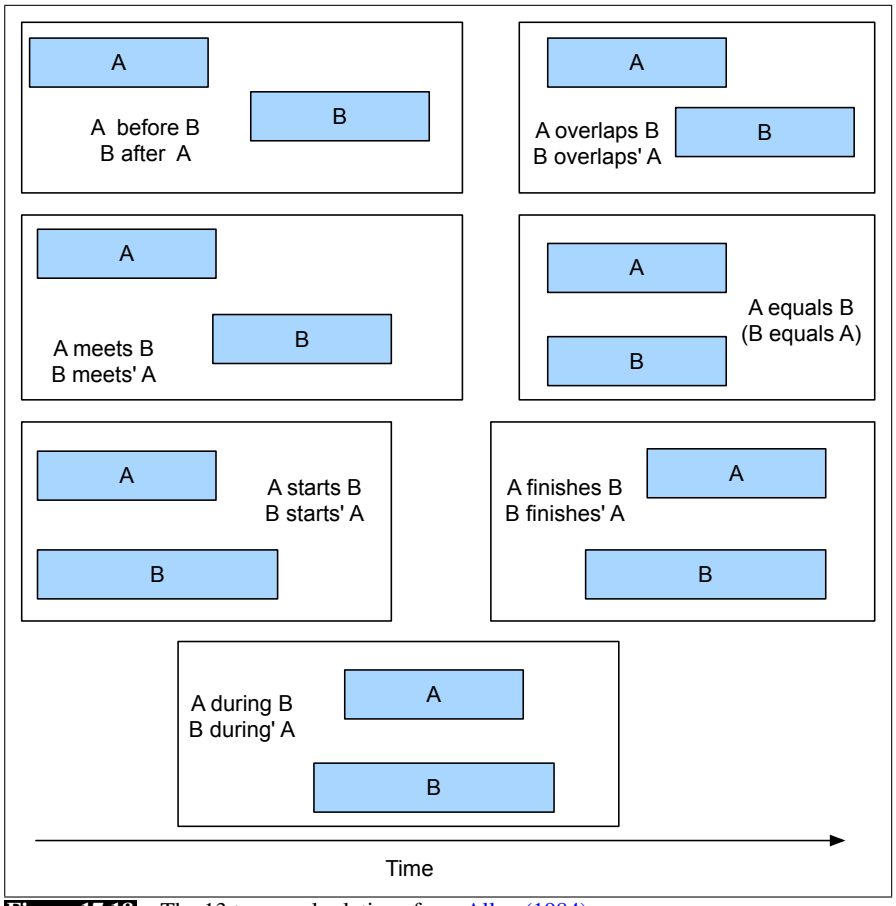

Figure 17.18 The 13 temporal relations from Allen (1984).

TimeBank

The **TimeBank** corpus consists of text annotated with much of the information we've been discussing throughout this section (Pustejovsky et al., 2003b). TimeBank 1.2 consists of 183 news articles selected from a variety of sources, including the Penn TreeBank and PropBank collections.

```
<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 17.19 Example from the TimeBank corpus.

Each article in the TimeBank corpus has had the temporal expressions and event mentions in them explicitly annotated in the TimeML annotation (Pustejovsky et al., 2003a). In addition to temporal expressions and events, the TimeML annotation provides temporal links between events and temporal expressions that specify the nature of the relation between them. Consider the following sample sentence and its corresponding markup shown in Fig. 17.19, selected from one of the TimeBank documents.

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

As annotated, this text includes three events and two temporal expressions. The events are all in the occurrence class and are given unique identifiers for use in further annotations. The temporal expressions include the creation time of the article, which serves as the document time, and a single temporal expression within the text.

In addition to these annotations, TimeBank provides four links that capture the temporal relations between the events and times in the text, using the Allen relations from Fig. 17.18. The following are the within-sentence temporal relations annotated for this example.

- 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$_{e3}$
- Declining$_{e4}$ **includes** soaring$_{e1}$

17.5 Template Filling

scripts

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.

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 17.5.2 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.

17.5.1 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.

template recognition

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.

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 22 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. Recent work focuses on 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).

17.5.2 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 17.20 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 17.20 The templates produced by FASTUS given the input text on page 20.

Early systems for dealing with these complex templates were based on cascades of transducers based on handwritten rules, as sketched in Fig. 17.21.

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 17.21 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.

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 FST-based recognizer 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.

```
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. 17.22) that must then be merged into the single hierarchical structure shown in Fig. 17.20. The merging algorithm, after performing coreference resolution, merges two activities that are likely to be describing the same events.

17.6 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

#	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 17.22 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. 17.20 on page 21.

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** through a combination of statistical learning and rule-based methods.
- **Events** can be detected and 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

KBP
slot filling

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).

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 coined the term 'distant supervision', 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.

HeidelTime (Strötgen and Gertz, 2013) and SUTime (Chang and Manning, 2012) are downloadable temporal extraction and normalization systems. The 2013 TempEval challenge is described in UzZaman et al. (2013); Chambers (2013) and Bethard (2013) give typical approaches.

Exercises

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

17.2 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?

17.3 Acquire the CMU seminar corpus and develop a template-filling system by using any of the techniques mentioned in Section 17.5. Analyze how well your system performs as compared with state-of-the-art results on this corpus.

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

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. *Proceedings of the 5th ACM International Conference on Digital Libraries*.

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence 23*(2), 123–154.

Banko, M., Cafarella, M., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction for the web. *IJCAI*.

Bethard, S. (2013). ClearTK-TimeML: A minimalist approach to TempEval 2013. *SemEval-13*.

Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: A high-performance learning namefinder. *ANLP*.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia—A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web 7*(3), 154–165.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *SIGMOD 2008*.

Brin, S. (1998). Extracting patterns and relations from the World Wide Web. *Proceedings World Wide Web and Databases International Workshop, Number 1590 in LNCS*. Springer.

Cardie, C. (1993). A case-based approach to knowledge acquisition for domain specific sentence analysis. *AAAI*.

Cardie, C. (1994). *Domain-Specific Knowledge Acquisition for Conceptual Sentence Analysis*. Ph.D. thesis, University of Massachusetts, Amherst, MA. Available as CMPSCI Technical Report 94-74.

Chambers, N. (2013). NavyTime: Event and time ordering from raw text. *SemEval-13*.

Chambers, N. and Jurafsky, D. (2011). Template-based information extraction without the templates. *ACL*.

Chang, A. X. and Manning, C. D. (2012). SUTime: A library for recognizing and normalizing time expressions. *LREC*.

Chinchor, N., Hirschman, L., and Lewis, D. L. (1993). Evaluating Message Understanding systems: An analysis of the third Message Understanding Conference. *Computational Linguistics 19*(3), 409–449.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *JMLR 12*, 2493–2537.

Craven, M. and Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. *ISMB-99*.

DeJong, G. F. (1982). An overview of the FRUMP system. Lehnert, W. G. and Ringle, M. H. (Eds.), *Strategies for Natural Language Processing*, 149–176. LEA.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence 165*(1), 91–134.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. *EMNLP*.

Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. (2005). Tides 2005 standard for the annotation of temporal expressions. Tech. rep., MITRE.

Grishman, R. and Sundheim, B. (1995). Design of the MUC-6 evaluation. *MUC-6*.

Hearst, M. A. (1992a). Automatic acquisition of hyponyms from large text corpora. *COLING*.

Hearst, M. A. (1992b). Automatic acquisition of hyponyms from large text corpora. *COLING*. COLING.

Hearst, M. A. (1998). Automatic discovery of WordNet relations. Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*. MIT Press.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2009). Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *5th International Workshop on Semantic Evaluation*.

Hobbs, J. R., Appelt, D. E., Bear, J., Israel, D., Kameyama, M., Stickel, M. E., and Tyson, M. (1997). FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. Roche, E. and Schabes, Y. (Eds.), *Finite-State Language Processing*, 383–406. MIT Press.

Huffman, S. (1996). Learning information extraction patterns from examples. Wertmer, S., Riloff, E., and Scheller, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning Natural Language Processing*, 246–260. Springer.

ISO8601 (2004). Data elements and interchange formats—information interchange—representation of dates and times. Tech. rep., International Organization for Standards (ISO).

Jacobs, P. S. and Rau, L. F. (1990). SCISOR: A system for extracting information from on-line news. *CACM 33*(11), 88–97.

Ji, H., Grishman, R., and Dang, H. T. (2010). Overview of the tac 2011 knowledge base population track. *TAC-11*.

Jones, R., McCallum, A., Nigam, K., and Riloff, E. (1999). Bootstrapping for text learning tasks. *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving pre-training by representing and predicting spans. *TACL 8*, 64–77.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lehnert, W. G., Cardie, C., Fisher, D., Riloff, E., and Williams, R. (1991). Description of the CIRCUS system as used for MUC-3. Sundheim, B. (Ed.), *MUC-3*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy Markov models for information extraction and segmentation. *ICML*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. *ACL IJCNLP*.

Morgan, A. A., Hirschman, L., Colosimo, M., Yeh, A. S., and Colombe, J. B. (2004). Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics 37*(6), 396–410.

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., and Katz, G. (2003a). TimeML: robust specification of event and temporal expressions in text. *Proceedings of the 5th International Workshop on Computational Semantics (IWCS-5)*.

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D. S., Ferro, L., and Lazo, M. (2003b). The TIMEBANK corpus. *Proceedings of Corpus Linguistics 2003 Conference*. UCREL Technical Paper number 16.

Pustejovsky, J., Ingria, R., Saurí, R., Castaño, J., Littman, J., Gaizauskas, R., Setzer, A., Katz, G., and Mani, I. (2005). *The Specification Language TimeML*, chap. 27. Oxford.

Riedel, S., Yao, L., and McCallum, A. (2010). Modeling relations and their mentions without labeled text. *Machine Learning and Knowledge Discovery in Databases*, 148–163. Springer.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. *NAACL HLT*.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. *AAAI*.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. *AAAI*.

Riloff, E. and Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. *AAAI*.

Ritter, A., Zettlemoyer, L., Mausam, and Etzioni, O. (2013). Modeling missing data in distant supervision for information extraction. *TACL 1*, 367–378.

Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum.

Snow, R., Jurafsky, D., and Ng, A. Y. (2005). Learning syntactic patterns for automatic hypernym discovery. *NeurIPS*.

Soderland, S., Fisher, D., Aseltine, J., and Lehnert, W. G. (1995). CRYSTAL: Inducing a conceptual dictionary. *IJCAI-95*.

Strötgen, J. and Gertz, M. (2013). Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation 47*(2), 269–298.

Sundheim, B. (Ed.). (1991). *Proceedings of MUC-3*.

Sundheim, B. (Ed.). (1992). *Proceedings of MUC-4*.

Sundheim, B. (Ed.). (1993). *Proceedings of MUC-5*, Baltimore, MD.

Sundheim, B. (Ed.). (1995). *Proceedings of MUC-6*.

Surdeanu, M. (2013). Overview of the TAC2013 Knowledge Base Population evaluation: English slot filling and temporal slot filling. *TAC-13*.

UzZaman, N., Llorens, H., Derczynski, L., Allen, J., Verhagen, M., and Pustejovsky, J. (2013). SemEval-2013 task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. *SemEval-13*.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Moszkowicz, J., and Pustejovsky, J. (2009). The TempEval challenge: Identifying temporal relations in text. *Language Resources and Evaluation 43*(2), 161–179.

Verhagen, M., Mani, I., Sauri, R., Knippen, R., Jang, S. B., Littman, J., Rumshisky, A., Phillips, J., and Pustejovsky, J. (2005). Automating temporal annotation with TARSQI. *ACL*.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledge base. *CACM 57*(10), 78–85.

Wu, F. and Weld, D. S. (2007). Autonomously semantifying Wikipedia. *CIKM-07*.

Wu, F. and Weld, D. S. (2010). Open information extraction using Wikipedia. *ACL*.

Zhang, Y., Zhong, V., Chen, D., Angeli, G., and Manning, C. D. (2017). Position-aware attention and supervised data improve slot filling. *EMNLP*.

Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring various knowledge in relation extraction. *ACL*.