

CHAPTER

21

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.

named entity
recognition

The first step in most IE tasks is to find the proper names or **named entities** mentioned in a text. The task of **named entity recognition** (NER) is to find each **mention** of a named entity in the text and label its type. What constitutes a named entity type is application specific; these commonly include people, places, and organizations but also more specific entities from the names of genes and proteins (Cohen and Demner-Fushman, 2014) to the names of college courses (McCallum, 2005).

Having located all of the mentions of named entities in a text, it is useful to link, or cluster, these mentions into sets that correspond to the entities behind the mentions, for example inferring that mentions of *United Airlines* and *United* in the sample text refer to the same real-world entity. We'll defer discussion of this task of **coreference resolution** until Chapter 23.

relation
extraction

The task of **relation extraction** is to find and classify semantic relations among the text entities, often binary relations like spouse-of, child-of, employment, part-whole, membership, and geospatial relations. Relation extraction has close links to populating a relational database.

event
extraction

The task of **event extraction** is to find 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*. We'll also need to perform **event coreference** to figure out which of the many 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'll do recognition of **temporal expressions** like days of the week (*Friday* and *Thursday*), months, holidays, etc., relative expressions like *two days from now* or *next year* and times such as *3:30 P.M.* or *noon*. The problem of **temporal expression normalization** is to map these temporal expressions 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 situations. The task of **template filling** is to find such situations in documents and fill the template slots with appropriate material. 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

The following sections review current approaches to each of these problems.

21.1 Named Entity Recognition

named entity

The first step in information extraction is to detect the **entities** in the text. A **named entity** is, roughly speaking, anything that can be referred to with a proper name: a person, a location, an organization. The term is commonly extended to include things that aren't entities per se, including dates, times, and other kinds of **temporal expressions**, and even numerical expressions like prices. Here's the sample text introduced earlier with the named entities marked:

temporal
expressions

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 contains 13 mentions of named entities including 5 organizations, 4 locations, 2 times, 1 person, and 1 mention of money.

In addition to their use in extracting events and the relationship between participants, named entities are useful for many other language processing tasks. In sentiment analysis we might want to know a consumer's sentiment toward a particular entity. Entities are a useful first stage in question answering, or for linking text to information in structured knowledge sources like wikipedia.

Figure 21.1 shows typical generic named entity types. Many applications will also need to use specific entity types like proteins, genes, commercial products, or works of art.

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	Turing is a giant of computer science.
Organization	ORG	companies, sports teams	The IPCC warned about the cyclone.
Location	LOC	regions, mountains, seas	The Mt. Sanitas loop is in Sunshine Canyon .
Geo-Political Entity	GPE	countries, states, provinces	Palo Alto is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the Tappan Zee Bridge .
Vehicles	VEH	planes, trains, automobiles	It was a classic Ford Falcon .

Figure 21.1 A list of generic named entity types with the kinds of entities they refer to.

Named entity recognition means finding spans of text that constitute proper names and then classifying the type of the entity. Recognition is difficult partly because of the ambiguity of segmentation; we need to decide what's an entity and what isn't, and where the boundaries are. Another difficulty is caused by type ambiguity. The mention JFK can refer to a person, the airport in New York, or any number of schools, bridges, and streets around the United States. Some examples of this kind of cross-type confusion are given in Figures 21.2 and 21.3.

Name	Possible Categories
<i>Washington</i>	Person, Location, Political Entity, Organization, Facility
<i>Downing St.</i>	Location, Organization
<i>IRA</i>	Person, Organization, Monetary Instrument
<i>Louis Vuitton</i>	Person, Organization, Commercial Product

Figure 21.2 Common categorical ambiguities associated with various proper names.

[PER Washington] was born into slavery on the farm of James Burroughs.
 [ORG Washington] went up 2 games to 1 in the four-game series.
 Blair arrived in [LOC Washington] for what may well be his last state visit.
 In June, [GPE Washington] passed a primary seatbelt law.
 The [FAC Washington] had proved to be a leaky ship, every passage I made...

Figure 21.3 Examples of type ambiguities in the use of the name *Washington*.

21.1.1 NER as Sequence Labeling

The standard algorithm for named entity recognition is as a word-by-word sequence labeling task, in which the assigned tags capture both the boundary and the type. A sequence classifier like an MEMM or CRF is trained to label the tokens in a text with tags that indicate the presence of particular kinds of named entities. Consider the following simplified excerpt from our running example.

[ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.

BIO Figure 21.4 shows the same excerpt represented with **BIO** tagging. In **BIO** tagging we introduce a tag for the beginning (B) and inside (I) of each entity type, and one for tokens outside (O) any entity. The number of tags is thus $2n + 1$ tags, where n is the number of entity types. **BIO** tagging can represent exactly the same information as the bracketed notation.

Words	BIO Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

Figure 21.4 Named entity tagging as a sequence model, showing **BIO** and **IO** encodings.

We've also shown **IO** tagging, which loses some information by eliminating the **B** tag. Without the **B** tag **IO** tagging is unable to distinguish between two entities of the same type that are right next to each other. Since this situation doesn't arise very often (usually there is at least some punctuation or other delimitator), **IO** tagging may be sufficient, and has the advantage of using only $n + 1$ tags.

Having encoded our training data with **IOB** tags, the next step is to select a set of features to associate with each input word token. Figure 21.5 lists standard features used in state-of-the-art systems.

word shape We've seen many of these features before in the context of part-of-speech tagging, particularly for tagging unknown words. This is not surprising, as many unknown words are in fact named entities. **Word shape** features are thus particularly important in the context of **NER**. Recall that **word shape** features are used to represent the abstract letter pattern of the word by mapping lower-case letters to 'x', upper-case to 'X', numbers to 'd', and retaining punctuation. Thus for example **I.M.F** would map to **X.X.X**, and **DC10-30** would map to **XXdd-dd**. A second class of shorter word shape features is also used. In these features consecutive character types are removed, so **DC10-30** would be mapped to **Xd-d** but **I.M.F** would still map to **X.X.X**. It turns out that this feature by itself accounts for a considerable part of the success of **NER** systems for English news text. Shape features are also particularly important in recognizing names of proteins and genes in biological texts.

identity of w_i
 identity of neighboring words
 part of speech of w_i
 part of speech of neighboring words
 base-phrase syntactic chunk label of w_i and neighboring words
 presence of w_i in a **gazetteer**
 w_i contains a particular prefix (from all prefixes of length ≤ 4)
 w_i contains a particular suffix (from all suffixes of length ≤ 4)
 w_i is all upper case
 word shape of w_i
 word shape of neighboring words
 short word shape of w_i
 short word shape of neighboring words
 presence of hyphen

Figure 21.5 Features commonly used in training named entity recognition systems.

For example the named entity token *L'Occitane* would generate the following non-zero valued feature values:

prefix(w_i) = L
 prefix(w_i) = L'
 prefix(w_i) = L'O
 prefix(w_i) = L'Oc
 suffix(w_i) = tane
 suffix(w_i) = ane
 suffix(w_i) = ne
 suffix(w_i) = e
 word-shape(w_i) = X'Xxxxxxxx
 short-word-shape(w_i) = X'Xx

gazetteer

A **gazetteer** is a list of place names, and they can offer millions of entries for all manner of locations along with detailed geographical, geologic, and political information.¹ In addition to gazetteers, the United States Census Bureau provides extensive lists of first names and surnames derived from its decadal census in the U.S.² Similar lists of corporations, commercial products, and all manner of things biological and mineral are also available from a variety of sources. Gazetteer features are typically implemented as a binary feature for each name list. Unfortunately, such lists can be difficult to create and maintain, and their usefulness varies considerably depending on the named entity class. It appears that gazetteers can be quite effective, while extensive lists of persons and organizations are not nearly as beneficial (Mikheev et al., 1999).

The relative usefulness of any of these features or combination of features depends to a great extent on the application, genre, media, language, and text encoding. For example, shape features, which are critical for English newswire texts, are of little use with materials transcribed from spoken text by automatic speech recognition, materials gleaned from informally edited sources such as blogs and discussion forums, and for character-based languages like Chinese where case information isn't available. The set of features given in Fig. 21.5 should therefore be thought of as only a starting point for any given application.

¹ www.geonames.org

² www.census.gov

Word	POS	Chunk	Short shape	Label
American	NNP	B-NP	Xx	B-ORG
Airlines	NNPS	I-NP	Xx	I-ORG
,	,	O	,	O
a	DT	B-NP	x	O
unit	NN	I-NP	x	O
of	IN	B-PP	x	O
AMR	NNP	B-NP	X	B-ORG
Corp.	NNP	I-NP	Xx.	I-ORG
,	,	O	,	O
immediately	RB	B-ADVP	x	O
matched	VBD	B-VP	x	O
the	DT	B-NP	x	O
move	NN	I-NP	x	O
,	,	O	,	O
spokesman	NN	B-NP	x	O
Tim	NNP	I-NP	Xx	B-PER
Wagner	NNP	I-NP	Xx	I-PER
said	VBD	B-VP	x	O
.	.	O	.	O

Figure 21.6 Word-by-word feature encoding for NER.

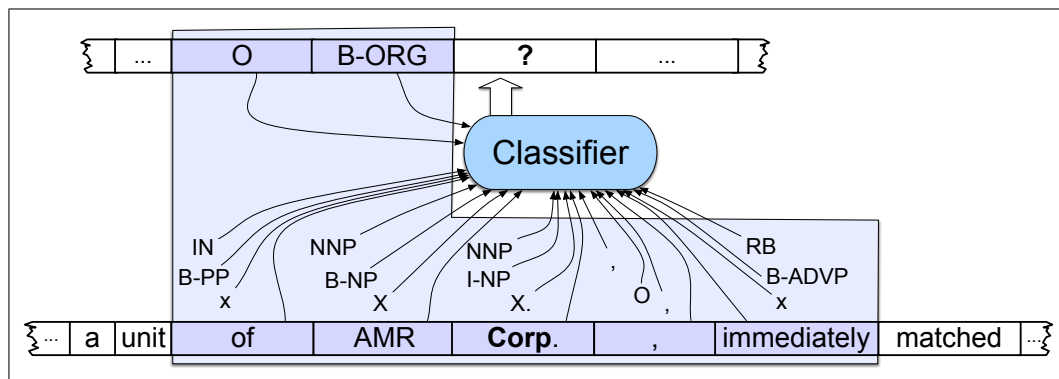


Figure 21.7 Named entity recognition as sequence labeling. The features available to the classifier during training and classification are those in the boxed area.

Figure 21.6 illustrates the result of adding part-of-speech tags, syntactic base-phrase chunk tags, and some shape information to our earlier example.

Given such a training set, a sequence classifier like an MEMM can be trained to label new sentences. Figure 21.7 illustrates the operation of such a sequence labeler at the point where the token *Corp.* is next to be labeled. If we assume a context window that includes the two preceding and following words, then the features available to the classifier are those shown in the boxed area.

21.1.2 Evaluation of Named Entity Recognition

The familiar metrics of **recall**, **precision**, and F_1 **measure** are used to evaluate NER systems. Remember that recall is the ratio of the number of correctly labeled responses to the total that should have been labeled; precision is the ratio of the number of correctly labeled responses to the total labeled; and F -measure is the harmonic

mean of the two. For named entities, the *entity* rather than the word is the unit of response. Thus in the example in Fig. 21.6, the two entities *Tim Wagner* and *AMR Corp.* and the non-entity *said* would each count as a single response.

The fact that named entity tagging has a segmentation component which is not present in tasks like text categorization or part-of-speech tagging causes some problems with evaluation. For example, a system that labeled *American* but not *American Airlines* as an organization would cause two errors, a false positive for O and a false negative for I-ORG. In addition, using entities as the unit of response but words as the unit of training means that there is a mismatch between the training and test conditions.

21.1.3 Practical NER Architectures

While pure statistical sequence models are the norm in academic research, commercial approaches to NER are often based on pragmatic combinations of lists, rules, and supervised machine learning (Chiticariu et al., 2013). One common approach is to make repeated passes over a text, allowing the results of one pass to influence the next. The stages typically first involve the use of rules that have extremely high precision but low recall. Subsequent stages employ more error-prone statistical methods that take the output of the first pass into account.

1. First, use high-precision rules to tag unambiguous entity mentions.
2. Then, search for substring matches of the previously detected names.
3. Consult application-specific name lists to identify likely name entity mentions from the given domain.
4. Finally, apply probabilistic sequence labeling techniques that make use of the tags from previous stages as additional features.

The intuition behind this staged approach is twofold. First, some of the entity mentions in a text will be more clearly indicative of a given entity's class than others. Second, once an unambiguous entity mention is introduced into a text, it is likely that subsequent shortened versions will refer to the same entity (and thus the same type of entity).

21.2 Relation Extraction

Next on our list of tasks is to discern the relationships that exist among the detected entities. Let's return to our sample airline text:

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

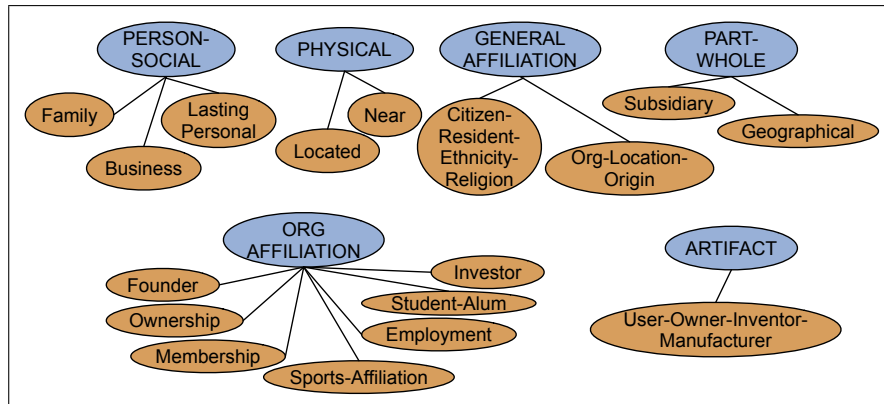


Figure 21.8 The 17 relations used in the ACE relation extraction task.

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

employs that are fairly frequent in news-style texts. Figure 21.8 lists the 17 relations used in the ACE relation extraction evaluations and Fig. 21.9 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 19 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 co-reference resolution, or to entities selected from a domain ontology. Figure 21.10 shows a model-based view of the set of entities and relations that can be extracted from our running example. 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:

(21.1) 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:

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	$PartOf = \{\langle a, b \rangle, \langle c, d \rangle\}$
American is a unit of AMR	
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 21.10 A model-based view of the relations and entities in our sample text.

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 = "John L. Hennessy"`. 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), has relations like

Freebase

```

people/person/nationality
location/location/contains
people/person/place-of-birth
biology/organism_higher_classification

```

Another kind of useful relation are ontological relations from WordNet or other ontologies 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 is-a animal...

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 ontologies or building them for new languages or domains.

There are four main classes of algorithms for relation extraction: **hand-written patterns**, **supervised machine learning**, **semi-supervised**, and **unsupervised**. We'll introduce each of these in the next four sections.

21.2.1 Using Patterns to Extract Relations

The earliest and still a common algorithm for relation extraction is the use of lexico-syntactic patterns, first developed by Hearst (1992a). 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**

$$NP_0 \text{ such as } NP_1 \{, NP_2 \dots, (and|or) NP_i\}, i \geq 1 \quad (21.2)$$

implies the following semantics

$$\forall NP_i, i \geq 1, \text{hyponym}(NP_i, NP_0) \quad (21.3)$$

allowing us to infer

$$\text{hyponym}(\text{Gelidium}, \text{red algae}) \quad (21.4)$$

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 <i>Gelidium</i>
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 21.11 Hand-built lexico-syntactic patterns for finding hypernyms, using { } to mark optionality (Hearst, 1992a, 1998).

Figure 21.11 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.

21.2.2 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 has three steps, illustrated in Fig. 21.12. Step one is to find pairs of named entities (usually in the same sentence). In step two, a filtering classifier is trained to make a binary decision as to whether a given pair of named entities are related (by any relation). Positive examples are extracted directly from all relations in the annotated corpus, and negative examples are generated from within-sentence entity pairs that are not annotated with a relation. In step 3, a classifier is trained to assign a label to the relations that were found by step 2. The use of the filtering classifier can speed up the final classification and also allows the use of distinct feature-sets appropriate for each task. For each of the two classifiers, we can use any of the standard classification techniques described in Chapter 7 (logistic regression, SVM, naive bayes, random forest, etc.).

```

function FINDRELATIONS(words) returns relations

    relations ← nil
    entities ← FINDENTITIES(words)
    forall entity pairs  $\langle e1, e2 \rangle$  in entities do
        if RELATED?(e1, e2)
            relations ← relations + CLASSIFYRELATION(e1, e2)

```

Figure 21.12 Finding and classifying the relations among entities in a text.

As with named entity recognition, the most important step in this process is to identify useful surface features that will be useful for relation classification. Let's look at some common features in the context of classifying the relationship between *American Airlines* (Mention 1, or M1) and *Tim Wagner* (Mention 2, M2) from this sentence:

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

Useful word features include

- 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
- Stemmed versions of the same

Useful named entity features include

- 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)

Finally, the **syntactic structure** of a sentence can signal many of the relationships among its entities. One simple and effective way to featurize a structure is to use strings representing **syntactic paths**: the path traversed through the tree in getting from one to the other. Constituency or dependency paths can both be helpful.

- Base syntactic chunk sequence from M1 to M2
NP NP PP VP NP NP
- Constituent paths between M1 and M2
NP ↑ NP ↑ S ↑ S ↓ NP
- Dependency-tree paths
Airlines ←_{subj} matched ←_{comp} said →_{subj} Wagner

Figure 21.13 summarizes many of the features we have discussed that could be used for classifying the relationship between *American Airlines* and *Tim Wagner* from our example text.

M1 headword	<i>airlines</i>
M2 headword	<i>Wagner</i>
Word(s) before M1	NONE
Word(s) after M2	<i>said</i>
Bag of words between	{ <i>a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman</i> }
M1 type	ORG
M2 type	PERS
Concatenated types	ORG-PERS
Constituent path	<i>NP ↑ NP ↑ S ↑ S ↓ NP</i>
Base phrase path	<i>NP → NP → PP → NP → VP → NP → NP</i>
Typed-dependency path	<i>Airlines ←_{subj} matched ←_{comp} said →_{subj} Wagner</i>

Figure 21.13 Sample of features extracted during classification of the <American Airlines, Tim Wagner> tuple; M1 is the first mention, M2 the second.

Supervised systems can get high accuracies with enough hand-labeled training data, if the test set is similar enough to the training set. But labeling a large training set is extremely expensive and supervised models are brittle: they don't generalize well to different genres.

21.2.3 Semisupervised Relation Extraction via Bootstrapping

Supervised machine learning assumes that we have a large collection of previously annotated material with which to train classifiers. Unfortunately, such collections are hard to come by.

seed patterns
seed tuples
bootstrapping

But suppose we just have a few high-precision **seed patterns**, like those in Section 21.2.1, 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. 21.14 sketches a basic algorithm.

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:

```

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 seeds
    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 21.14 Bootstrapping from seed entity pairs to learn relations.

- (21.6) Budget airline Ryanair, which uses Charleroi as a hub, scrapped all weekend flights out of the airport.
- (21.7) All flights in and out of Ryanair’s Belgian hub at Charleroi airport were grounded on Friday...
- (21.8) 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:

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

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

$$Conf_{RlogF}(p) = \frac{hits_p}{finds_p} \times \log(finds_p) \quad (21.10)$$

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 $1 - \text{Conf}(p)$; 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.

$$\text{Conf}(t) = 1 - \prod_{p \in P'} (1 - \text{Conf}(p)) \quad (21.11)$$

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.

21.2.4 Distant Supervision for Relation Extraction

Although text that has been hand-labeled with relation labels is extremely expensive to produce, there are ways to find indirect sources of training data.

distant
supervision

The **distant supervision** (Mintz et al., 2009) method 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. As with supervised relation extraction, we use 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:

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

we would learn rich conjunction features like this one:

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

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. 21.15.

```
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 21.15 The distant supervision algorithm for relation extraction.

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:

```
NPH like NP    Many hormones like leptin...
NPH called NP  ...using a markup language called XHTML
NP is a NPH   Ruby is a programming language...
NP, a NPH    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.

But 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.

21.2.5 Unsupervised Relation 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).

Open
Information
Extraction

ReVerb

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 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 labelling 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. 21.16 shows some sample features used in the classification.

For example the following sentence:

(x,r,y) covers all words in s
 the last preposition in r is *for*
 the last preposition in r is *on*
 $\text{len}(s) \leq 10$
 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 21.16 Features for the classifier that assigns confidence to relations extracted by the Open Information Extraction system REVERB (Fader et al., 2011).

(21.13) 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.

21.2.6 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

$$\hat{P} = \frac{\# \text{ of correctly extracted relation tuples in the sample}}{\text{total } \# \text{ of extracted relation tuples in the sample.}} \quad (21.14)$$

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.

21.3 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.

21.3.1 Temporal Expression Extraction

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 21.17 lists some sample temporal expressions in each of these categories.

Absolute
temporal
expressions
Relative
temporal
expressions
Durations

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 21.17 Examples of absolute, relational and durational temporal expressions.

Temporal expressions are grammatical constructions that have temporal **lexical triggers** as their heads. Lexical triggers might be nouns, proper nouns, adjectives, and adverbs; full temporal expression consist of their phrasal projections: noun phrases, adjective phrases, and adverbial phrases. Figure 21.18 provides examples.

lexical triggers

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

Figure 21.18 Examples of temporal lexical triggers.

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 21.3.2).

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 21.19 gives a small representative fragment from a rule-based system written in Perl.

```
# yesterday/today/tomorrow
$string = ` s/((($OT+(early|earlier|later?)$CT+\s+)?(($OT+the$CT+\s+)?$OT+day$CT+\s+
$OT+(before|after)$CT+\s+)?$OT+$TERelDayExpr$CT+(\s+$OT+(morning|afternoon|
evening|night)$CT+?)/<TIMEX2 TYPE=\ "DATE\">$1</TIMEX2>/gio;

$string = ` s/($OT+\w+$CT+\s+)
<TIMEX2 TYPE=\ "DATE\" [^>]*>($OT+(Today|Tonight)$CT+)</TIMEX2>/$1$2/gso;

# this/that (morning/afternoon/evening/night)
$string = ` s/((($OT+(early|earlier|later?)$CT+\s+)?$OT+(this|that|every|the$CT+\s+
$OT+(next|previous|following))$CT+\s*$OT+(morning|afternoon|evening|night)
$CT+(\s+$OT+thereafter$CT+?)/<TIMEX2 TYPE=\ "DATE\">$1</TIMEX2>/gosi;
```

Figure 21.19 Fragment of Perl code from MITRE’s TempEx temporal tagging system.

Sequence-labeling approaches follow the same BIO 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 B, I, and O 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 21.20 lists standard features used in temporal tagging.

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

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:

(21.15) *1984* tells the story of Winston Smith...

(21.16) ...U2’s classic *Sunday Bloody Sunday*

21.3.2 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. 21.21 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 UAL Corp's 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 21.21 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 Pnx, 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 21.22 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.

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 21.22 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. As in the compositional rule-to-rule approach introduced in Chapter 20, the meaning of a constituent is computed from the meaning of its parts using a method specific to the constituent, al-

though here the semantic composition rules involve temporal arithmetic rather than λ -calculus attachments.

Fully qualified
date expressions

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*.

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

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 *FQTE* constituent can, in turn, be accessed as *FQTE.val* during further processing.

temporal
anchor

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 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.

(21.17) 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.

21.4 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 18 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. In these cases, the verb is simply providing a syntactic structure for the arguments to an event expressed by the direct object as in *took a flight*.

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).

Event extraction is generally modeled via machine learning, detecting events via sequence models with BIO tagging, and assigning event classes and attributes with multi-class classifiers. Common features include surface information like parts of speech, lexical items, and verb tense information; see Fig. 21.23.

reporting
events

Feature	Explanation
Character affixes	Character-level prefixes and suffixes of target word
Nominalization suffix	Character level suffixes for nominalizations (e.g., <i>-tion</i>)
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 21.23 Features commonly used in both rule-based and statistical approaches to event detection.

21.4.1 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 21.2. One common approach to this problem is to operationalize it by attempting to identify which of Allen’s temporal relations shown in Fig. 21.24 hold between events. Most systems employ statistical classifiers of the kind discussed earlier in Section 21.2, trained on the TimeBank corpus and using features like words, parse paths, tense and aspect.

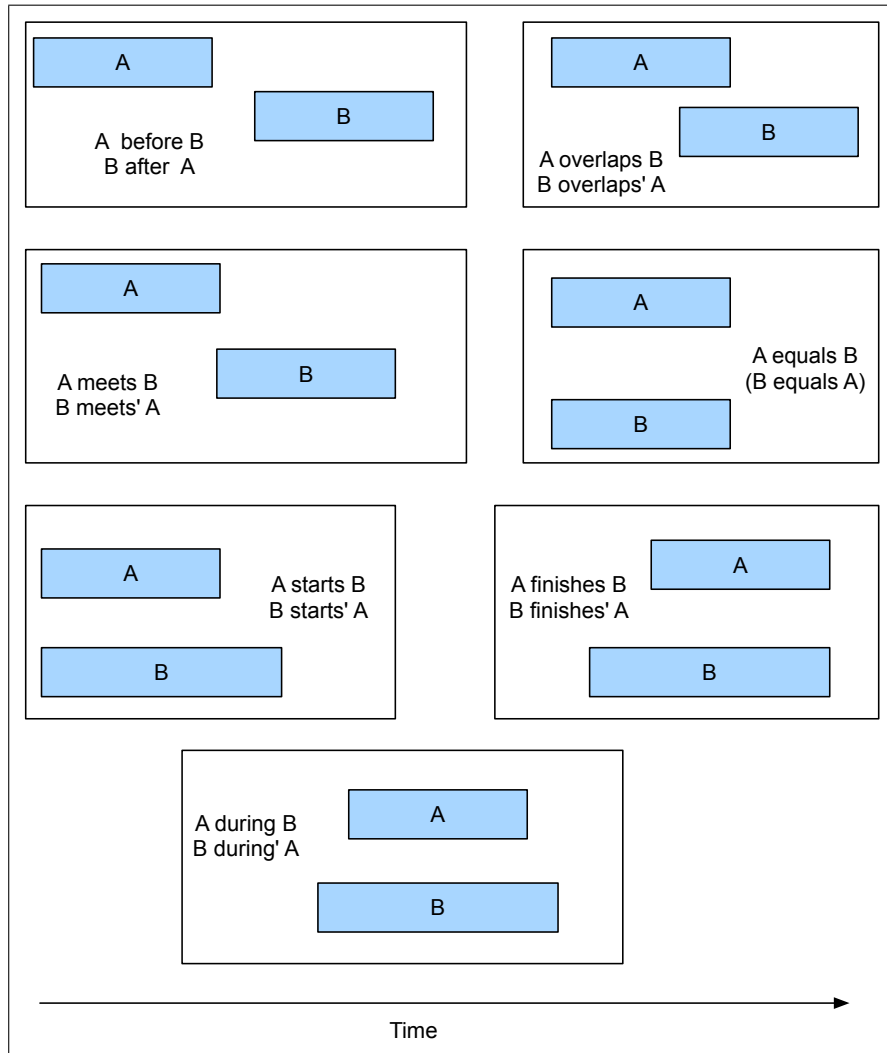


Figure 21.24 Allen’s 13 possible temporal relations.

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.

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

```

<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 21.25 Example from the TimeBank corpus.

nature of the relation between them. Consider the following sample sentence and its corresponding markup shown in Fig. 21.25, selected from one of the TimeBank documents.

(21.18) 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. 21.24. 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}

21.5 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.

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 21.5.2 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.

21.5.1 Statistical Approaches to Template Filling

The standard paradigm for template filling assumes we are trying to fill fixed known templates with known slots, and also assumes we are given training documents labeled with examples of each template, with the fillers of each slot marked in the text. The template filling task is then creation of one template for each event in the input documents, with the slots filled with text from the document.

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, 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 23 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).

21.5.2 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 21.26 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. a local concern a Japanese trading house	PRODUCT	iron and “metal wood” clubs
JOINT VENTURE	Bridgestone Sports Taiwan Co.	START DATE	DURING: January 1990
ACTIVITY	Activity-1		
AMOUNT	NT\$20000000		

Figure 21.26 The templates produced by FASTUS given the input text on page 25.

Early systems for dealing with these complex templates were based on cascades of transducers based on hand-written rules, as sketched in Fig. 21.27.

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 21.27 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 hand-written 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. 21.28) that must then be merged into the single hierarchical structure shown in Fig. 21.26. The merging algorithm, after performing coreference resolution, merges two activities that are likely to be describing the same events.

21.6 Summary

This chapter has explored a series of techniques for extracting limited forms of semantic content from texts. Most techniques can be characterized as problems in

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

Figure 21.28 The five partial templates produced by Stage 5 of the FASTUS system. These templates will be merged by the Stage 6 merging algorithm to produce the final template shown in Fig. 21.26 on page 25.

detection followed by classification.

- **Named entities** can be recognized and classified by sequence labeling techniques.
- **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** 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 and was performed in the context of the Frump system (DeJong, 1982). Later work was stimulated by the U.S. government-sponsored MUC conferences (Sundheim, 1991, 1992, 1993, 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 reusing or porting systems from one domain to another, attention shifted to automatic knowledge acquisition. The earliest supervised learning approaches to IE are described in Cardie (1993), Cardie (1994), Riloff (1993), Soderland et al. (1995), Huffman (1996), and Freitag (1998). These early learning

efforts focused on automating the knowledge acquisition process for mostly finite-state rule-based systems. Their success, and the earlier success of HMM-based methods for automatic speech recognition, led to the development of statistical systems based on sequence labeling. Early efforts applying HMMs to IE problems include [Bikel et al. \(1997, 1999\)](#) and [Freitag and McCallum \(1999\)](#). Subsequent efforts demonstrated the effectiveness of a range of statistical methods including MEMMs ([McCallum et al., 2000](#)), CRFs ([Lafferty et al., 2001](#)), and SVMs ([Sasano and Utsuro, 2000](#); [McNamee and Mayfield, 2002](#)). [Zhou et al. \(2005\)](#) explored different features for relation extraction. Progress in this area continues to be stimulated by formal evaluations with shared benchmark datasets. In the US, after the MUC evaluations of the mid-1990s the Automatic Content Extraction (ACE) evaluations of 2000-2007 focused on named entity recognition, relation extraction, and temporal expression extraction and normalization.³ These were followed by the **KBP (Knowledge Base Population)** evaluations ([Ji et al. 2010b](#), [Ji et al. 2010a](#), [Surdeanu 2013](#)) which included relation extraction tasks like **slot filling** (extracting values of attributes ('slots') like age, birthplace, and spouse for a given entity from text corpora). In addition, a new task was defined, **entity linking**, linking mentions of entities to their unique records in a database like Wikipedia; we return to entity linking in Chapter 23.

KBP
slot filling
entity linking

Semisupervised relation extraction was first proposed by [Hearst \(1992b\)](#). Important extensions included systems like DIPRE ([Brin, 1998](#)), and SNOWBALL ([Agichtein and Gravano, 2000](#)). The distant supervision algorithm we describe was drawn from [Mintz et al. \(2009\)](#), where the term 'distant supervision' was first defined, but similar ideas 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

- 21.1 Develop a set of regular expressions to recognize the character shape features described in Fig. ??.
- 21.2 The IOB labeling scheme given in this chapter isn't the only possible one. For example, an E tag might be added to mark the end of entities, or the B tag can be reserved only for those situations where an ambiguity exists between adjacent entities. Propose a new set of IOB tags for use with your NER system. Experiment with it and compare its performance with the scheme presented in this chapter.

³ www.nist.gov/speech/tests/ace/

- 21.3** Names of works of art (books, movies, video games, etc.) are quite different from the kinds of named entities we've discussed in this chapter. Collect a list of names of works of art from a particular category from a Web-based source (e.g., gutenberg.org, amazon.com, imdb.com, etc.). Analyze your list and give examples of ways that the names in it are likely to be problematic for the techniques described in this chapter.
- 21.4** Develop an NER system specific to the category of names that you collected in the last exercise. Evaluate your system on a collection of text likely to contain instances of these named entities.
- 21.5** 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.
- 21.6** 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?
- 21.7** Acquire the CMU seminar corpus and develop a template-filling system by using any of the techniques mentioned in Section 21.5. Analyze how well your system performs as compared with state-of-the-art results on this corpus.

- Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*.
- Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction for the web. In *IJCAI*, Vol. 7, pp. 2670–2676.
- Bethard, S. (2013). ClearTK-TimeML: A minimalist approach to TempEval 2013. In *SemEval-13*, pp. 10–14.
- Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: A high-performance learning name-finder. In *ANLP 1997*, pp. 194–201.
- Bikel, D. M., Schwartz, R., and Weischedel, R. (1999). An algorithm that learns what's in a name. *Machine Learning*, 34, 211–231.
- 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. In *SIGMOD 2008*, pp. 1247–1250.
- Brin, S. (1998). Extracting patterns and relations from the World Wide Web. In *Proceedings World Wide Web and Databases International Workshop, Number 1590 in LNCS*, pp. 172–183. Springer.
- Cardie, C. (1993). A case-based approach to knowledge acquisition for domain specific sentence analysis. In *AAAI-93*, pp. 798–803. AAAI Press.
- 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. In *SemEval-13*, pp. 73–77.
- Chambers, N. and Jurafsky, D. (2011). Template-based information extraction without the templates. In *ACL 2011*.
- Chang, A. X. and Manning, C. D. (2012). SUTime: A library for recognizing and normalizing time expressions.. In *LREC-12*, pp. 3735–3740.
- 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.
- Chiticariu, L., Li, Y., and Reiss, F. R. (2013). Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!. In *EMNLP 2013*, pp. 827–832.
- Cohen, K. B. and Demner-Fushman, D. (2014). *Biomedical natural language processing*. Benjamins.
- Craven, M. and Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. In *ISMB-99*, pp. 77–86.
- DeJong, G. F. (1982). An overview of the FRUMP system. In Lehnert, W. G. and Ringle, M. H. (Eds.), *Strategies for Natural Language Processing*, pp. 149–176. Lawrence Erlbaum.
- 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. In *EMNLP-11*, pp. 1535–1545.
- Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. (2005). Tides 2005 standard for the annotation of temporal expressions. Tech. rep., MITRE.
- Freitag, D. (1998). Multistrategy learning for information extraction. In *ICML 1998*, Madison, WI, pp. 161–169.
- Freitag, D. and McCallum, A. (1999). Information extraction using HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Retrieval*.
- Grishman, R. and Sundheim, B. (1995). Design of the MUC-6 evaluation. In *MUC-6*, San Francisco, pp. 1–11.
- Hearst, M. A. (1992a). Automatic acquisition of hyponyms from large text corpora. In *COLING-92*, Nantes, France.
- Hearst, M. A. (1992b). Automatic acquisition of hyponyms from large text corpora. In *COLING-92*, Nantes, France. COLING.
- Hearst, M. A. (1998). Automatic discovery of WordNet relations. In Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*. MIT Press.
- 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. In Roche, E. and Schabes, Y. (Eds.), *Finite-State Language Processing*, pp. 383–406. MIT Press.
- Huffman, S. (1996). Learning information extraction patterns from examples. In Wertmer, S., Riloff, E., and Scheller, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning Natural Language Processing*, pp. 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. *Communications of the ACM*, 33(11), 88–97.
- Ji, H., Grishman, R., and Dang, H. T. (2010a). Overview of the tac 2011 knowledge base population track. In *TAC-11*.
- Ji, H., Grishman, R., Dang, H. T., Griffith, K., and Ellis, J. (2010b). Overview of the tac 2010 knowledge base population track. In *TAC-10*.
- Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, Stanford, CA.
- Lehnert, W. G., Cardie, C., Fisher, D., Riloff, E., and Williams, R. (1991). Description of the CIRCUS system as used for MUC-3. In Sundheim, B. (Ed.), *MUC-3*, pp. 223–233.
- McCallum, A. (2005). Information extraction: Distilling structured data from unstructured text. *ACM Queue*, 3(9), 48–57.

- McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy Markov models for information extraction and segmentation. In *ICML 2000*, pp. 591–598.
- McNamee, P. and Mayfield, J. (2002). Entity extraction without language-specific resources. In *CoNLL-02*, Taipei, Taiwan.
- Mikheev, A., Moens, M., and Grover, C. (1999). Named entity recognition without gazetteers. In *EACL-99*, Bergen, Norway, pp. 1–8.
- Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *ACL IJCNLP 2009*.
- 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. In *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. In *Proceedings of Corpus Linguistics 2003 Conference*, pp. 647–656. 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. In *Machine Learning and Knowledge Discovery in Databases*, pp. 148–163. Springer.
- Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *NAACL HLT 2013*.
- Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *AAAI-93*, Washington, D.C., pp. 811–816.
- Riloff, E. and Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI-99*, pp. 474–479.
- Ritter, A., Zettlemoyer, L., Mausam, and Etzioni, O. (2013). Modeling missing data in distant supervision for information extraction. *TACL*, 1, 367–378.
- Sassano, M. and Utsuro, T. (2000). Named entity chunking techniques in supervised learning for Japanese named entity recognition. In *COLING-00*, Saarbrücken, Germany, pp. 705–711.
- 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. In Saul, L. K., Weiss, Y., and Bottou, L. (Eds.), *NIPS 17*, pp. 1297–1304. MIT Press.
- Soderland, S., Fisher, D., Aseltine, J., and Lehnert, W. G. (1995). CRYSTAL: Inducing a conceptual dictionary. In *IJCAI-95*, Montreal, pp. 1134–1142.
- 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. In *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. In *SemEval-13*, pp. 1–9.
- 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.
- Wu, F. and Weld, D. S. (2007). Autonomously semantifying Wikipedia. In *CIKM-07*, pp. 41–50.
- Wu, F. and Weld, D. S. (2010). Open information extraction using Wikipedia. In *ACL 2010*, pp. 118–127.
- Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring various knowledge in relation extraction. In *ACL-05*, Ann Arbor, MI, pp. 427–434.