22 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 hikes and the behavior of their stocks on the following day. Now historical data about stock prices is easy to come by, but what about the information about airline announcements? To do a reasonable job on this task, you would 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, this information resides in archives of news articles reporting on airlines actions, as in the following recent example.

> 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.

Now while distilling information like names, dates, amounts and the relations among them from typical news articles is a pretty trivial task, plowing through tens of thousands of articles would likely get a little tedious. Clearly this is exactly the kind of thing that we'd like to turn over to a computer for automatic processing. That is, we'd like to have programs that can quickly scan through documents and accurately extract superficial semantic information from them.

INFORMATION EXTRACTION

This process of **information extraction** (IE) turns the unstructured seman-

tic information hidden in texts into structured data. More concretely, information extraction is an effective way to to populate the contents of a relational database. Once the information is encoded formally, we are free to use all the computational mechanisms provided by database systems, statistical analysis packages and other forms of decision support systems to solve whatever we're addressing.

As we proceed through this chapter, we'll see that robust solutions to IE problems are really clever combinations of techniques we've seen earlier in the book. In particular, the finite-state methods described in Chs. 2 and 3, the probabilistic models introduced in Chs. 4 and 6 and the syntactic chunking methods from Ch. 13 form the core of most current approaches to information extraction. Before diving into the details of how these techniques are applied, let's quickly introduce the major problems in IE and how they are approached.

NAMED ENTITY RECOGNITION

The first step in most information extraction tasks is to detect and classify all the proper names mentioned in a text; a task generally referred to as **named entity recognition** (NER). Not surprisingly, what constitutes a proper name and the particular scheme used to classify them is application-specific. Generic NER systems tend to focus on finding the names of people, places and organizations that are mentioned in ordinary news texts; practical applications have also been built to detect everything from the names of genes and proteins (Settles, 2005) to the names of college courses (McCallum, 2005).

NAMED ENTITY MENTIONS

Our introductory example contains 13 instances of proper names, which we'll refer to as **named entity mentions**, that can be classified as either organizations, people, places, times or amounts.

Having located all 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. This is the task of **reference resolution**, which we introduced in Ch. 21, and is also an important component in IE.

In our sample text, we would like to know that the *United Airlines* mention in the first sentence and the *United* mention in the third sentence refer to the same real world entity. This general reference resolution problem also includes anaphora resolution as a sub-problem. In this case, determining that the two uses of *it* refer to *United Airlines* and *United* respectively.

RELATION DETECTION AND CLASSIFICATION

The task of **relation detection and classification** is to find and classify semantic relations among the entities discovered in a given text. In most practical settings, the focus of relation detection is on small fixed sets of binary relations. Generic relations that appear in standard system evaluations include family, employment, part-whole, membership, and geospatial relations. The relation detection and classification task is the one that most closely corresponds to the problem of populating a relational database. Relation detection among entities is also closely related to the problem of discovering semantic relations among words in-

troduced in Ch. 20.

Our sample text contains 3 explicit mention of generic relations: *United* is a part of *UAL*, *American Airlines* is a part of *AMR* and *Tim Wagner* is an employee of *American Airlines*. Domain specific relations from the airline industry would include the fact that *United* serves *Chicago*, *Dallas*, *Denver* and *San Francisco*.

EVENT DETECTION AND CLASSIFICATION

In addition to knowing about the entities in a text and their relation to one another, we might like to find and classify the events in which the entities are participating; this is the problem of **event detection and classification**. In our sample text, the key events are the fare increase by *United* and the ensuing increase by *American*. In addition, there are several events reporting these main events as indicated by the two uses of *said* and the use of *cite*. As with entity processing, event detection brings with it the problem of reference resolution; we need to figure out which of the many event mentions in a text refer to the same event. In our running example, the events referred to as *the move* and *the increase* in the second and third sentences are the same as the *increase* in the first sentence.

TEMPORAL EXPRESSION DETECTION

TEMPORAL ANALYSIS

The problem of figuring out when the events in a text happened and how they relate to each other in time raises the twin problems of **temporal expression detection** and **temporal analysis**. Temporal expression detection tells us that our sample text contains the temporal expressions *Friday* and *Thursday*. Temporal expressions include date expressions such as days of the week, months, holidays, etc, as well as indirect expressions including phrases like *two days from now* or *next year*. They also include expressions for clock times such as *noon* or *3:30PM*.

The overall problem of **temporal analysis** is to map temporal expressions onto specific calendar dates or times of day and then to use those times to situate events in time. It includes the following subtasks:

- Fixing the temporal expressions with respect to an anchoring date or time, typically the dateline of the story in the case of news stories;
- Associating temporal expressions with the events in the text;
- Arranging the events into a complete and coherent timeline.

In our sample text, the temporal expressions *Friday* and *Thursday* should be anchored with respect to the dateline associated with the article itself. We also know that *Friday* refers to the time of United's announcement, and *Thursday* refers to the time that the fare increase went into effect (ie. the Thursday immediately preceding the Friday). Finally, we can use this information to produce a timeline where United's announcement follows the fare increase and American's announcement follows both of those events. Temporal analysis of this kind is useful in pretty much any NLP application that deals with meaning, including question answering, summarization and dialog systems.

TEMPLATE-FILLING

Finally, many texts describe stereotypical situations that recur with some frequency in the domain of interest. The task of **template-filling** is to find documents that evoke such situations and then fill the slots in templates with appropriate material. 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 via some additional processing (times, amounts, entities from an ontology, etc.).

Our airline text is an example of this kind of stereotypical situation since airlines are often attempting to raise fares and then waiting 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 by which fares are being raised, *Thursday* as the effective date for the fare increase, and *American* as an airline that followed along. 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

The following sections will review current approaches to each of these problems in the context of generic news text. Sec. 22.5 then presents a short case-study illustrating their application to biology texts.

22.1 NAMED ENTITY RECOGNITION

NAMED ENTITY

The starting point for most information extraction applications is the detection and classification of the named entities in a text. By **named entity**, we simply mean anything that can be referred to with a proper name. This process of **named entity recognition** refers to the combined task of finding spans of text that constitute proper names and then classifying the entities being referred to according to their type.

Generic news-oriented NER systems focus on the detection of things like people, places, and organizations. Figures 22.1 and 22.2 provide lists of typical named entity types with examples of each. Specialized applications may be concerned with many other types of entities including commercial products, weapons, works of art, or as we'll see in Sec. 22.5, proteins, genes and other biological entities. What these applications all share is a concern with proper names, the characteristic ways that such names are signaled in a given language or genre, and a fixed set of categories of entities from a domain of interest.

By the way that names are signaled, we simply mean that names are denoted in a way that sets them apart from ordinary text. For example, if we're dealing

Type	Tag	Sample Categories
People	PER	Individuals, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-Political Entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicles	VEH	Planes, trains and automobiles

Figure 22.1 A list of generic named entity types with the kinds of entities they refer to.

Type	Example
People	*Turing* is often considered to be the father of modern computer science.
Organization	The *IPCC* said it is likely that future tropical cyclones will become more intense.
Location	The *Mt. Sanitas* loop hike begins at the base of *Sunshine Canyon*.
Geo-Political Entity	*Palo Alto* is looking at raising the fees for parking in the University Avenue district
Facility	Drivers were advised to consider either the *Tappan Zee Bridge* or the *Lincoln Tunnel*.
Vehicles	The updated *Mini Cooper* retains its charm and agility.

Figure 22.2 Named entity types with examples.

with standard English text, then two adjacent capitalized words in the middle of a text are likely to constitute a name. Further, if they are are preceded by a *Dr.* or followed by an *MD* then it is likely that we're dealing with a person. In contrast, if they are preceded by *arrived in* or followed by *NY* then we're probably dealing with a location. Note that these signals include facts about the proper names as well as their surrounding contexts.

The notion of a named entity is commonly extended to include things that aren't entities per se, but nevertheless have practical importance and do have characteristic signatures that signal their presence; examples include dates, times, named events and other kinds of **temporal expressions**, as well as measurements, counts, prices and other kinds of **numerical expressions**. We'll consider some of these later in Sec. 22.3.

TEMPORAL EXPRESSIONS

NUMERICAL EXPRESSIONS

Let's revisit the sample text introduced earlier with the named entities marked (with TIME and MONEY used to to mark the temporal and monetary 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 [$_{PERS}$ Tim Wagner] said.

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

Figure 22.3 Common categorical ambiguities associated with various proper names.

[$_{ORG}$ 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].

As shown, this text contains 13 mentions of named entities including 5 organizations, 4 locations, 2 times, 1 person, and 1 mention of money. The 5 organizational mentions correspond to 4 unique organizations, since *United* and *United Airlines* are distinct mentions that refer to the same entity.

22.1.1 Ambiguity in Named Entity Recognition

Named entity recognition systems face two types of ambiguity. The first arises from the fact the same name can refer to different entities of the same type. For example, *JFK* can refer to the former president or his son. This is basically a reference resolution problem and approaches to resolving this kind of ambiguity are discussed in Ch. 21.

The second source of ambiguity arises from the fact that identical named entity mentions can refer to entities of completely different types. For example, in addition to people *JFK* might refer to the airport in New York, or to any number of schools, bridges and streets around the United States. Some examples of this kind of cross-type confusion are given in Figures 22.3 and 22.4.

Notice that some of the ambiguities shown in Fig. 22.3 are completely coincidental; there is no relationship between the financial and organizational uses of the name *IRA*; they simply arose coincidentally as acronyms from different sources (*Individual Retirement Account* and *International Reading Association*). On the other hand, the organizational uses of *Washington* and *Downing St.* are examples of a LOCATION-FOR-ORGANIZATION **metonymy**, as discussed in Ch. 19.

22.1.2 NER as Sequence Labeling

The standard way to approach the problem of named entity recognition is as a word-by-word sequence labeling task, where the assigned tags capture both the

[$_{PERS}$ Washington] was born into slavery on a rural farm in southwestern Virginia.
[$_{ORG}$ Washington] went up 2 games to 1 in the four-game series.
Tony 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 in her.

Figure 22.4 Examples of type ambiguities in the use of the name *Washington*.

boundary and the type of any detected named entities. Viewed in this light, named entity recognition looks very much like the problem of syntactic base-phrase chunking. In fact, the dominant approach to NER is based on the same statistical sequence labeling techniques introduced in Ch. 5 for part of speech tagging and Ch. 13 for syntactic chunking.

In this sequence labeling approach to NER, classifiers are trained to label the tokens in a text with tags that indicate the presence of particular kinds of named entities. This approach makes use of the same style of IOB encoding employed for syntactic chunking. Recall that in this scheme an I is used to label tokens *inside* a chunk, B is used to mark the beginning of a chunk, and O labels tokens outside any chunk of interest. Consider the following sentence from our running example.

(22.1) [$_{ORG}$ American Airlines], a unit of [$_{ORG}$ AMR Corp.] immediately matched the move, spokesman [$_{PERS}$ Tim Wagner] said.

This bracketing notation provides us with the extent and the type of the named entities in this text. Fig. 22.5 shows a standard word-by-word IOB-style tagging that captures the same information. As with syntactic chunking, the tagset for such an encoding consists of 2 tags for each entity type being recognized, plus 1 for the O tag outside any entity, or $(2 \times N) + 1$ tags.

As with statistical chunking, the next step is to select a set of features to associate with each input example (ie. each of the tokens to be labeled in Fig. 22.5). These features should be plausible predictors of the class label and should be easily and reliably extractable from the source text. Recall that such features can be based not only on characteristics of the token to be classified, but also on the text in a surrounding window as well.

Fig. 22.6 gives a list of standard features employed in state-of-the-art named entity recognition systems. We've seen many of these features before in the context of part-of-speech tagging and syntactic base-phrase chunking. Several, however, are particularly important in the context of NER. The **shape** feature includes the usual upper case, lower case and capitalized forms, as well as more elaborate patterns designed to capture expressions that make use of numbers (*A9*), punctuation (*Yahoo!*) and atypical case alternations (*eBay*). It turns out that this feature by itself accounts for a considerable part of the success of NER systems for English news

Words	Label
American	B_{ORG}
Airlines	I_{ORG}
,	O
a	O
unit	O
of	O
AMR	B_{ORG}
Corp.	I_{ORG}
,	O
immediately	O
matched	O
the	O
move	O
,	O
spokesman	O
Tim	B_{PERS}
Wagner	I_{PERS}
said	O
.	O

Figure 22.5 IOB encoding for a sample sentence.

text. And as we'll see in Sec. 22.5, shape features are also particularly important in recognizing names of proteins and genes in biological texts. Fig. 22.7 describes some commonly employed shape feature values.

The **presence in a named entity list** feature can be very predictive. Extensive lists of names for all manner of things are available from both publicly available and commercial sources. Lists of place names, called **gazetteers**, contain millions of entries for all manner of locations along with detailed geographical, geologic and political information.[1] 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.

GAZETTEERS

This feature is typically implemented as a binary vector with a bit for each available kind of name list. Unfortunately, such lists can be difficult to create and maintain, and their usefulness varies considerably based on the named entity class.

[1] www.geonames.org

[2] www.census.gov

Feature	Explanation
Lexical items	The token to be labeled
Stemmed lexical items	Stemmed version of the target token
Shape	The orthographic pattern of the target word
Character affixes	Character level affixes of the target and surrounding words
Part of speech	Part of speech of the word
Syntactic chunk labels	Base phrase chunk label
Gazetteer or name list	Presence of the word in one or more named entity lists
Predictive token(s)	Presence of predictive words in surrounding text
Bag of words/Bag of N-grams	Words and/or N-grams occurring in the surrounding context.

Figure 22.6 Features commonly used in training named entity recognition systems.

Shape	Example
Lower	cummings
Capitalized	Washington
All caps	IRA
Mixed case	eBay
Capitalized initial with period	H.
Ends in digit	A9
Contains Hyphen	H-P

Figure 22.7 Selected shape features.

It appears that gazetteers can be quite effective, while extensive lists of persons and organizations are not nearly as beneficial (Mikheev et al., 1999).

Finally, features based on the presence of **predictive words and N-grams** in the context window can also be very informative. When they are present, preceding and following titles, honorifics, and other markers such as *Rev.*, *MD* and *Inc.* can accurately indicate the class of an entity. Unlike name lists and gazetteers, these lists are relatively short and stable over time and are therefore easy to develop and maintain.

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 via 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. 22.6 should therefore be thought of as only a starting point for any given application.

Once an adequate set of features have been developed, they are extracted from

Features				Label
American	NNP	B_{NP}	cap	B_{ORG}
Airlines	NNPS	I_{NP}	cap	I_{ORG}
,	PUNC	O	punc	O
a	DT	B_{NP}	lower	O
unit	NN	I_{NP}	lower	O
of	IN	B_{PP}	lower	O
AMR	NNP	B_{NP}	upper	B_{ORG}
Corp.	NNP	I_{NP}	cap_punc	I_{ORG}
,	PUNC	O	punc	O
immediately	RB	B_{ADVP}	lower	O
matched	VBD	B_{VP}	lower	O
the	DT	B_{NP}	lower	O
move	NN	I_{NP}	lower	O
,	PUNC	O	punc	O
spokesman	NN	B_{NP}	lower	O
Tim	NNP	I_{NP}	cap	B_{PER}
Wagner	NNP	I_{NP}	cap	I_{PER}
said	VBD	B_{VP}	lower	O
.	PUNC	O	punc	O

Figure 22.8 Simple word-by-word feature encoding for NER.

a representative training set and encoded in a form appropriate to train a machine learning-based sequence classifier. A standard way of encoding these features is to simply augment our earlier IOB scheme with more columns. Fig. 22.8 illustrates the result of adding part-of-speech tags, syntactic base-phrase chunk tags, and shape information to our earlier example.

Given such a training set, a sequential classifier can be trained to label new sentences. As with part-of-speech tagging and syntactic chunking, this problem can be cast either as Markov-style optimization using HMMs or MEMMs as described in Ch. 6, or as a multi-way classification task deployed as a sliding-window labeler as described in Ch. 13. Figure Fig. 22.9 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 2 preceding and following words, then the features available to the classifier are those shown in the boxed area. Fig. 22.10 summarizes the overall sequence labeling approach to creating a NER system.

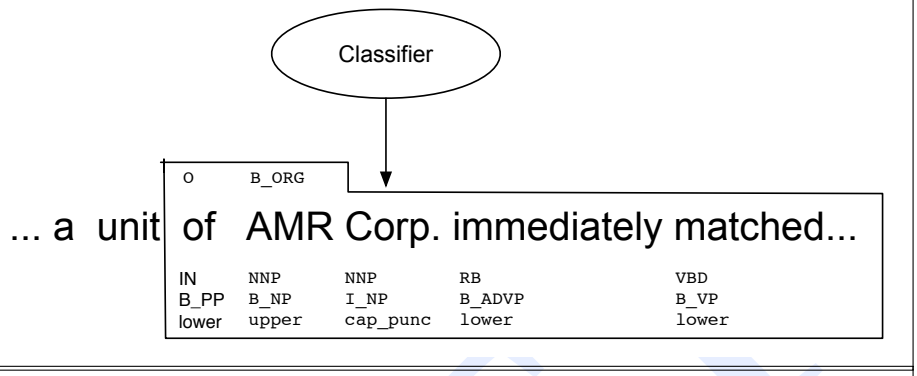

Figure 22.9 Named entity recognition as sequence labeling. The features available to the classifier during training and classification are those in the boxed area.

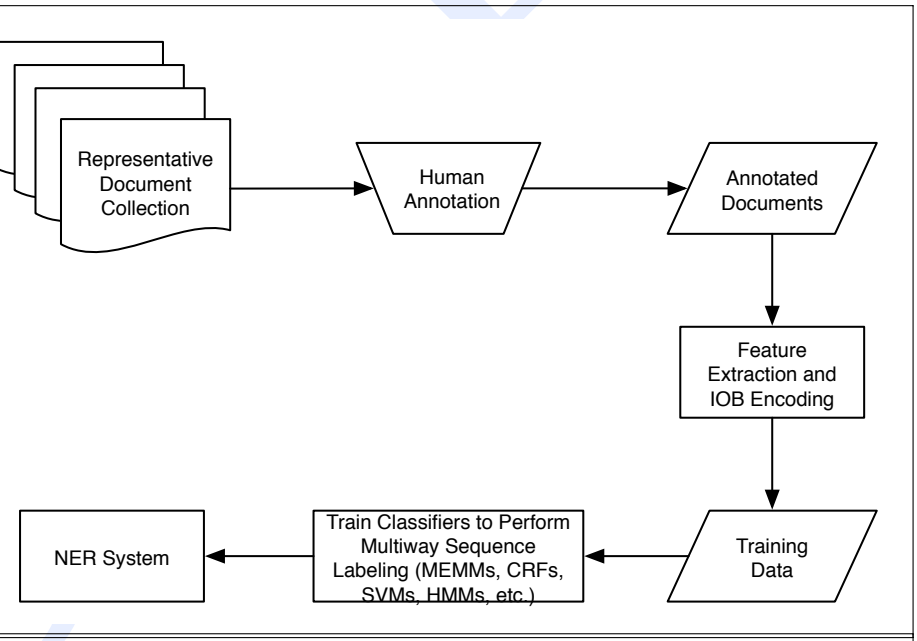

Figure 22.10 Basic steps in the statistical sequence labeling approach to creating a named entity recognition system.

22.1.3 Evaluating Named Entity Recognition

The familiar metrics of **recall**, **precision** and F_1 **measure** introduced in Ch. 13 are used to evaluate NER systems. Recall 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 produced; and the F_1 measure is the harmonic mean of the two, or $\frac{2PR}{P+R}$.

As with syntactic chunking, it is important to distinguish the metrics used to measure performance at the application level from those used during training. At the application level, recall and precision are measured with respect to the actual named entities detected. On the other hand, with an IOB encoding scheme the learning algorithms are attempting to optimize performance at the tag level. Performance at these two levels can be quite different; since the vast majority of tags in any given text are outside any entity, simply emitting an O tag for every token gives fairly high tag-level performance.

High performing systems at recent standardized evaluations have entity level F-measures around .92 for PERSONS and LOCATIONS, and around .84 for ORGANIZATIONS (Sang and De Meulder, 2003).

22.1.4 Practical NER Architectures

Commercial approaches to NER are often based on pragmatic combinations of lists, regular expression patterns and supervised machine learning. 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 sure-fire rules that are 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 regular expressions to tag unambiguous entity mentions;
2. Then search for sub-string matches of the previously detected names using probabilistic string matching metrics (as described in Ch. 19).
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 two-fold. 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).

Relations		Examples	Types
Affiliations			
	Personal	*married to, mother of*	PER → PER
	Organizational	*spokesman for, president of*	PER → ORG
	Artifactual	*owns, invented, produces*	(PER | ORG) → ART
Geospatial			
	Proximity	*near, on outskirts*	LOC → LOC
	Directional	*southeast of*	LOC → LOC
Part-Of			
	Organizational	*a unit of, parent of*	ORG → ORG
	Political	*annexed, acquired*	GPE → GPE

Figure 22.11 Typical semantic relations with examples and the named entity types they involve.

22.2 Relation Detection and Classification

Next on our list of tasks is the ability to discern the relationships that exist among the entities detected in a text. To see what this means, let's return to our sample airline text with all the entities marked.

> 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 [$_{PERS}$ 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].

This text stipulates a set of relations among the named entities mentioned within it. We know, 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 are all binary relations that can be seen as instances of more generic relations such as **part-of** or **employs** that occur with fairly high frequency in news-style texts. Fig. 22.11 shows a list of generic relations of the kind used in recent standardized evaluations.[3] More domain specific relations that might be extracted include 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 Ch. 17 to ground the meanings of the logical forms. That is, a relation

[3] http://www.nist.gov/speech/tests/ace/

Domain	$\mathscr{D} = \{a,b,c,d,e,f,g,h,i\}$
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 22.12 A model-based view of the relations and entities in our sample text.

consists of set of ordered tuples over elements of a domain. In most standard information extraction applications, the domain elements correspond either 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. Fig. 22.12 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.

22.2.1 Supervised Learning Approaches to Relation Analysis

Supervised machine learning approaches to relation detection and classification follow a scheme that should be familiar by now. Texts are annotated with relations chosen from a small fixed set by human analysts. These annotated texts are then used to train systems to reproduce similar annotations on unseen texts. Such annotations indicate the text spans of the two arguments, the roles played by each argument and the type of the relation involved.

The most straightforward approach breaks the problem down into two subtasks: detecting when a relation is present between two entities and then classifying any detected relations. In the first stage, a classifier is trained to make a binary decision as to whether or not a given pair of named entities participate in a relation or not. Positive examples are extracted directly from the annotated corpus, while negative examples are 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 22.13 Finding and classifying the relations among entities in a text.

In the second phase, a classifier is trained to label the relations that exist between candidate entity pairs. As discussed in Ch. 6, techniques such as decision trees, naïve Bayes or MaxEnt handle multiclass labeling directly. Binary approaches based on discovering separating hyperplanes such as SVMs solve multiclass problems by employing a one-versus-all training training paradigm. In this approach, a sets of classifiers are trained where each classifier is trained on one label as the positive class and all the other labels as the negative class. Final classification is performed by passing each instance to be labeled to all of the classifiers and then choosing the label from the classifier with the most confidence, or returning a rank ordering over the positively responding classifiers. Fig. 22.13 illustrates the basic approach for finding and classifying relations among the named entities within a discourse unit.

As with named entity recognition, the most important step in this process is to identify surface features that will be useful for relation classification (Zhou et al., 2005). The first source of information to consider are **features of the named entities** themselves:

- Named entity types of the two candidate arguments
- Concatenation of the two entity types
- Head words of the arguments
- Bag of words from each of the arguments

The next set of features are derived from **the words in the text** being examined. It's useful to think of these features as being extracted from three locations: the text between the two candidate arguments, a fixed window before the first argument, and a fixed window after the second argument. Given these locations the following word-based features have proven to be useful.

- The bag of words and bag of bigrams between the entities
- Stemmed versions of the same
- Words and stems immediately preceding and following the entities

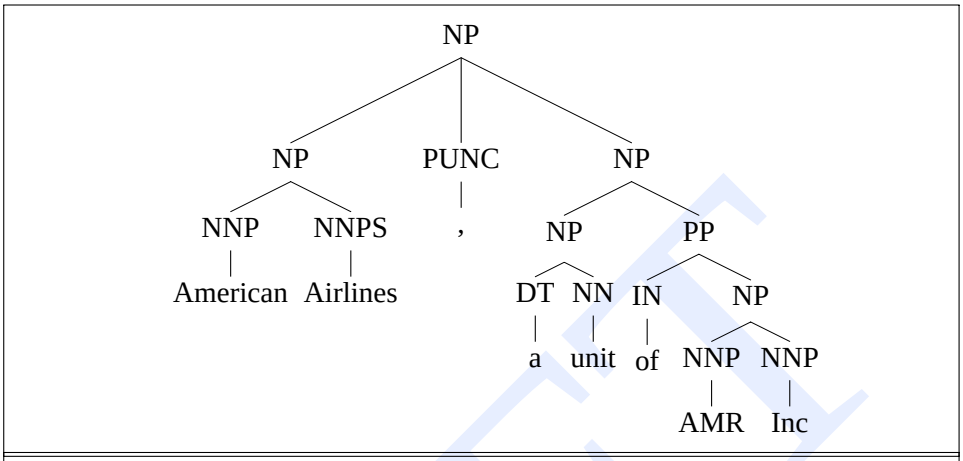

Figure 22.14 Here's a caption

- Distance in words between the arguments
- Number of entities between the arguments

Finally, **the syntactic structure** of a sentence can signal many of the relationships among any entities contained within it. The following features can be derived from various levels of syntactic analysis including base-phrase chunking, dependency parsing and full constituent parsing.

- Presence of particular constructions in a constituent structure
- Chunk base-phrase paths
- Bags of chunk heads
- Dependency-tree paths
- Constituent-tree paths
- Tree distance between the arguments

One method of exploiting parse trees is to create detectors that signal the presence of particular syntactic constructions and then associate binary features with those detectors. As an example of this, consider the sub-tree shown in Fig. 22.14 that dominates the named entities *American* and *AMR Inc.*. The *NP* construction that dominates these two entities is called an appositive construction and is often associated with both **part-of** and **a-kind-of** relations in English. A binary feature indicating the presence of this construction can be useful in detecting these relations.

This method of feature extraction relies on a certain amount of a priori linguistic analysis to identify those syntactic constructions that may be useful predictors of certain classes. An alternative method is to automatically encode certain

Entity-based features	
Entity$_1$ type	ORG
Entity$_1$ head	*airlines*
Entity$_2$ type	PERS
Entity$_2$ head	*Wagner*
Concatenated types	ORGPERS
Word-based features	
Between-entity bag of words	{ *a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman* }
Word(s) before Entity$_1$	NONE
Word(s) after Entity$_2$	*said*
Syntactic features	
Constituent path	$NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
Base syntactic chunk path	$NP \rightarrow NP \rightarrow PP \rightarrow NP \rightarrow VP \rightarrow NP \rightarrow NP$
Typed-dependency path	$Airlines \leftarrow_{subj} matched \leftarrow_{comp} said \rightarrow_{subj} Wagner$

Figure 22.15 Sample of features extracted while classifying the <American Airlines, Tim Wagner> tuple.

aspects of tree structures as feature values and allow the machine learning algorithms to determine which values are informative for which classes. One simple and effective way to do this this involves the use of **syntactic paths** through trees. Consider again the tree discussed earlier that dominates *American Airlines* and *AMR Inc.*. The syntactic relationship between these arguments can be characterized by the path traversed through the tree in getting from one to the other:

$NP \uparrow NP \downarrow NP \downarrow PP \downarrow NP$

Similar path features defined over syntactic dependency trees as well as flat base-phrase chunk structures have been shown to be useful for relation detection and classification (Ray and Craven, 2001; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005). Recall that syntactic path features featured prominently in Ch. 20 in the context of semantic role labeling.

Fig. 22.15 illustrates some of the features that would be extracted while trying to classify the relationship between *American Airlines* and *Tim Wagner* from our example text.

22.2.2 Lightly Supervised Approaches to Relation Analysis

The supervised machine learning approach just described assumes that we have ready access to a large collection of previously annotated material with which to train classifiers. Unfortunately, this assumption is impractical in many real

world settings. A simple approach to extracting relational information without large amounts of annotated material is to use regular expression patterns to match text segments that are likely to contain expressions of the relations in which we're interested.

Consider the problem of building a table containing all the hub cities that various airlines make use of. Assuming we have access a search engine that permits some form of phrasal search with wildcards, we might try something like the following as a query:

```
/ * has a hub at * /
```

Given access to a reasonable amount of material of the right kind, such a search will yield a fair number of correct answers. A recent Google search using this pattern yields the following relevant sentences among the return set.

(22.2) Milwaukee-based Midwest has a hub at KCI.

(22.3) Delta has a hub at LaGuardia.

(22.4) Bulgaria Air has a hub at Sofia Airport, as does Hemus Air.

(22.5) American Airlines has a hub at the San Juan airport.

Of course, patterns such as this can fail in the two ways we discussed all the way back in Ch. 2: finding some things they shouldn't, and by failing to find things they should. As an example of the first kind of error, consider the following sentences that were also included the earlier return set.

(22.6) airline j has a hub at airport k

(22.7) The catheter has a hub at the proximal end

(22.8) A star topology often has a hub at its center.

We can address these errors by specializing our proposed pattern to eliminate these responses. In this case, replacing the unrestricted wildcard operator with a named entity class restriction would rule these examples out:

```
/[ORG] has a hub at [LOC]/
```

The second problem is that we can't know if we've found all the hubs for all airlines since we've limited ourselves to this one rather specific pattern. Consider the following close calls missed by our first pattern.

(22.9) No frills rival easyJet, which has established a hub at Liverpool...

(22.10) Ryanair also has a continental hub at Charleroi airport (Belgium).

These examples are missed because they contain minor variations that cause the original pattern to fail. There are two ways to address this problem. The first is to generalize our pattern to capture expressions like these that contain the information we're seeking. This can be accomplished by relaxing the pattern to allow matches that skip parts of the candidate text. Of course, this approach is likely introduce

more of the false positives that we tried to eliminate by making our pattern more specific in the first place.

The second, more promising solution, is to expand our set of specific high precision patterns. Given a large and diverse document collection, an expanded set of patterns should be able to capture more of the information we're looking for. One way to acquire these additional patterns is to simply have human analysts familiar with the domain come up with more patterns and hope to get better coverage. A more interesting automatic alternative is to induce new patterns by **bootstrapping** from the initial search results from a small set of **seed patterns**.

BOOTSTRAPPING

SEED PATTERNS

To see how this works, let's assume that we've discovered that Ryanair has a hub at Charleroi. We can use this fact to discover new patterns by finding other mentions of this relation in our corpus. The simplest way to do this is to search for the terms *Ryanair*, *Charleroi* and *hub* in some proximity. The following are among the results from a recent search in GoogleNews.

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

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

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

From these results, patterns such as the following can be extracted that look for relevant named entities of various types in the right places.

```
/ [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.

Fig. 22.16 illustrates the overall bootstrapping approach. This figure shows that the dual nature of patterns and seeds permits the process to start with either a small set of **seed tuples** or a set of **seed patterns**. This style of bootstrapping and pattern-based relation extraction is closely related to the techniques discussed in Ch. 20 for extracting hyponym and meronym-based lexical relations.

There are, of course, a fair number of technical details to be worked out to actually implement such an approach. Among the key problems are the following:

- Representing the search patterns
- Assessing the accuracy and coverage of discovered patterns
- And assessing the reliability of the discovered tuples

Patterns are typically represented in a way that captures the following four factors.

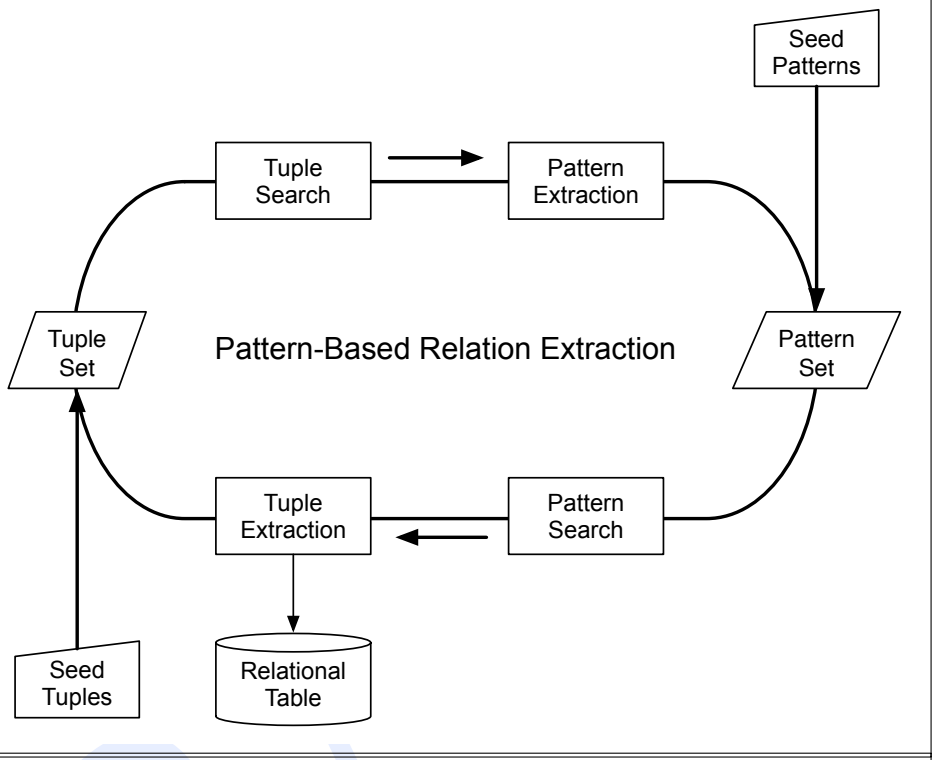

Figure 22.16 Pattern and bootstrapping-based relation extraction.

- Context prior to the first entity mention
- Context between the entity mentions
- Context following the second mention
- The order of the arguments in the pattern

Contexts are either captured as regular expression patterns or as vectors of features similar to those described earlier for machine learning-based approaches. In either case, they can be defined over character strings, word-level tokens, or syntactic and semantic structures. In general, regular expression approaches tend to be very specific, yielding high precision results; feature-based approaches, on the other hand, are more capable of ignoring potentially inconsequential elements of contexts.

Our next problem is how to assess the reliability of newly discovered patterns and tuples. Recall, that we don't, in general, have access to annotated materials giving us the right answers. We therefore have to rely on the accuracy of the initial seed sets of patterns and/or tuples for gold-standard evaluation, and we have to insure that we don't permit any significant **semantic drift** to occur as we're

SEMANTIC DRIFT

learning new patterns and tuples.

There are two factors that need to be balanced in assessing a proposed new pattern: 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 $\mathscr{D}$, a current set of tuples T, and a proposed pattern p, there are three factors we need to track:

- *hits*: the set of tuples in T that p matches while looking in $\mathscr{D}$;
- *misses*: The set of tuples in T that p misses while looking at $\mathscr{D}$;
- $finds$: The total set of tuples that p finds in $\mathscr{D}$.

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

$$Conf_{RlogF}(p) = \frac{hits_p}{hits_p + misses_p} \times log(finds_p)$$

It's useful to be able to treat this metric as a probability, so we'll need to normalize it somehow. A simple way to do this is to track the range of confidences in a development set and divide by some previously observed maximum confidence (Agichtein and Gravano, 2000).

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 $\mathscr{D}$ (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 assume that for a proposed tuple to be false, *all* of its supporting patterns must have been in error, and that the sources of their individual failures are independent. If we loosely treat our confidence measures as probabilities, then the probability of any individual pattern p failing is simply $1 - Conf(p)$; the probability of all the supporting patterns for a tuple being wrong is simply the product of their individual failure probabilities, leaving us with the following equation for our confidence in a new tuple.

NOISY-OR

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

Of course, the independence assumptions underlying the noisy-or model are very strong indeed. If the failure mode of the patterns are not independent then the method overestimates the confidence for the tuple. This overestimate is typically compensated for by setting the threshold for the acceptance of a tuple to be fairly high.

Given these measures, we can dynamically assess our confidence in both new tuples and patterns as the bootstrapping process iterates. Setting conservative thresholds for the acceptance of new patterns and tuples should help prevent the system from drifting from the targeted relation.

Although there have been no standardized evaluations for this style of relation extraction on publicly available sources, the technique has gained wide acceptance as a practical way to quickly populate relational tables from open source materials (most commonly from the Web) (Etzioni et al., 2005).

22.2.3 Evaluating Relation Analysis Systems

There are two separate methods for evaluating relation detection systems. In the first approach, the focus is on how well systems can find and classify all the relation mentions in a given text. In this approach, labeled and unlabeled recall, precision and F-measures are used to evaluate systems against a test collection with human annotated gold-standard relations. Labeled precision and recall requires the system to classify the relation correctly, while unlabeled methods simply measure a system's ability to detect entities that are related.

The second approach focuses on the tuples to be extracted from a body of text, rather than on the relation mentions. In this method, systems need not detect every mention of a relation to be scored correctly. Instead, the final 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.

This method has typically used to evaluate unsupervised methods of the kind discussed in the last section. In these evaluations human analysts simply examine the set of tuples produced by the system. Precision is simply the fraction of correct tuples out of all the tuples produced as judged by the human experts.

Recall remains a problem in this approach. It is obviously too costly to search by hand for all the relations that could have been extracted from a potentially large collection such as the Web. One solution is to compute recall at various levels of precision as described in Ch. 25 (Etzioni et al., 2005). Of course, this isn't true recall, since we're measuring against the number of correct tuples discovered rather than the number of tuples that are theoretically extractable from the text.

Another possibility is to evaluate recall on problems where large resources containing comprehensive lists of correct answers are available. Examples of include gazetteers for facts about locations, the Internet Movie Database (IMDB) for facts about movies or Amazon for facts about books. The problem with this approach is that it measures recall against a database that may be far more comprehensive than the text collections used by relation extraction system.

22.3 TEMPORAL AND EVENT PROCESSING

Our focus thus far has been on extracting information about entities and their relations to one another. However, in most texts entities are introduced in the course of describing the events in which they are take part. Finding and analyzing the events in a text is therefore critical to extracting a more complete picture of the contents of a text. Among the most critical features of events that we can determine are when they occur in time. Temporal information can critical be critical in applications such as question answering and summarization.

In question answering whether or not a system detects a correct answer may depend on temporal relations extracted from both the question and the potential answer text. As an example of this consider the following sample question and potential answer text.

When did airlines as a group last raise fares?

> Last week, Delta boosted thousands of fares by $10 per round trip, and most big network rivals immediately matched the increase. (Dateline 7/2/2007).

This snippet does provide an answer to the question, but extracting it requires temporal reasoning to anchor the phrase *last week*, to link that time to the *boosting* event, and finally to link the time of the *matching* event to that.

The following sections introduce approaches to recognizing temporal expressions, figuring out the times that those expressions refer to, detecting events and finally associating times with those events.

22.3.1 Temporal Expression Recognition

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 via 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.) Fig. 22.17 provides some sample temporal expressions in each of these categories.

LEXICAL TRIGGERS

Syntactically, temporal expressions are syntactic constructions that have temporal **lexical triggers** as their heads. In the annotation scheme in widest use, lexical triggers can be nouns, proper nouns, adjectives, and adverbs; full temporal expression consist of their phrasal projections: noun phrases, adjective phrases and adverbial phrases. Figure 22.18 provides examples of lexical triggers from 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 22.17 Examples of absolute, relation and durational temporal expressions.

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 22.18 Examples of temporal lexical triggers.

The annotation scheme in widest use derives from the TIDES standard for the annotation of temporal expressions (Ferro et al., 2005). The approach presented here is based on the TimeML effort (Pustejovsky et al., 2005). TimeML provides an XML tag, TIMEX3, along with various attributes to that tag, for annotating temporal expressions. The following example illustrates the basic use of this scheme (ignoring the additional attributes, which we'll discuss as needed later in Sec. 22.3.2.)

> 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</TIMEX2>.

TEMPORAL EXPRESSION RECOGNITION

The **temporal expression recognition** task consists of finding the start and end of all the text spans that correspond to such temporal expressions. Although there are myriad ways to compose time expressions in English, the set of temporal trigger terms is, for all practical purposes, static and the set of constructions used to generate temporal phrases is quite conventionalized. These facts suggest that any of the major approaches to finding and classifying text spans that we've already studied should be successful. The following three approaches have all been successfully employed in recent evaluations:

- Rule-based systems based on partial parsing or chunking
- Statistical sequence classifiers based on standard token-by-token IOB encoding
- Constituent-based classification as used in semantic role labeling

Rule-based approaches to temporal expression recognition use cascades of automata to recognize patterns at increasing levels of complexity. Since temporal expressions are limited to a fixed set of standard syntactic categories, most of these systems make use of pattern-based methods for recognizing syntactic chunks. That is, tokens are first part-of-speech tagged and then larger and larger chunks are recognized using the results from previous stages. The only difference from the usual partial parsing approaches is the fact that temporal expressions must contain temporal lexical triggers. Patterns must, therefore, contain either specific trigger words (eg. *February*), or patterns representing classes (eg. *MONTH*). Fig. 22.19 illustrates this approach with a small representative fragment from a rule-based system written in Perl.

Sequence labeling approaches follow exactly the same scheme introduced in Ch. 13 for syntactic chunking. The three tags I, O and B are used to mark tokens that are either inside, begin or outside the extent of a temporal expression, as delimited by TIMEX3 tags. Example 22.3.1 would be labeled as follows in this scheme.

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

As expected, features are extracted from the context surrounding any tokens to be tagged and a statistical sequence labeler is trained using those features. As with syntactic chunking and named entity recognition, any of the usual statistical sequence methods can be applied. Fig. 22.20 lists the standard features used in the machine learning-based approach to temporal tagging.

Constituent-based methods combine aspects of both chunking and token-by-token labeling. In this approach, a complete constituent parse is produced by automatic means. The nodes in the resulting tree are then classified, one by one, as to whether they contain a temporal expression or not. This task is accomplished by training a binary classifier with annotated training data using many of the same features employed in IOB-style training. This approach separates the classification problem from the segmentation problem by assigning the segmentation problem to the syntactic parser. The motivation for this choice was mentioned earlier; in currently available training materials temporal expressions are limited to syntactic constituents in one of a fixed set of syntactic categories. Based on this, it makes sense to allow a syntactic parser to solve the segmentation part of the problem.

In standard evaluations, temporal expression recognizers are evaluated using the usual recall, precision and F-measures. In recent evaluations, both rule-based and statistical systems achieve about the same level of performance, with the best systems reaching an F-measure of around .87 on a strict exact match criteria. On a looser criteria based on overlap with gold standard temporal expressions, the best

```
# 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 22.19 Fragment of Perl code from MITRE's TempEx temporal tagging system.

Feature	Explanation
Token	The target token to be labeled
Tokens in window	Bag of tokens in the window around 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 window
Lexical triggers	Presence in a list of temporal terms.

Figure 22.20 Typical features used to train IOB style temporal expression taggers.

systems reach an F-measure of .94.

The major difficulties for all of these approaches are achieving reasonable coverage, correctly identifying the extent of temporal expressions and dealing with expressions that trigger false positives. The problem of false positives arises from the use of temporal trigger words as parts of proper names. For example, all of the following examples are likely to cause false positives for either rule-based or statistical taggers.

(22.14) *1984* tells the story of Winston Smith and his degradation by the totalitarian state in which he lives.

(22.15) Edge is set to join Bono onstage to perform U2's classic *Sunday* Bloody *Sunday*.

(22.16) Black *September* tried to detonate three car bombs in New York City in March 1973.

<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 22.21 TimeML markup include normalized values for temporal expressions.

22.3.2 Temporal Normalization

TEMPORAL NORMALIZATION

The task of recognizing temporal expressions is typically followed by the task of normalization. **Temporal normalization** refers to the process of mapping a temporal expression to either a specific point in time, or to a duration. Points in time correspond either to calendar dates or to times of day (or both). Durations primarily consist of lengths of time, but may also include information concerning the start and end points of a duration when that information is available.

Normalized representations of temporal expressions are captured using the VALUE attribute are represented using the ISO 8601 standard for encoding temporal values(ISO8601, 2004). To illustrate some aspects of this scheme, let's return to our earlier example, reproduced in Fig. 22.21 with the value attributes added in.

FULLY QUALIFIED DATE

The dateline, or document date, for this text was *July 2, 2007)*. The ISO representation for this kind of **fully qualified date** 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. Let's consider each of these temporal expressions in turn.

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 using 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 using 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 both the ISO 8601 standard and the various temporal

Unit	Pattern	Sample Value
Fully Specified Dates	YYYY-MM-DD	1991-09-28
Weeks	YYYY-nnW	2007-27W
Weekends	PnWE	P1WE
24 hour clock times	HH:MM:SS	11:13:45
Combined Dates and Times	YYYY-MM-DDTHH:MM:SS	1991-09-28T11:00:00
Financial quarters	Qn	1999-3Q

Figure 22.22 Sample ISO patterns for representing various times and durations.

annotation standards, far too much to cover here. Fig. 22.22 describes some of the basic ways that other times and durations are represented. Interested readers should consult (ISO8601, 2004; Ferro et al., 2005; Pustejovsky et al., 2005) for more details.

Most current approaches to temporal normalization employ rule-based methods that associate semantic analysis procedures with patterns matching particular temporal expressions. This is a domain-specific instantiation of the compositional rule-to-rule approach introduced in Ch. 18. In this approach, the meaning of a constituent is computed from the meaning of its parts, and the method used to perform this computation is specific to the constituent being created. The only difference here is that the semantic composition rules involve simple temporal arithmetic rather than λ-calculus attachments.

To normalize temporal expressions, we'll need rules for four kinds of expressions.

- Fully qualified temporal expressions
- Absolute temporal expressions
- Relative temporal expressions
- Durations

Fully qualified temporal 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 the fully-qualified temporal expression used in expressions like *April 24, 1916*.

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

In this rule, the non-terminals *Month*, *Date*, and *Year* represent constituents have already been recognized and assigned semantic values, accessed via 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. Most temporal expressions in news articles are incomplete and are only implicitly anchored, often with respect to the dateline of the article, which we'll refer to as the document's **temporal anchor**. The values of relatively simple 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, while the attachments for *tomorrow* and *yesterday* add a day and subtract a day from the anchor, respectively. Of course, given the circular 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.

TEMPORAL ANCHOR

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.

(22.17) 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 (ie. the coming weekend). The information that signals this 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*. To illustrate this, consider the expression *last week* from our example. From the document date, we can determine that the ISO week for the article is week 27, so *last week* is simply 1 minus the current week.

Again, even simple constructions such as this can be ambiguous in English. The resolution of expressions involving *next* and *last* must take into account the distance from the anchoring date to the nearest unit in question. For example, a phrase such as *next Friday* can refer to either the immediately next Friday, or to the Friday following that. The determining factor has to do with the proximity to the reference time. The closer the document date is to a Friday, the more likely it is that the phrase *next Friday* will skip the nearest one. Such ambiguities are handled by encoding language and domain specific heuristics into the temporal attachments.

The need to associate highly idiosyncratic temporal procedures with particular temporal constructions accounts for the widespread use of of rule-based methods in temporal expression recognition. Even when high performance statistical methods are used for temporal recognition, rule-based patterns are still required for normalization. Although the construction of these patterns can be tedious and filled with exceptions, it appears that sets of patterns that provide good coverage in newswire domains can be created fairly quickly (Ahn et al., 2005).

Finally, many temporal expressions are anchored to events mentioned in a

text and not directly to other temporal expressions. Consider the following example.

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

To determine when JetBlue issued its customer bill of rights we need to determine the time of *the storm* event, and then that time needs to be modified by the temporal expression *one week after*. We'll return to this issue when we take up event detection in the next section.

22.3.3 Event Detection and Analysis

EVENT DETECTION AND CLASSIFICATION

The task of **event detection and classification** is to identify mentions of events in texts and then assign those events to a variety of classes. 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 Example 22.3.1 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*.

Both rule-based and statistical machine learning approaches have been applied to the problem of event detection. Both approaches make use of surface information such as parts of speech information, presence of particular lexical items, and verb tense information. Fig. 22.23 illustrates the key features used in current event detection and classification systems.

Having detected both the events and the temporal expressions in a text, 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

Feature	Explanation
Character affixes	Character-level prefixes and suffixes of target word
Nominalization suffix	Character level suffixes for nominalizations (eg. *-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 22.23 Features commonly used in both rule-based and statistical approaches to event detection.

and summarization. This ambitious task is 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 would be to determine 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 Sec. 22.2.

Current approaches to this problem attempt to identify a subset of Allen's 13 temporal relations discussed earlier in Ch. 17, and shown here in Fig. 22.24. Recent evaluation efforts have focused on detecting the *before*, *after* and *during* relations among the temporal expressions, document date and event mentions in a text (Verhagen et al., 2007). Most of the top-performing systems employ statistical classifiers, of the kind discussed earlier in Sec. 22.2, trained on the TimeBank corpus (Pustejovsky et al., 2003b).

22.3.4 TimeBank

As we've seen with other tasks, it's tremendously useful to have access to text annotated with the types and relations in which we're interested. Such resources facilitate both corpus-based linguistic research as well as the training of systems to perform automatic tagging. The **TimeBank** corpus consists of text annotated with much of the information we've been discussing throughout this section (Pustejovsky et al., 2003b). The current release (TimeBank 1.2) of the corpus consists of 183 news articles selected from a variety of sources, including the Penn TreeBank and PropBank collections.

TIMEBANK

Each article in the TimeBank corpus has had the temporal expressions and

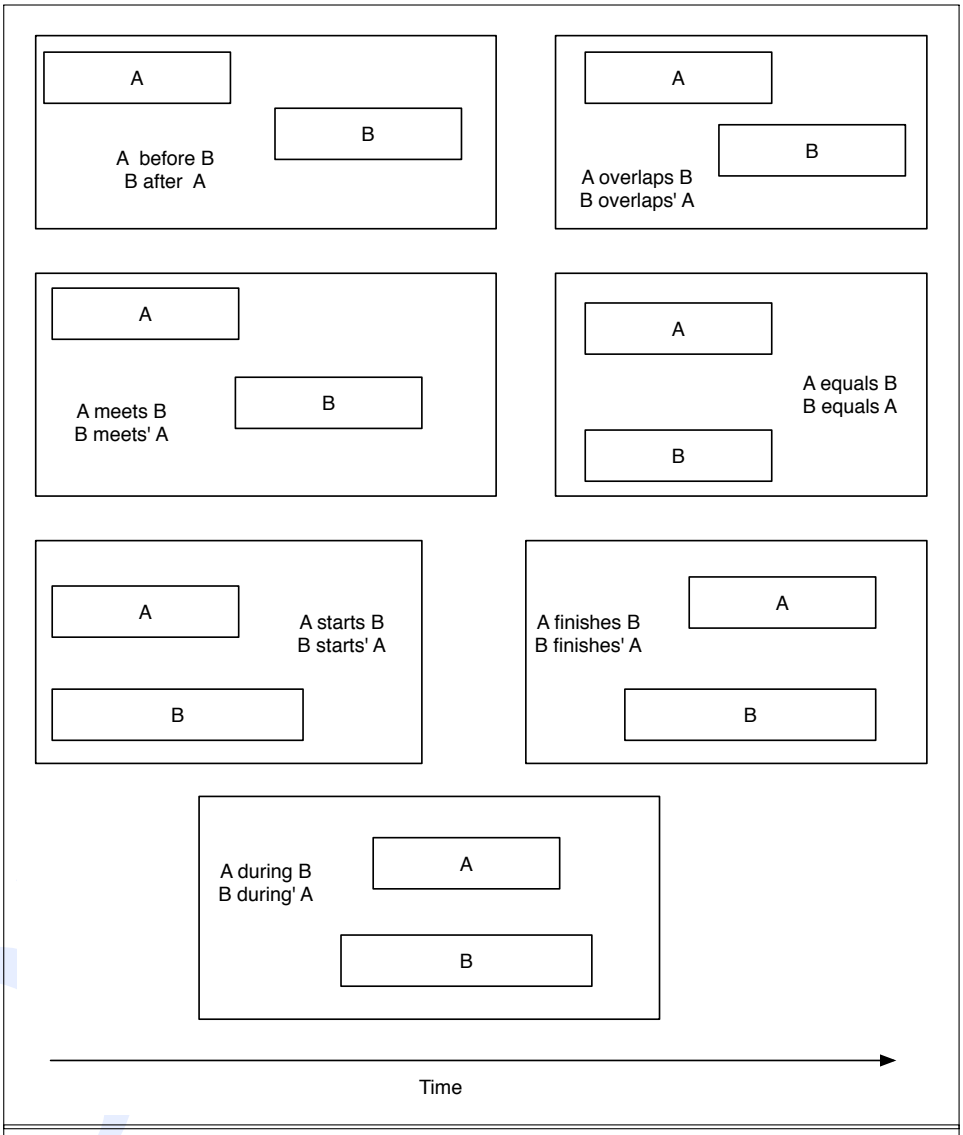

Figure 22.24 Allen's 13 possible temporal relations.

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. 22.25 selected from one of the TimeBank documents.

<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 22.25 Example from the TimeBank corpus.

(22.19) Delta Air Lines 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 4 links that capture the temporal relations between the events and times in the text. 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$_{e1}$

The set of 13 temporal relations used in TimeBank are based on Allen's (Allen, 1984) relations introduced earlier in Fig. 22.24.

22.4 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 can be characterized as **scripts**, in that they consist of prototypical sequences of sub-events, participants, roles and props (Schank and Abelson, 1977). The use of explicit representations of such scripts in language processing can assist in many of the IE tasks we've been discussing. In particular, 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. Sec. 22.6 discusses some of the more advanced, research oriented applications of scripts.

TEMPLATES

In their simplest form, such scripts can be represented as **templates** consisting of fixed sets of **slots** which 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 via some additional processing (times, amounts, entities from an ontology, etc.)

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

Note that as is often the case, the slot-fillers in this example all correspond to detectable named entities of various kinds (organizations, amounts and times). This suggests that template-filling applications should rely on tags provided by named entity recognition, temporal expression and co-reference algorithms to identify candidate slot-fillers.

The next section describes a straightforward approach to filling slots using sequence labeling techniques. Sec. 22.4.2 then describes a system designed to address a considerably more complex template-filling task, based on the use of cascades of finite-state transducers.

22.4.1 Statistical Approaches to Template-Filling

A surprisingly effective approach to template-filling simply casts it as a statistical sequence labeling problem. In this approach, systems are trained to label sequences of tokens as potential fillers for particular slots. There are two basic ways to instantiate this approach: the first is to train separate sequence classifiers for each slot to be filled and then send the entire text through each labeler, the other is to train one large classifier (usually an HMM) that assigns labels for each of the slots to be recognized. We'll focus on the former approach here; we'll take up the single large classifier approach in Ch. 23.

Under the one classifier per slot approach, slots are filled with the text segments identified by each slot's corresponding classifier. As with the other IE tasks described earlier in this chapter, all manner of statistical sequence classifiers have been applied to this problem, all using the usual set of features: tokens, shapes of tokens, part-of-speech tags, syntactic chunk tags, and named entity tags.

There is the possibility in this approach that multiple non-identical text segments will be labeled with the same slot label. This situation can arise in two ways: from competing segments that refer to the same entity using different referring expressions, or from competing segments that represent truly distinct hypotheses. In our sample text, we might expect the segments *United*, *United Airlines* to be labeled as the LEAD AIRLINE. These are not incompatible choices and the reference resolution techniques introduced in Ch. 21 can provide a path to a solution.

Truly competing hypotheses arise when a text contains multiple entities of the expected type for a given slot. In our example, *United Airlines* and *American Airlines* are both airlines and it is possible for both to be tagged as LEAD AIRLINE based on their similarity to exemplars in the training data. In general, most systems simply choose the hypothesis with the highest confidence. Of course, the implementation of this confidence heuristic is dependent on the style of sequence classifier being employed. Markov-based approaches simply select the segment with the highest probability labeling (Freitag and McCallum, 1999).

A variety of annotated collections have been used to evaluate this style of approach to template-filling, including sets of job announcements, conference calls for papers, restaurant guides and biological texts. A frequently employed collection is the CMU Seminar Announcement Corpus[4], a collection of 485 seminar announcements retrieved from the Web with slots annotated for the SPEAKER, LOCATION, START TIME and END TIME. State-of-the-art F-measures on this dataset range from around .98 for the start and end time slots, to as high as .77 for the speaker slot (Roth and tau Yih, 2001; Peshkin and Pfefer, 2003).

As impressive as these results are, they are due as much to the constrained nature of the task as to the techniques they have been employed. Three strong task constraints have contributed to this success. First, in most evaluations all the documents in the collection are all relevant and homogeneous, that is they are known to contain the slots of interest. Second, the documents are all relatively small, providing little room for distractor segments that might incorrectly fill slots. And finally, the target output consists solely of a small set of slots which are to be filled with snippets from the text itself.

22.4.2 Finite State Template-Filling Systems

The tasks introduced in the *Message Understanding Conferences* (MUC) (Sundheim, 1993), a series of U.S. Government-organized information extraction evaluations, represent a considerably more complex template-filling problem. Consider the following sentences selected from the MUC-5 materials from Grishman and

[4] http://www.isi.edu/info-agents/RISE/

TIE-UP-1:	
RELATIONSHIP:	TIE-UP
ENTITIES:	"Bridgestone Sports Co."
	"a local concern"
	"a Japanese trading house"
JOINTVENTURECOMPANY	"Bridgestone Sports Taiwan Co."
ACTIVITY	ACTIVITY-1
AMOUNT	NT$20000000
ACTIVITY-1:	
COMPANY	"Bridgestone Sports Taiwan Co."
PRODUCT	"iron and "metal wood" clubs"
STARTDATE	DURING: January 1990

Figure 22.26 The templates produced by the FASTUS (Hobbs et al., 1997) information extraction engine given the input text on page 36.

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 evaluation task required systems to produce hierarchically linked templates describing the participants in the joint venture, the resulting company, and its intended activity, ownership and capitalization. Fig. 22.26 shows the resulting 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 to be filled.

The FASTUS system produces the template given above, based on a cascade of transducers in which each level of linguistic processing extracts some information from the text, which is passed on to the next higher level, as shown in Figure 22.27

Most systems base most of these levels on finite-automata, although in practice most complete systems are not technically finite-state, either because the individual automata are augmented with feature registers (as in FASTUS), or because they are used only as preprocessing steps for full parsers (e.g., Gaizauskas et al., 1995; Weischedel, 1995) , or are combined with other components based on statistical methods (Fisher et al., 1995).

Let's sketch the FASTUS implementation of each of these levels, following Hobbs et al. (1997) and Appelt et al. (1995). After tokenization, the second level

No.	Step	Description
1	**Tokens:**	Transfer an input stream of characters into a token sequence.
2	**Complex Words:**	Recognize multi-word phrases, numbers, and proper names.
3	**Basic phrases:**	Segment sentences into noun groups, verb groups, and particles.
4	**Complex phrases:**	Identify complex noun groups and complex verb groups.
5	**Semantic Patterns:**	Identify semantic entities and events and insert into templates.
6	**Merging:**	Merge references to the same entity or event from different parts of the text.

Figure 22.27 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.

recognizes multiwords like *set up*, and *joint venture*, and names like *Bridgestone Sports Co.*. The named entity recognizer is a transducer, composed of a large set of specific mappings designed to handle the usual set of named entities.

The following are typical rules for modeling names of performing organizations like *San Francisco Symphony Orchestra* and *Canadian Opera Company*. While the rules are written using a context-free syntax, there is no recursion and therefore they can be automatically compiled into finite-state transducers:

Performer-Org → (pre-location) Performer-Noun+ Perf-Org-Suffix
pre-location → locname | nationality
locname → city | region
Perf-Org-Suffix → orchestra, company
Performer-Noun → symphony, opera
nationality → Canadian, American, Mexican
city → San Francisco, London

The second stage also might transduce sequences like *forty two* into the appropriate numeric value (recall the discussion of this problem in Ch. 8).

The third FASTUS stage implements chunking and produces a sequence of basic syntactic chunks, such as noun groups, verb groups, and so on, using finite-state rules of the sort discussed in Ch. 13.

The output of the FASTUS basic phrase identifier is shown in Figure 22.28; note the use of some domain-specific basic phrases like *Company* and *Location*.

Recall that Ch. 13 described how these basic phrases can be combined into

Company	Bridgestone Sports Co.
Verb Group	said
Noun Group	Friday
Noun Group	it
Verb Group	had set up
Noun Group	a joint venture
Preposition	in
Location	Taiwan
Preposition	with
Noun Group	a local concern
Conjunction	and
Noun Group	a Japanese trading house
Verb Group	to produce
Noun Group	golf clubs
Verb Group	to be shipped
Preposition	to
Location	Japan

Figure 22.28 The output of Stage 2 of the FASTUS basic-phrase extractor, which uses finite-state rules of the sort described by Appelt and Israel (1997) and shown on page **??**.

(1)	RELATIONSHIP:	TIE-UP
	ENTITIES:	"Bridgestone Sports Co."
		"a local concern"
		"a Japanese trading house"
(2)	ACTIVITY:	PRODUCTION
	PRODUCT	"golf clubs"
(3)	RELATIONSHIP:	TIE-UP
	JOINTVENTURECOMPANY:	"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 22.29 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. 22.26 on page 36.

more complex noun groups and verb groups. This is accomplished in Stage 4 of FASTUS, by dealing with conjunction and with the attachment of measure phrases as in the following:

20,000 iron and “metal wood” clubs a month,

and preposition phrases:

production of 20,000 iron and “metal wood” clubs a month,

The output of Stage 4 is a list of complex noun groups and verb groups. Stage 5 takes this list, ignoring all input that has not been chunked into a complex group, recognizes entities and events in the complex groups, and inserts the recognized objects into the appropriate slots in templates. The recognition of entities and events is done by hand-coded finite-state automata whose transitions are based on particular complex-phrase types annotated by particular head words or particular features like *company*, *currency*, or *date*.

As an example, the first sentence of the news story above realizes the semantic patterns based on the following two regular expressions (where NG indicates Noun-Group and VG Verb-Group):

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

The second sentence realizes the second pattern above as well as the following two patterns:

- NG(Company) VG-Passive(Capitalized) at NG(Currency)
- NG(Company) VG(Start) NG(Activity) in/on NG(Date)

The result of processing these two sentences is the set of five draft templates shown in Fig. 22.29. These five templates must then be merged into the single hierarchical structure shown in Fig. 22.26. The merging algorithm decides whether two activity or relationship structures are sufficiently consistent that they might be describing the same events, and merges them if so. The merging algorithm must also perform reference resolution as described in Ch. 21.

22.5 Advanced: Mining Biomedical Texts

22.6 Advanced: Script-Based Text Understanding

22.7 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 detection followed by classification.

- **Named entities** can be recognized and classified by **statistical sequence labeling** techniques.
- **Relations among entities** can be detected and classified using supervised learning methods when annotated training data is available; lightly supervised **bootstrapping** methods can be used when small numbers of **seed tuples** or **seed patterns** are available.
- Reasoning about time can be facilitated by detecting and normalizing **temporal expressions** through a combination of statistical learning and rule-based methods.
- Rule-based and statistical methods can be used to detect, classify and order **events** in time. The **TimeBank corpus** can facilitate the training and evaluation of temporal analysis systems.
- **Template-filling** applications can recognize stereotypical situations in texts and assign elements from the text to roles represented as **fixed sets of slots**.
- Information extraction techniques have proven to be particularly effective in processing texts from the **biological domain**.
- Scripts, plans and goals...

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). Chinchor et al. (1993) describes the evaluation techniques used in the MUC-3 and MUC-4 conferences. Hobbs (1997) partially credits the inspiration for FASTUS to the success of the University of Massachusetts CIRCUS system (Lehnert et al., 1991) in MUC-3. The SCISOR system is another system based loosely on cascades and semantic expectations that did well in MUC-3 (Jacobs and Rau, 1990).

Due to the difficulty of reusing or porting systems from one domain to another, attention shifted to the problem of automatic knowledge acquisition for these systems. 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 the work of 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 (Sassano and Utsuro, 2000; McNamee and Mayfield, 2002).

Progress in this area continues to be stimulated by formal evaluations with shared benchmark datasets. The MUC evaluations of the mid-1990s were succeeded by the Automatic Content Extraction (ACE) program evaluations held periodically from 2000 to 2007.[5] These evaluations focused on the named entity recognition, relation detection, and temporal expression detection and normalization tasks. Other IE evaluations included the 2002 and 2003 CoNLL shared tasks on language-independent named entity recognition(Sang, 2002; Sang and De Meulder, 2003), and the 2007 SemEval tasks on temporal analysis(Verhagen et al., 2007), people search(Artiles et al., 2007).

The scope of information extraction continues to expand to meet the ever-increasing needs of applications for novel kinds of information. Some of the emerging IE tasks that we haven't discussed include the classification of gender (Koppel et al., 2002), moods (Mishne and de Rijke, 2006), sentiment, affect and opinions (Qu et al., 2004). Much of this work involves **user generated content** in the context of **social media** such as blogs, discussion forums, newsgroups and the like. Research results in this domain have been the focus of a number of recent workshops and conferences (Nicolov et al., 2006; Nicolov and Glance, 2007).

USER GENERATED CONTENT

SOCIAL MEDIA

EXERCISES

22.1 Develop a set of regular expressions to extract the character shape features described in Fig. 22.7.

22.2 Using a statistical sequence modeling toolkit of your choosing, develop and evaluate an NER system.

22.3 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. Perform experiments and compare its performance against the scheme presented in this chapter.

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

22.4 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 (eg. 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.

22.5 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.

22.6 Acronym expansion, the process of associating a phrase with a particular 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 against the Wikipedia TLA page (`en.wikipedia.org/wiki/Category:Lists_of_TLAs`).

22.7 Collect a corpus of biographical Wikipedia entries of prominent people from some coherent area of interest (sports, business, computer science, linguistics, etc.). Develop a system that can extract an occupational timeline for the subjects of these articles. For example, the Wikipedia entry for Peter Norvig might result in the ordering: Sun, Harlequin, Junglee, NASA, Google; the entry for David Beckham would be: Manchester United, Real Madrid, Los Angeles Galaxy.

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

22.9 Develop and evaluate a recognition system capable of recognizing temporal expressions of the kind appearing in your email corpus.

22.10 Design a system capable of normalizing these expressions to the degree required to insert them into a standard calendaring application.

22.11 Acquire the CMU seminar announcement corpus and develop a template-filling system using any of the techniques mentioned in Sec. 22.4. Analyze how well your system performs as compared to state-of-the-art results on this corpus.

22.12 Develop a new template that covers a situation commonly reported on by standard news sources. Carefully characterize your slots in terms of the kinds of entities that appear as slot-fillers. Your first step in this exercise should be to acquire a reasonably sized corpus of stories that instantiate your template.

22.13 Given your corpus, develop an approach to annotating the relevant slots in your corpus so that it can serve as a training corpus. Your approach should involve some hand-annotation, but should not be based solely on it.

22.14 Retrain your system and analyze how well it functions on your new domain.

22.15 biology

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*.

Ahn, D., Adafre, S. F., and de Rijke, M. (2005). Extracting temporal information from open domain text: A comparative exploration. In *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR'05)*.

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

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., and Tyson, M. (1995). SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Francisco, pp. 237–248. Morgan Kaufmann.

Appelt, D. E. and Israel, D. (1997). ANLP-97 tutorial: Building information extraction systems. Available as `www.ai.sri.com/~appelt/ie-tutorial/`.

Artiles, J., Gonzalo, J., and Sekine, S. (2007). The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, pp. 64–69. Association for Computational Linguistics.

Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

Bikel, D. M., Schwartz, R., and Weischedel, R. (1999). An algorithm that learns what's in a name. *Machine Learning*, *34*, 211–231.

Bunescu, R. C. and Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 724–731.

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.

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.

Culotta, A. and Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.

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.

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

Fisher, D., Soderland, S., McCarthy, J., Feng, F., and Lehnert, W. G. (1995). Description of the UMass system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Francisco, pp. 127–140. Morgan Kaufmann.

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*.

Gaizauskas, R., Wakao, T., Humphreys, K., Cunningham, H., and Wilks, Y. (1995). University of Sheffield: Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Francisco, pp. 207–220. Morgan Kaufmann.

Grishman, R. and Sundheim, B. (1995). Design of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Francisco, pp. 1–11. Morgan Kaufmann.

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, Berlin.

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

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

Koppel, M., Argamon, S., and Shimoni, A. R. (2002). Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, *17*(4), 401–412.

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.), *Proceedings of the Third Message Understanding Conference*, pp. 223–233. Morgan Kaufmann.

McCallum, A. (2005). Information extraction: Distilling structured data from unstructured text. *ACM Queue*, 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 *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2002)*, Taipei, Taiwan.

Mikheev, A., Moens, M., and Grover, C. (1999). Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Morristown, NJ, USA, pp. 1–8. Association for Computational Linguistics.

Mishne, G. and de Rijke, M. (2006). MoodViews: Tools for blog mood analysis. In Nicolov, N., Salvetti, F., Liberman, M., and Martin, J. H. (Eds.), *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, Stanford, Ca. AAAI.

Nicolov, N. and Glance, N. (Eds.). (2007). *Proceedings of the First International Conference on Weblogs and Social Media (ICWSM)*, Boulder, CO.

Nicolov, N., Salvetti, F., Liberman, M., and Martin, J. H. (Eds.). (2006). *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, Stanford, Ca. AAAI.

Peshkin, L. and Pfefer, A. (2003). Bayesian information extraction network. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

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

Pustejovsky, J., Hanks, P., Saur, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., and Lazo, M. (2003b). The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003 Conference*, pp. 647–656.

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

Qu, Y., Shanahan, J., and Wiebe, J. (Eds.). (2004). *Exploring Attitude and Affect in Text: Papers from the 2004 Spring Symposium*, Stanford, Ca. AAAI.

Ray, S. and Craven, M. (2001). Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*.

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 *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, pp. 474–479.

Roth, D. and tau Yih, W. (2001). Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1257–1263.

Sang, E. F. T. K. (2002). Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pp. 155–158. Taipei, Taiwan.

Sang, E. F. T. K. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent

named entity recognition. In Daelemans, W. and Osborne, M. (Eds.), *Proceedings of CoNLL-2003*, pp. 142–147. Edmonton, Canada.

Sassano, M. and Utsuro, T. (2000). Named entity chunking techniques in supervised learning for japanese named entity recognition. In *COLING-00*, Saarbrcken, Germany, pp. 705–711.

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

Settles, B. (2005). ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, *21*(14), 3191–3192.

Soderland, S., Fisher, D., Aseltine, J., and Lehnert, W. G. (1995). CRYSTAL: Inducing a conceptual dictionary. In *IJCAI-95*, Montreal, pp. 1134–1142.

Sundheim, B. (Ed.). (1991). *Proceedings of the Third Message Understanding Conference*. Morgan Kaufmann.

Sundheim, B. (Ed.). (1992). *Proceedings of the Fourth Message Understanding Conference*. Morgan Kaufmann.

Sundheim, B. (Ed.). (1993). *Proceedings, Fifth Message Understanding Conference (MUC-5), Baltimore, MD*. Morgan Kaufmann.

Sundheim, B. (Ed.). (1995). *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., and Pustejovsky, J. (2007). Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, pp. 75–80. Association for Computational Linguistics.

Weischedel, R. (1995). BBN: Description of the PLUM system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Francisco, pp. 55–70. Morgan Kaufmann.

Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, Michigan, pp. 427–434. Association for Computational Linguistics.