Information Extraction: Capabilities and Challenges

Notes prepared for the
2012 International Winter School in Language and Speech Technologies
Rovira i Virgili University
Tarragona, Spain

Ralph Grishman

January 21, 2012

Contents

Chapter 1

What Is Information Extraction?

Information extraction (IE) is the process of identifying within text instances of specified classes of entities and of predications involving these entities. An early and oft-cited example is the extraction of information about *management succession* – executives starting and leaving jobs.[1] If we were given the text

> Fred Flintstone was named CTO of Time Bank Inc. in 2031. The next year he got married and became CEO of Dinosaur Savings & Loan.

we would produce

person	company	position	year	in/out
Fred Flintstone	Time Bank Inc.	CTO	2031	in
Fred Flintstone	Time Bank Inc.	CTO	2032	out
Fred Flintstone	Dinosaur Savings & Loan	CEO	2032	in

A few things to note about this table:

- information about his marriage was not captured above; extraction seeks in general to cover only a predefined set of predications.
- the second sentence refers to "he", but that would not be very meaningful in the table; we resolve "he" to the name of the entity being referenced, "Fred Flintstone".
- the two events use quite different choices of words to express the event ("was named" and "became"), but the extraction procedure recognizes these as instances of the same type of event. Such recognition of paraphrases is a critical aspect of information extraction.

Why do information extraction? A short answer is that it makes the information in the text *more accessible* for further processing. Getting a person's employment history from a large collection of news articles is a complex procedure; getting it from a table such as that shown above is far easier. Similarly, searching for an interaction of two particular genes in the scientific literature will be much easier if these relations have been extracted in advance.

This is not a new idea. Zellig Harris, over 50 years ago, described how one might identify the basic semantic structures in a field of science, and then map scientific articles into these structures [Har58]. These structures could then be used for literature searches. While we have made progress in information extraction (and in natural language processing more generally) in these 50 years, we are still some ways away from fully realising Harris's vision.

[1]This is a simplified version of the task for Message Understanding Conference-6, to be discussed below.

A History of Evaluations One notable characteristic of IE is the degree to which its research has been driven by a series of U.S. Goverment-sponsored evaluations. By the mid-1980's there had been several efforts at information extraction from news [DeJ82] and medical reports [SFLmotLSP87], but evaluation was limited, particularly with regard to comparing one system with another, so it was hard to tell whether progress was being made.

To better assess progress, the Navy convened the first Message Understanding Conference (MUC-1) in 1987. The development corpus consisted of 10 Navy oprep messages, each a few sentences. Participants each proposed a suitable output for each message.

By MUC-3 the process was more formalized, with a standard data base template, common test data, and a standard scoring metric. These meetings / evaluations continued through MUC-7 in 1998, adding additional tasks in MUC-6 and MUC-7.[2]

Each of the MUC evaluations was focused on a single topic. The next series of evaluations, Automatic Content Extraction (ACE) aimed to cover news stories more generally, including politics and international affairs, through a broader range of entities, relations, and events. ACE had roughly annual evaluations from 2000 to 2008.[3] Later versions of ACE included multi-lingual extraction, with texts in English, Chinese, and Arabic. Although there was a small effort at cross-document analysis, almost all the processing in ACE was on a document-by-document basis ... systems were judged separately on their ability to correctly extract the information in each document.

This is the primary aspect which has shifted with the current series of evaluations, Knowledge Base Population (KBP), part of the Text Analysis Conferences.[4] In KBP the extraction system is given a name, an article containing that name (so that, if you are given "John Smith", you know which John Smith), and a collection of about 2 million news articles and blog posts and has to gather information about the named person or organization. This raises various questions about search, name variation, redundant and conflicting information which were not addressed in the earlier evaluations. KBP started in 2009 and is continuing each year.

What's to come In the chapters which follow, we will consider different aspects of information extraction, starting with names and building up to larger units. Our focus will be primarily on extraction from general news; we will look at some other domains in the final chapter. The original IE systems, and many currently deployed systems, rely on hand-coded rules, but there has been a clear movement towards *trainable* systems, a trend reflected in these notes. Accordingly, before getting into IE, we shall very briefly review some of the concepts in machine learning we will be using,

The literature, particularly for the last fifteen years, has reported significant advances in IE, and we will review some of these reports. Just reading these papers, however, may give an incomplete and overly rosy picture to someone who has to build an IE system of their own. We will therefore complement the literature summaries with our observations regarding issues often overlooked and the many challenges which remain.

[2]Proceedings for MUC-3 through MUC-7 are available through the ACL web site, http://aclweb.org/anthology-new/

[3]ACE task descriptions are available through NIST at http://www.itl.nist.gov/iad/mig/tests/ace/

[4]Text Analysis Conference proceedings are available at http://www.nist.gov/tac/publications/index.html

Chapter 2

Machine learning preliminaries

2.1 Classifiers

A classifier is given a data item x and assigns it a label y taken from a finite set of labels. If there are only two labels, it is a *binary classifier*, else an *n-ary classifier*. In general, each data item will be treated as a set of features and their values. In NLP, most features take on discrete values.

A trainable classifier accepts as input some labeled data $(x_1, y_1), ..., (x_n, y_n)$ and from this training data produces a classifier $C(x)$ which can operate on any input x.

For the most part, we will treat the trainable classifiers as "black boxes" and will not be concerned with their internal operation. Two of the most widely used types of classifiers in NLP are *maximum entropy classifiers* (MaxEnt) and *support vector machines* (SVM).

Maximum entropy is a form of multinomial logistic regression. The probability that data item x has class label c is given by

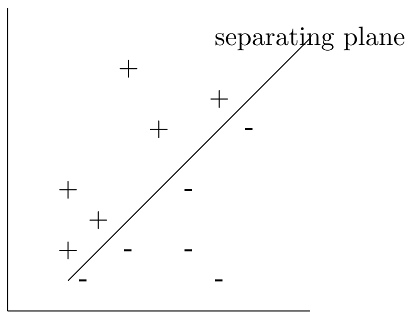
$$p(c|x) = \frac{1}{Z} \exp \sum_{j=0}^{N} w_j h_j$$

where Z is a normalizing constant, h_j is the value of the i$^{\text{th}}$ *indicator function*, and w_j is the weight assigned to indicator function j by the training process. Each indicator function is a combination of a particular class label c and a particular feature of data item x. In most cases features will be binary (true or false). The indicator function will take the value 1 if the feature is true for that class, otherwise 0. For example, for a part-of-speech tagger,h_{23} might be "the word ends in *ly* and the class is *adverb*"; h_{24} might be "the word ends in *ly* and the class is *adjective*". A positive value of w_j means that the feature makes the class more likely; a negative value of w_j means that the feature makes the class less likely. Thus we would expect w_{23} to be positive and w_{24} to be negative.

One basic consequence of this formula is that features combine multiplicatively. If a conjunction of two features behaves quite differently from the two separately, it is necessary to explicitly code the conjunction as a third feature.

Maximum entropy can naturally handle n-way classification.

Support vector machines (SVMs) are *binary* classifiers for data items with N features. The simplest situation for SVMs occurs if the data items are linearly separable; in other words, if there exists a hyperplane in N dimensional space such that all positively labeled items are on one side of the plane and all negatively labeled items are on the other side. There may in general be many such hyperplanes; the SVM training procedure selects the plane which maximizes the distance from the plane to the positive and negative training examples. Once a plane has been selected, a new data point can be classified by seeing on which side of the plane it falls.



Such a *linear SVM* will be successful if the data points are (nearly) linearly separable. By using a different *kernel function* in the SVM, it is possible to handle data which is only separable in a higher-dimension space. Through this "kernel trick" it is possible to systematically include combinations of the original features.

2.2 Sequence models

In some situations, we are classifying data items separately. For example, in doing text classification for a collection of documents, each one will be classified independently. Such data is treated as independent and identically distributed ("i.i.d.").

On the other hand, there are many cases in NLP where we are classifying items appearing in a sequence, and the classification of one item affects the classification of others in the sequence. A typical example is part-of-speech tagging: if one word is a determiner, the next one is not likely to be a verb.

Imagine first a random process for generating sequences of words taken from a vocabulary of N different words. In general, the i$^{\text{th}}$ word can depend on all the previous words. We can simplify the model, creating a [first-order] Markov process, by assuming that the choice of a word depends only on the immediately preceding word, and not on those that come before. This *Markov Model* is conventionally represented by a graph, with states labeled by words and transitions represented by arcs between the states. Associated with each arc is the probability of a transition from state i to state j.

We can build on this idea to create a sequential classifier. We do this by separating the notion of state from the specific words in the sequence, and allowing each state to be associated with a set of words. Consider for example the task of part-of-speech tagging: we use a model with one state for each part of speech. We create a probabilistic model which captures the probability of part-of-speech sequences and the association of particular words with particular parts of speech. Given a sentence w_1 ... w_n, using this model, we determine the most likely sequence of states s_1 ... s_n:

$$S = \arg\max_{s_1 ... s_n} P(s_1...s_n|w_1...w_n)$$

Suppose the current word depends only on the current state. Then (using Bayes' rule)

$$S = \arg\max_{s_1 ... s_n} P(w_1...w_n|s_1...s_n)P(s_1...s_n)$$

$$= \arg\max_{s_1 ... s_n} \prod_{i=1}^{n} P(w_i|s_i)P(s_i|s_{i-1})$$

This is the formula for a *Hidden Markov Model* (HMM). The first probability is the *emission probability* – the probability that one will emit a given word in a particular state. The second probability is the *transition probability.*

An HMM can be used to label sequences of items. First, the HMM can be trained using an annotated corpus, where the state associated with each word (for example, the word's part of speech) has been labeled. In using the HMM to tag new data, we find the sequence of states which is most likely to have generated the sequence of words. This can be done efficiently by a form of dynamic programming called *Viterbi decoding.*

Hidden Markov Models work quite well for a number of sequential tagging tasks. However, it is quite difficult to improve their performance by incorporating linguistic insights. For example, suppose we had reason to believe that a token enclosed in parentheses was more likely to be a noun. To capture this information, we would have to add several states: one for "(", one for the token following "(", and one for ")". If we had several such contexts we wanted to capture, the graph would get quite complex.

Maximum Entropy Markov Models (MEMM) make this job much simpler. In an MEMM, the probability of the next state is computed as a function of the current state and the words in the sentence:

$$S = \arg\max_{s_1 \ldots s_n} \prod_{i=1}^{n} P(s_i | s_{i-1} w_1 \ldots w_n)$$

Here P is implemented by a MaxEnt model. Note that P depends only on the immediately preceding state (Markov constraint) but can access the entire word sequence. This gives great flexibility in capturing insights by adding separate indicator functions (feature / outcome pairs) to the model.

Although MEMMs are very flexible, they can suffer from the *label bias problem.* This problem is avoided by a more general class of sequence models, *conditional random fields.*

2.3 Flavors of learning

When we mentioned training in the last two sections, we had in mind a situation where the input to the training procedure consisted entirely of labeled data points. This is termed *supervised training.* Supervised training makes a lot of sense if there are large amounts of naturally occurring labeled data. For example, if you are training a case-restoration procedure (which takes monocase input and capitalizes words appropriately), you can use large quantities of mixed-case text on the web. On the other hand, if the training data needs to be labeled by hand (for example, for a part-of-speech tagger), supervised training can be expensive.

An alternative is a *semi-supervised training* procedure, which takes as input some labeled data and some (typically much more) unlabeled data [Abn08]. Semi-supervised training relies on having a classifier which can not only classify new data but can estimate how confident it is in its classification. For example, for a binary probabilistic classifier, we can use the probability to estimate the confidence; probabilities near 0 or near 1 would represent high confidence, while probabilities near 0.5 would represent lower confidence.

Semi-supervised training involves two sets of data, the labeled data L and the unlabeled data U. The initial set of labeled data (provided as an input to the process) is termed the *seed.* The process works iteratively, gradually building up the set L and thereby improving the classifier. At each iteration, we train the classifier on L, apply the classifier to U, and then take the most confidently classified data points, remove them from U and add them (with their labels) to L.

One particular variant of semi-supervised training is *co-training.* Co-training uses a pair of trainable classifiers C_1 and C_2, which are applied in alternation. We train C_1 on L, apply it to U, take the most confidently labeled points and add them to L. Then we do the same with C_2, then C_1 again, etc. Ideally for co-training the two 'views' (the sets of features used by the two classifiers) are conditionally independent and are each sufficient to label the data. Real NLP tasks rarely meet these conditions, but if they come close, co-training can be quite effective, as we shall see in later chapters.

One problem with semi-supervised learning is the lack of good stopping criteria. Training can be done for a fixed number of iterations or until no new data is labeled. We will consider this problem in connection with specific applications of self-training.

Another way of reducing the amount of annotated data which is required is through *active learning.* In preparing annotated data for supervised learning, we typically annotate data in sequence, going through a collection document-by-document. This is an inefficient scheme for many natural language tasks, because of the Zipfean distribution of many natural language phenomena – a few example types are very common, while many are rare. Doing part-of-speech tagging, one might tag hundreds of instances of "the" or "said" before tagging a single instance of "cow" or "milked". Tagging "the" as a determiner for the hundredth time does little to improve the part-of-speech model. Active learning seeks to select informative examples for an annotater to tag – examples which are likely to reduce the error rate of the model. We will look at active learning in more detail in our discussion of name tagging.

Chapter 3

Extracting Names

3.1 Some History

We will begin by looking at a restricted IE task, the task of identifying and classifying *names* in text. Consider the text from Chapter 1:

> Fred Flintstone was named CTO of Time Bank Inc. in 2031. The next year he got married and became CEO of Dinosaur Savings & Loan.

If we began by labeling the names of people and organizations in the text, something like this:

> [Fred Flintstone]$^{\text{person}}$ was named CTO of [Time Bank Inc.]$^{\text{org}}$ in 2031. The next year he got married and became CEO of [Dinosaur Savings & Loan]$^{\text{org}}$.

it would be simpler to extract the management succession events.

In the course of developing systems for Message Understanding Conferences 3, 4, and 5, researchers began to see the central role that name identification and classification played in information extraction. Names are very common in text, particularly in news text; furthermore, typically many of the columns in an extracted table are to be filled with names. This led to the introduction of a new, separately evaluated task called *named entity recognition* (NER) as part of MUC-6 [GS96]. Three classes of names were recognized: persons, organizations, and locations.

NER proved a very popular task and grew along several dimensions. NER was tried for a number of different languages, initially as part of the MET and CoNLL evaluations [TKS03, TKSDM03]. By now, named entity recognizers have been developed for several dozen languages. There is a general recognition that NER is a valuable preprocessing step for many types of language analysis, not just for IE. For example, NER can be helpful to MT because names are typically translated very differently from the words which make up the name. NER has become an essential component of Question Answering because the answer to a question is very often a name of a particular type.

The role in Question Answering in particular has motivated a broadening and refinement of the categories of names. For example, the organization category has been divided into government organizations, sports organizations, commercial organizations, etc. Categories which have been added include works of art (books, movies, plays, ...), manufactured products (cars, foods, drinks, ...), and legal documents (Schengen Agreement, Dodd-Frank Act). The result has been Named Entity sets with 100-200 categories arranged in a hierarchy [SN04]. These often include semantic categories which not everyone would consider "names" (for example, colors).

Most of these NE sets have been designed for news texts and other "general interest" materials. Specialized technical texts call for quite different sets of categories. For example, there has been a great deal of work over the past decade on genomics, where the main categories of entities are genes and proteins (see Chapter 8).

3.2 Hand-coded Methods

One straightforward way of finding names is to write a set of patterns (regular expressions) for the different categories of names, with each pattern using some clue to identify the type of name. The pattern

title [capitalized-token+]$^{\text{person}}$

where *title* matches "Mr.", "Mrs.", "Miss", "Ms.", "Dr.", "Prof.", ... , and *capitalized-token +* matches one or more capitalized tokens, can find person names. Here the brackets [...] indicate the portion of the matched sequence that is the actual name; the title is ordinarily not considered part of a person's name. Middle initials generally indicate person names:

[capitalized-token initial capitalized-token]$^{\text{person}}$

where *initial* matches "A." ... "Z.". Common first names can also be helpful:

[common-first-name capitalized-token]$^{\text{person}}$

although these will differ from country to country.[1]

Organization names can often be identified by their final token ("Company", "Associates", "Motors", ...) or some suffix ("Inc.", "Ltd.", "Corp.", ...). The best known company names will often appear without any contextual clues, so a list of these names will be required. Lists are also needed for the best known locations: countries, states (in the U.S.), and major cities. Minor cities will often appear with the phrase "city of" or followed by ", state" or ", country".

Hand-coded systems often rely on extensive lists of people, organizations, locations, and other entity types. Many such lists are now available from the Web. These lists can be quite helpful, and for some entity types they are essential because good contextual patterns are not available, but they should be used with some caution. Systems which are primarily list based will suffer when new names arise, either because of changes in the input texts or the passage of time. Also, large lists may include entries which are proper names but more often are capitalized forms of common words (for example, the towns of Why (Arizona), Normal (Illinois), Kissing (Germany) and Hell (Norway)); such entries may need to be pruned before using these lists for NER.

By building up a set of rules for each entity type, it is possible to create a quite effective NER system. The task is somewhat easier for mixed-case English text because capitalization provides a good clue for identifying (although not for classifying) names. However, obtaining high performance does require some degree of skill. It also generally requires an annotated corpus which can be used to evaluate the rule set after each revision; without such a corpus there is a tendency – after a certain point – for added rules to actually worsen overall performance. We will discuss evaluation later in this chapter.

3.3 Supervised Methods

If the creation of a high-quality NER by hand requires an annotated corpus, it is natural to ask whether an NER can be trained *automatically* from such a corpus. Such *supervised methods* for training an NER will be considered in this section.

[1]For the U.S., lists of common first names are available from the Census. 1000 common first names account for 85% of all men and 75% of all women.

3.3.1 Using Sequence Models

In order to build a trainable model, we need first of all some way of enumerating the possible outputs of NER for a sentence. This can be done in a straightforward way through a set of token-level tags such as *BIO tags* (we shall consider other tagging schemes shortly). In the BIO scheme, the first token of a name of type c is assigned tag *B-c*; subsequent tokens of the same name are assigned tag *I-c*; tokens which are not part of a name are assigned tag *O*. A token tagged *I-c* must be preceded by a token tagged *B-c* or *I-c*. For example,

Fred	Flintstone	met	Wilma	in	Madrid	yesterday
B-person	I-person	O	B-person	O	B-location	O

This reduces the problem of NER to one of assigning one of these tags to each token in sequence. This problem of *sequence modeling* has been extensively explored by the machine learning community, and a variety of sequence models have been developed, including HMMs, MEMMs, and CRFs (see Section 2.2). Because NER is a relatively simple and intuitive task, all of these types of models, and several others, have been applied to the NER task.

Nymble [BMSW97], one of the first trainable NER systems, was based on an HMM with a single state for each name type, plus one state for "other". Conditional probabilities were used to capture contextual information. Maximum entropy methods were introduced for NER at MUC-7 by Borthwick [BSAG98], [Bor99] and by the Edinburgh group [MGM98]. [NS07] provides a recent survey of much of the work in named entity recognition.

3.3.2 States

The description just above would suggest that one should have $2n + 1$ states corresponding to the $2n + 1$ different tags. If using an MEMM (or CRF), this is a reasonable starting point. Information that particular words (e.g., "Mr.", "Ms.") precede a B-person state can be captured by including the preceding word as a MaxEnt feature.

For an HMM, the word emitted only depends on the current state, so a different trick must be employed: introducing additional states such as *pre-person* for the token preceding a person name and *post-person* for the token following a person name. When the HMM is trained, "Mr." and "Ms." will be emitted primarily from the pre-person state.

For MEMM or HMM, some gain can be obtained by having multiple states which reflect the internal structure of names. Borthwick used a model with 4 states per name category, C_start, C_continue, C_end, and C_unique [BSAG98]. Ratinov and Roth [RR09] compared such a scheme (which they dubbed BILOU) with a stardard BIO scheme and demonstrated a performance gain of roughly 1%.

3.3.3 Local Features

The power of a feature-based tagger is that such a wide variety of features can be brought to bear. These include

- lexical features: whether the current (preceding, following) word has a specific value
- dictionary features: whether the current word is in a particular dictionary; these may include lists of full names (names of major companies, countries, states, cities) and name components (e.g., common first names)
- shape features: the form of the token – whether it is all lower case, capitalized, all caps, numeric (including typical forms of dates, phone numbers, etc.
- part-of-speech features

These features, by themselves or conjoined with the prior state, form the heart of an NE tagger. Beyond these, it is possible to include any pattern which might be useful in identifying some names. If one has a hand-coded pattern set for names, one can include each of these patterns as a feature, thus gaining the benefit of both hand-coded patterns and machine-learned patterns [BSAG98].

3.3.4 Long-Range Features

Most names represent the same name type (person / org / location) whereever they appear, particularly within a single document but in most cases across documents as well.[2] Some contexts will provide a clear indication of the name type, while others will be ambiguous. We would like to use the unambiguous contexts to resolve the ambiguity across the document or across the corpus. For example,

> On vacation, Fred visited Gilbert Park.
> Mr. Park was an old friend from college.

We can capture this information with a two-pass strategy. On the first pass, we build a table ("name cache") which records each name and the type(s) it is assigned. Then on the second pass we incorporate a feature reflecting the types, or the dominant name type, from the first pass. This can be done across an individual document or across the corpus. Borthwick observed an improvement of 1 to 2% using this feature [Bor99]. Ratinov and Roth tried several variants of this scheme, getting anywhere from 1% to 4% gain [RR09].

3.4 Semi-supervised Methods

Supervised methods spare us the task of writing rules by hand but still require substantial labor to prepare an annotated corpus. Can we reduce the amount of corpus which we need to annotate? This is possible through *semi-supervised* learning methods, including in particular those based on *co-training*. Semi-supervised methods make use of limited amounts of labeled data together with large amounts of unlabeled data.

The basic observation, already made use of in the long-range features of the supervised tagger, is that multiple instances of the same name have the same type. Some of these instances may occur in contexts which are indicative of a particular name type, and these may be used to tag other instances of the same name. In semi-supervised learning we apply these principles repeatedly: starting from a few common names of each type, we look for the contexts in which each of these names appears. If a context appears only with names of one type, we treat this as a predictive context; we look for other names which appear in this context and tag them with the predicted type. We add them to the set of names of known type, and the process repeats.

This is an example of *co-training*. In co-training, we have two *views* of the data – two sets of features which can predict the label on the data (see Section 2.3). In our case the two views are the context of a name and the 'spelling' of a name (this includes the complete name and the individual tokens of the name). For each view, we are able to build a model from the labeled data; then we can use this model to label the unlabeled data and associate a *confidence* with each label. We build a model based on the first view, generate a label for each unlabeled datum, and keep the most confident labels, thus growing the labeled set. Then we do the same using the second view. Gradually the labeled set grows and the models are refined.

This approach was described by Collins and Singer [CS99] for mixed-case English newspaper text using the usual three categories, person, organization, and location. Starting with a small seed, they were able to get very good results: 83% correct classification (91%

[2]There are exceptions, of course. Some people names are also location names ("Washington"). Sometimes the name of a person also refers to a company or product line ("Perry Ellis").

if names not belonging to any of these 3 categories were excluded). However, it should be noted that these results were the consequence of favorable test conditions. They assumed that the names were already identified and only needed to be classified; accurate identification in mixed-case English is not difficult (because names are capitalized) but is not trivial. They only considered names occurring in specific syntactic contexts. And they used a set of name categories which covers most instances of names in the news.

Attempts at semi-supervised training of NER under less favorable conditions have proven much more difficult. One problem is *semantic drift* ... a procedure which is given a seed consisting of one type of name gradually starts assigning that label to names of another type. This is particularly likely if the two sets of names intersect, such as women's first names and names of flowers (Rose, Violet, ...). More generally, these bootstrapping methods often lack a good stopping criterion, and so tend to label too many examples.

This problem is less severe if we are assigning labels to all the examples and learn all the types concurrently. For example, capitalization is a fairly good clue to identifying names in English, and most names in news text are people, organizations, or locations. So concurrent self-training on these three types can be quite effective.

It is considerably more difficult to learn to tag technical terms, where capitalization does not help in identification. Here Yangarber *et al.* have shown the benefit of 'competition', where one identifies and adds additional name classes (and trains on all the classes concurrently) in order to improve the training of the original classes [YLG02]. A similar approach was taken in Weighted Mutual Exclusion Bootstrapping (WMEB) by McIntosh and Curran [MC08]. The disadvantage of this approach is that it requires a manual analysis and creation of these competing classes.

The manual creation of competing classes can be avoided by using unsupervised term clustering (based on all the contexts in which a term appears) to create these negative categories [McI10].

3.5 Active Learning

Another approach to reducing the amount of data which must be annotated is *active learning.* In preparing for conventional supervised learning, one selects a corpus and annotates the entire corpus from beginning to end. However, the distribution of names in a typical news corpus is likely to be very skewed – you are likely to tag hundreds of instances of "George" or "Barak" before encountering the first instance of "Tara" or "Alisa". This is not very efficient – at some point, tagging one more instance of "George" as a person name doesn't improve the performance of the model at all. The idea of active learning involves having the system select examples for the user to annotate which are likely to be informative – which are likely to improve the accuracy of the model.

How do we decide what are the most informative examples? There are two standard approaches: uncertainty-based sampling and committee-based sampling. In uncertainty-based sampling, we use some estimate of the uncertainty regarding the label for a data point; for example, for a probabilistic classifier involving a binary choice, how close the probability is to 0.5. If there are many possible labelings (for example, all the name labelings of a sentence) the margin (the difference of the probability of the top two hypotheses) can be used [SDW01]. In committee-based sampling, we create two or more classifiers (based, for example, on different subsets of the features) and prefer data points on which they disagree. If the classifiers produce a probability distribution over labels, we can use a measure of the degree to which these two distributions differ (for example, KL-divergence).

Selecting outliers With sequential labeling, common features will be heavily represented in the training set. With uncertainty-based sampling, the active learner may prefer features or combinations of features which have never been seen before, and even the most common features may only appear once in the training set. Furthermore, in choosing among features not yet in the training set, the active learner won't distinguish between rare features and common features, even though labeling items with common features may do much more to improve the classifier. To provide better balance, some active learners include a representativeness weight, so that data points involving common features are preferred to those involving rare features.

Compute time In principle, we want to select an example, label it, update the model, and then select the next example. But updating the model and re-evaluating all the unlabeled points may be slow. A natural solution is to select a batch of data points at each iteration. Unfortunately, if the same criteria and model are used, the data points selected in the batch are likely to be very similar, defeating the benefits of active learning. So systems which do batch active learning need to add a diversity condition – requiring that the data points in the batch not be too similar.

Annotation speed and the unit of annotation Speed-up is typically measured in terms of number of items annotated. However, this may not be a realistic measure of true speed-up (reduction in labeling time) in some cases. Suppose you are training a named-entity classifier, and at each iteration you ask the user about the class of a single named entity. If you ask the user about a single name in the middle of a document, the user has to read at least the containing sentence and perhaps several prior sentences. The burden per name annotated is considerably less for sequential annotation. This suggests that the optimal unit of annotation may be at least an entire sentence.

Simulated active learning Real active learning implies a person in the loop. That is a very costly approach if one wants to conduct multiple experiments with different active learning strategies. Consequently most A.L. experiments use simulated A.L., where the training corpus is actually labeled in advance but the labels are revealed to the learner only on request.

Active learning for named entities Active learning has been applied to many NLP tasks ... part-of-speech tagging, parsing, machine translation. There have been a few efforts to apply it to named entity tagging:

[BHAG05] describes a system developed a large annotated biomedical corpus (GENIA) and then applied to a new subdomain (astronomy) with limited test data. They used a committee-based method involving two Markov Model name taggers with different feature sets and selected sentences based on difference of the probability distributions of NE tags. They report typical reductions of 40% in labels required to reach the same level of performance.

[SZS$^+$04] uses an SVM as binary discriminative classifier to decide whether a token is a member of a given name class (it tags just one type of name at a time). This procedure incorporates all 3 measures (informativeness, representativeness, and diversity), using distance to the hyperplane to measure informativeness. It was applied to both MUC-6 and GENIA tasks, achieving typically a 60% reduction in label data required for the same level of performance.

3.6 Evaluation

Evaluating NER requires first of all a clear statement of the task to be performed. This may seem obvious, but the difficulty of preparing clear guidelines is often overlooked. Even with three categories (people, organizations, locations) difficult cases arise. Should characters in books and movies be labeled as people? Should planets be labeled as locations? Is "New York Criminal Court" one name or two?

One general problem is that of systematic name polysemy. Names of organizations also represent the locations they occupy ("I'll meet you in front of MacDonald's"); names of locations also represent their governments ("Germany signed a treaty with the Soviet Union.") and sometimes the population or industry ("Japan eats more tuna that the U.S."). A basic decision must be made whether to tag a name based on its primary meaning or its meaning in context. The ACE evaluations, for example, tagged based on the primary meaning of a name. In recognition of the particular problem with locations, ACE introduced the category *GPE*, Geo-Political Entity, for locations with governments. "Japan" would be tagged as a GPE whether used in a context representing the land, the government, or the population. The category *location* was reserved for natural landmarks ("the Atlantic Ocean", "Mt. Everest").

In preparing an evaluation, it is also important to characterize the test corpus. As we have already noted, the performance of an NER can be very sensitive to these characteristics. In particular, a supervised model may do very well if the test corpus is similar to the training corpus – possibly because it has simply memorized lots of names from the training corpus – and may rapidly degrade for more varied corpora. We should be aware of the source of the texts, the epoch of the texts, and whether they were limited to particular topics.

In general, if annotators are familiar with the topics discussed in a text, name annotation (for a small set of name types) quality can be high. However, if annotators are not familiar with the topic, they may have difficulty correctly assigning name types and, in languages where names are not distinguished by case, in distinguishing names from non-names.

Comparing the name tags produced by an NER against an annotated reference corpus, we mark tags as

correct the tags agree in extent and type

spurious a tag in system output with no matching tag in the reference corpus

missing a tag in the reference corpus with no matching tag in the system output

From these we compute

$$\begin{aligned} recall &= \frac{correct}{correct + missing} \\ precision &= \frac{correct}{correct + spurious} \\ F &= \frac{2 \times precision \times recall}{precision + recall} \end{aligned}$$

The MUC evaluation gave part credit for a system name tag which matched the reference in extent but did not match in type. More recent papers generally follow the CoNLL scoring, which gives no credit for such partial matches (so that an error in type is considered a spurious and a missing tag). As a result, the early MUC scores cannot be directly compared with more recent results.

For the CoNLL multi-site evaluations in 2002 and 2003 [TKS03, TKSDM03], the best named entity F scores were:

English	88
Spanish	81
Dutch	77
German	72

These tests were run with uniform training and test corpora (in the case of English, the Reuters newswire). Performance degrades sharply if there are differences between training and test, even if both are news corpora. Ciaramita and Altun reported that a system trained on the CoNLL 2003 Reuters dataset achieved an F-measure of 91 when it was tested on a similar Reuters corpus but only 64 on a Wall Street Journal dataset [CA05]. They were able to mitigate the domain-specificity to some extent (to F=69), but more work in this area is required to better adapt name taggers to new domains.

Chapter 4

Names, Mentions, and Entities

Information extraction gathers information about discrete *entities*, such as people, organizations, vehicles, books, cats, etc. The texts contain *mentions* of these entities. These mentions may take the form of

- names ("Sarkozy")
- noun phrases headed by common nouns ("the president")
- pronouns ("he")

In the previous chapter we considered methods for identifying names. Noun phrases and pronouns can be identified through syntactic analysis – either chunking or full parsing.

Information extraction finally seeks to build data bases of relations. Filling these data base entries with nouns or pronouns is not very helpful. We can't do much with an entry like

murders		
agent	victim	date
he	her	last week

At the very least, we want to fill in the names of the entities. But even names may be ambiguous; there are lots of people named "Michael Collins" or "Robert Miller". So we may want to create a data base of entities with unique ID's, and express relations and events in terms of these IDs.

The first step in either case is in-document coreference – linking all mentions in a document which refer to the same entity. If one of these mentions is a name, this allows us to use the name in the extracted relations. Coreference has been extensively studied independently of IE, typically by constructing statistical models of the likelihood that a pair of mentions are coreferential. We will not review these models here.

The performance of these models has been gradually improving over the last decade; accuracies are quite good for pronouns (70% to 80%) but not so good for common nouns. Because a large fraction of extracted relations involve nouns and pronouns, which can be mapped to names only if coreference is correctly analyzed, coreference continues to be a major limiting factor in relation and event extraction.

If we are aggregating extracted information across multiple documents, we will want to perform cross-document coreference to link together the entities mentioned by individual documents. This is generally limited to entities which have been mentioned by name in each document. This process will produce a data base of entities with IDs. As additional documents are processed, the entities mentioned will either be linked to existing entries in the data base or lead to the creation of new entries.

The study of cross-document coreference is relatively recent, and has been conducted mostly as part of IE evaluations: ACE 2008, WePS[1], and KBP. It involves modeling

- possible spelling and name variation, due for example to variant transliterations (Osama bin Laden / Usama bin Laden) or to nicknames (Bill Clinton vs. William Jefferson Clinton)
- likelihood of coreference, based on shared or conflicting attributes (extracted by IE from the individual documents) or on co-occurring terms in the documents

[1]Web People Search, http://nlp.uned.es/weps/

Chapter 5

Extracting Relations

5.1 Introduction

A *relation* is a predication about a pair of entities:

- Rodrigo works for UNED.
- Alfonso lives in Tarragona.
- Otto's father is Ferdinand.

Typically relations represent information which is permanent or of extended duration. However, the crucial characteristic is that they involve a pair of entities; this makes training somewhat simpler than for more general, n-ary predications such as the events discussed in the next chapter.

Relation detection and classification was introduced as a separate IE task in MUC-7 (1998). The MUC-7 relations task covered three relations involving organizations: location_of, employee_of, and product_of. Relations were extensively studied as part of the ACE evaluations, starting in 2002. The current KBP evaluations incorporate a form of relation extraction.

The ACE relation task ("Relation Detection and Characterization") was introduced in 2002 and repeatedly revised in order to create a set of relations which can be more consistently annotated. Most research has been done using the task definitions from 2003, 2004, and to a lesser extent, 2005. Each task defined a set of relation types and subtypes: for 2003: 5 types, 24 subtypes; for 2004: 7 types, 23 subtypes; and for 2005: 7 types, 19 subtypes. For example, the 2004 task had the following types and subtypes:

relation type	*subtypes*
physical	located, near, part-whole
personal-social	business, family, other
employment/membership/ subsidiary	employ-executive, employ-staff, employ-undetermined, member-of-group, partner, subsidiary, other
agent-artifact	user-or-owner, inventor-or-manufacturer, other
person-org affiliation	ethnic, ideology, other
GPE affiliation	citizen-or-resident, based-in, other
discourse	-

One crucial property of the ACE relation tasks was that mentions of both arguments had to appear explicitly in the sentence which expressed the relation. Thus a system would not be expected to recognize the father-son or mother-son relation in

Ferdinand and Isabella were married in 1469. A son was born the next year.

or an employ-executive relation in

There was a complete turnover in the Citibank executive suite. Fred Smith was named president.

As we shall see, most of the procedures for recognizing relations rely on this constraint.

In keeping with the distinction between entities and [entity] mentions, ACE makes a distinction between relations and relation mentions. A relation mention is a segment of text expressing a connection R between two entity mentions m_1 and m_2. If m_1 is a mention of entity e_1 and m_2 is a mention of entity e_2, then this is a mention of a relation R between e_1 and e_2. The hard part is finding the relation mentions in the text; given a relation mention and the coreference information (linking entity mentions to entities), it is simple to determine the corresponding relation.

If there are several mentions of an entity in a sentence, there is some ambiguity about specifying the relation mention. Consider

Fred died yesterday. His son, Eric, said the eulogy.

Here there are two entities, Fred (with mentions "Fred" and "his") and Eric (with mentions "His son" and "Eric"). In ACE, the relation mention will be recorded between the two closest mentions in the same sentence ... in this case, between "his" and "his son". If we want to report the corresponding relation in a table, we would find the entities, look for mentions of these entities which are names, and report those names ("Fred" and "Eric").

In the KBP evaluations, systems have to extract a set of attributes for a named person or organization. Many of these attributes have values which are themselves entities (employee_of, member_of, country_of_birth) so the task has strong parallels to earlier relation extraction tasks. Unlike ACE, however, there is no requirement that both entities be explicitly mentioned in the same sentence. Some cross-sentence analysis is required for roughly 15% of the attributes [JG11].

5.2 Hand-crafted patterns

Most instances of relations can be identified by the types of the entities involved and the words between the entities, so we can start building a relation identifier by listing word sequences:

- *person* lives in *location*
- *person* lived in *location*
- *person* resides in *location*
- *person* owns a home in *location*

(The words between the entities are not always sufficient: "Fred and Mary got married.")

The challenge we face is capturing the wide variety of natural language forms in which a given relation may be expressed, including morphological, syntactic, and lexical variations. We can get better coverage by introducing some syntactic generalization. For example, we can use base forms in a pattern and let them match any inflected form:

person <v base=reside> in *location*

If we have a chunker, we can specify a verb or noun group in the pattern:

person <vgroup base=reside> in *location*

This would match “resided”, “had resided”, “will reside”, etc. We can also allow for optional modifiers

> *person* <vgroup base=reside> [<pp>] in *location*

where “[<pp>]” designates an optional prepositional phrase.

Generalization can also be achieved by using paths in labeled dependency trees.

$$\begin{tikzcd}
& \mathit{resides} \arrow[dl, "\mathit{subject}"'] \arrow[dr, "\mathit{in}"] & \\
\mathit{person} & & \mathit{location}
\end{tikzcd}$$

In principle, this can handle any variety and number of modifiers on the arguments and the verb (“Fred Smith, who won the Nobel Prize, resided for the last five years in Baltimore.”).

5.3 Supervised Models

Instead of creating and refining these patterns by hand, we can build a relation extractor from an annotated corpus. We will need a corpus which has been annotated both for entity mentions and for relations.

We then want to convert relation tagging into a classification problem. To handle a single type of relation, we will train a classifier which classifies each pair of entity mentions appearing in the same sentence as either having or not having this relation. To handle n relation types, we can train an $n + 1$-way classifier. Alternatively, we can create two classifiers: a binary classifier which determines whether the entities bear *some* relation, and an n-way classifier (applied to instances which are passed by the first classifier) which determines which relation is involved.

To apply this to new data, we first run an entity tagger and then apply the relation classifier to every pair of entities.

A large number of features can and have been used in relation classifiers. These include

- the heads of the two entities
- the types of the two entities (person, organization, ...)
- the distance between the entities, in words (in the expectation that entities which are further away are less likely to be related)
- the sequence of words between the entities
- the individual words between the entities (each as a separate feature)
- the dependency path between the entities
- the individual words on the dependency path

[Kam04] gave the first report on the use of this feature-based approach to the ACE relation task. This paper used a maximum entropy model to combine a large number of features, including lexical features, syntactic features (taken from both a dependency tree and a parse tree), and entity type information. An even larger set of features, including word lists and an extensive set of word-level features, coupled with an SVM model, obtained several points improvement in F-measure [ZSZZ05]. A systematic study of the contribution of sequence, parse, and dependency features [JZ07] found that combining a set of basic features of each of the three types produced performance competitive with a richer feature set.

Kernel Methods If we step back a moment and consider what we are trying to do, when we are given an instance (a pair of entity mentions) to classify, we want to find an instance (or several instances) in the training corpus which are 'similar' and use the label assigned in the training set. For some information about the instance, such as the semantic type of the entities, a feature-based approach is natural. However, for some aspects of the instance, such as the word sequence or dependency path between the entities, or the structure of the parse tree subsuming the two entities, a feature-based approach is really awkward. We'd like to directly capture the similarity of two sequences or trees – to say, for example, that the sequences are the same except for a single word.

We can do so by defining a *kernel function* between pairs of objects. A kernel function can be thought of as a similarity measure between two objects. For example, a similarity measure between word sequences might count the number of insertions and deletions needed to transform the first sequence into the second sequence. SVMs and nearest-neighbor classifiers can use kernel functions directly. Both sequence kernels and dependency tree kernels have been used for relation extraction.

There has been some work on relation extraction at the token level, using sequence kernels [BM05], with moderate results. However, most of the work has been based on tree kernels. A variety of tree kernels have been explored, [ZAR03, ZZSZ06, ZZJZ07], leading to gradually improving performance, at present slightly better than the best feature-based systems. These kernels differ in several respects. Some initial work was done using shallow (partial) parse trees [ZAR03], but most recent work has been done on full parse trees or dependency trees.

A second variation concerns the span of the trees which should be compared. A common choice is the Shortest Path-enclosed Tree – basically the portion of the tree enclosed by the path which connects the two arguments to the relation [ZZSZ06]. This is comparable to classifying relations based on the words between the arguments. As we have noted above, this works much but not all of the time ("Fred and Mary were married."). Some improvement can be obtained by using a larger span in some cases [ZZJZ07]. A third variation concerns the kernel function itself. Some functions are relatively 'rigid' and will only match nodes at the same tree depth, while others allow a wider range of matches.

In addition to these variations, it is possible to create composite kernels which combine information from two basic kernels (e.g., sequence and tree information [ZZSZ06]) and can also incorporate feature-like information [ZG05].

5.4 Semi-Supervised Models

Preparing annotated training corpora for relation extraction is considerably more expensive than labeling named entities. Entity mentions must first be tagged (or corrected, if tagged automatically); then a number of different relations must be annotated. The case for reducing the annotation task through semi-supervised learning is therefore more compelling here than for names.

The basic approach – a form of co-training – is similar. Suppose we want to train a classifier for a relation R between names of type t_X and names of type t_Y. We will use two views:

- the two names
- the context of the two names

We begin with a seed – a few pairs of names bearing the specified relation. For example, if we are learning the book–author relation, [Moby Dick, Herman Melville] and [Animal Farm, George Orwell]. In its simplest form, the procedure would then collect contexts in which these names appear; for example "was written by" and ", the book by". Given these contexts, the procedure would then collect additional pairs, add these pairs to the seed, and repeat the process.

More generally, we want to train a classifier of contexts, based on positive and negative examples of the relation. If the relation R is a function, and [X, Y] is a seed, then [X, Y'], Y'≠Y can serve as a negative example. This classifier should produce a confidence measure (probability estimate) that the entities bear relation R. For example, if the classifier consists of a set of patterns P,

$$Conf(P) = \frac{P.positive}{(P.positive + P.negative)}$$

where P.positive is the number of positive matches for P and P.negative is the number of negative matches [AG00]. We train this classifier and apply it to the unlabeled examples. Next, we compute a confidence measure for each pair of names [X, Y] in the corpus and add the most confident pairs to the seed.

The confidence measure for a pair depends on the model we use for combining the evidence from individual contexts; we can take the confidence of the most confident context or, following [AG00], we can treat the contexts as independent:

$$Conf(X, Y) = 1 - \prod_{P \in contexts\_of\_X,Y} (1 - Conf(P))$$

Some problems There are several potential problems with this general approach. One is the need for negative examples in order to be able to assess patterns. If R is not a functional relation, these may be difficult to generate; even for functional relations, the result will be noisy.

The approach is based on the assumption that there is a unique relation associated with a pair of entities. If many pairs associated with relation R are also instances of relation R', we will see a gradual expansion of the seed set to include both R and R' – an example of semantic drift.

Furthermore, there is no natural stopping point for the process. In the semi-supervised training of NERs, it was possible to identify a set of name classes which covered almost all instances of names. It is much harder to imagine a complete set of relations.

Some recent work [SG10] indicates that these two problems (semantic drift and stopping) can be addressed by clustering the context patterns, identifying the cluster containing the seed, and terminating the bootstrapping if a substantial number of the patterns being added fall outside this cluster.

Some papers The DIPRE system [Bri98] was one of the first systems for semi-supervised relation extraction, and one of the first designed to operate on the Web. The context patterns were based on character sequences before, between, and after the entities, and so could make use of both lexical contexts and XML mark-up contexts. The patterns were associated with particular web sites. There was no ranking of patterns or entity pairs; instead, some heuristics were used to insure that the patterns were sufficiently specific. The entire procedure was demonstrated on the book–author relation.

The Snowball system [AG00] introduced several improvements, including the use of name types, the ranking of patterns and entity pairs (as described above, using negative examples) and more relaxed patten matching. It was applied to the company–headquarters relation, a functional relation.

5.5 Distant Supervision

In the semi-supervised relation learner just described, each iteration includes steps where we take a set of pairs, retrieve sentences involving those pairs, and then use these sentences to train an improved classifier. This is done repeatedly, starting from a small seed and gradually building up a large inventory of pairs.

Suppose we could start out with a large data base of instances of the relation of interest, along with a large text corpus containing text segments corresponding to many of the instances in the data base. We can use this combination to create a set of positive and negative instances for training a relation classifier. This approach has been described as 'learning from weakly labeled data' [CK99] and more recently dubbed 'distant supervision' [MBSJ09]: the data base provides supervision, but not by direct annotation of the text data.

In training the model, one takes an instance [X, Y] of the relation in the data base, collects all sentences in the corpus containing both names X and Y, and generates a set of features for each text instance. These become the positive training instances. A sampling of pairs [X, Y] not appearing in the data base serve as negative examples. The model is applied just like a supervised model, to each entity pair in a sentence.

Alternatively, the set of sentences containing X and Y can be treated as a single positive example (with the union of the features from the separate sentences). In applying the model to a new pair [X, Y] one must then collect all instances of X, Y across the corpus as an input to the classifier. This provides a somewhat different evaluation: assessing whether a relation is implied by a corpus rather than by a particular sentence.

Some problems Distant supervision can produce a large amount of training data but the data may be quite noisy. One potential problem (as for semi-supervised methods) is that a pair of entities may be connected by multiple relations, so that a sentence retrieved as an instance of a relation R may instead express a relation R'. For example, many of the sentences retrieved based on the relation *governor_of* may in fact express the relation *resides_in* or *born_in*. These are *false positives* for training a *governor_of* classifier. This effect will be reduced (but not eliminated) if the data base was built *from* the associated text – for example, Wikipedia entries and their associated infoboxes.

A different problem arises if the data base relation is incomplete – for example, if a *resides_in* relation only contains a subset of the locations where a person has lived. Since entity pairs not in the data base are treated as negative examples, this will lead to *false negatives*.

Both of these issues are affected by the set of relations being trained, the completeness of the data base, and the connection between the texts and the data base. The effects of these properties on distant supervision need to be further explored.

Some papers An early effort of this type [CK99] focused on one relation in the biology literature, *subcellular localization*. They used a database (the Yeast Protein Database) in which some of the entries contained links to the articles which reported a particular relation. Using a symbolic learner, they were able to train a classifier for extracting this relation.

The experiments reported in [MBSJ09] used Freebase as the source of relations (the 102 largest relations in Freebase were used), while Wikipedia was used as the text source. A combination of token-based and dependency features are used, aiming for high precision. The set of sentences involving an X, Y pair were treated as a single training/test instance. Their primary evaluation took the most-confidently-classified 100 or 1000 instances of a set of relations and evaluated their precision by hand; precisions of 67% to 69% were reported.

A more standard evaluation of distant supervision was reported in [NM11]. They annotated 200 documents with a set of 52 relations and measured system recall and precision; overall recall was 63%; overall precision 91%.

5.6 Evaluation

Evaluation of relation extraction may seem relatively straightforward: we prepare a test corpus annotated with entities and relations; the relation extractor is run on this text; we count correct, missing, and spurious relations (as we did for name tags) and then compute recall, precision, and F measure. Understandably the scores will depend on the corpus and the set of relation types, but there are many other factors:

- perfect mentions vs. system mentions: The relation classifier operates on a set of entity mentions. Was it run on the hand-identified entities of the annotated test corpus, or on a set of automatically generated entities? The latter will have some errors and reduce relation extraction performance, often substantially. For this reason most papers report results on perfect mentions; a few also report scores on system mentions. [With system mentions there is also the subtle problem of entity mention alignment: if entity mentions differ slightly from those in the key, when do you count them as 'the same' so you can get credit for a correct relation?]

- type vs. subtype: in matching relations, do you require a match just on the type or also on the subtype? Most papers report scores based just on type. Some subtype distinctions (e.g., deciding between employees and group members) can be quite difficult.

- between names vs between all mentions: the semi-supervised (bootstrapping) methods learn patterns which connect names, while the ACE data generally used for evaluation includes relation mentions connecting all types of entity mentions. Some adjustment of the evaluation is required.

In assessing the performance of a relation tagger, the quality of the test corpus is of course important. High quality annotation is considerably harder than for name tagging, especially if there are a large number of relations, as in ACE. In particular, there is a tendency for annotators to miss a substantial number of relations. Multiple annotation and adjudication is essential for a high-quality corpus.

The best current systems get about F=75 on ACE data, using reference entities and only classifying to the relation type level (not to subtypes). Performance is much lower (roughly F=40) using system-produced entities.

Chapter 6

Extracting Events and Scenarios

In the previous chapter we examined the task of relation extraction – extracting predications involving two arguments. In this chapter we consider the more general problem of predications involving arbitrary numbers of arguments.[1]

There is a wide variation in the complexity of the information structures we can try to extract from texts. At one end of the spectrum are elementary events which may take 2 or 3 primary arguments plus optional time and place modifiers. The ACE 2005 event extraction task included 33 such event types, listed in the table just below. The goal in selecting these types was to identify types of events which occurred frequently over a wide range of news stories. No attempt was made to capture the relations between events.

Event type	Subtypes
Life	Be-born, Marry, Divorce, Injure, Die
Movement	Transport
Transaction	Transfer-ownership, Transfer-money
Business	Start-org, Merge-org, Declare-bankruptcy, End-org
Conflict	Attack, Demonstrate
Personnel	Start-position, End-position, Nominate, Elect
Justice	Arrest-jail, Release-parole, Trial-hearing Charge-indict, Sue, Convict, Sentence, Fine, Execute, Extradite, Acquit, Appeal, Pardon

In analogy with relation extraction, the ACE event task had the notions of event mentions and events. Each event mention had a *trigger*, the main word expressing the event.

If we focus our attention on documents about one particular topic, we can observe greater regularities and build larger structures to capture the connections. For example, if we look at reports of terrorist events, they will include the attack itself, the effect on people (killed, wounded) and the effect on facilities (damaged, destroyed). These will be implictly connected by a causal relation. The MUC evaluations each involved collections of documents on a single topic, and constructed such larger structures, called *scenario templates.* Examples of such topics included terrorist incidents, international joint ventures, management succession, and plane crashes. A sample template for MUC-3 (terrorist incidents) is shown below.

```
0.  MESSAGE ID                          TST1-MUC3-0099
1.  TEMPLATE ID                         1
```

[1]A specialized form of event extraction involves documents that are known to describe a single event of a known type. For example, seminar announcements (almost always) describe a single seminar. This simplifies the task since we know that all the arguments we extract (seminar speaker, time, location, title,) will refer to the same event. Such a task is sometimes referred to as Implicit Relation Extraction (IRE). IRE can be treated as a sequential modeling task, like named entity tagging, although there may be an added constraint on the number of times an argument can appear. We do not further consider IRE tasks here.

```
2.  DATE OF INCIDENT                   24 OCT 89 - 25 OCT 89
3.  TYPE OF INCIDENT                   BOMBING
4.  CATEGORY OF INCIDENT               TERRORIST ACT
5.  PERPETRATOR: ID OF INDIV(S)        "THE MAOIST SHINING PATH GROUP"
                                       "THE GUEVARIST TUPAC AMARU REVOLUTIONARY
MOVEMENT ( MRTA ) GROUP"
                                       "TERRORISTS"
6.  PERPETRATOR: ID OF ORG(S)          "SHINING PATH"
                                       "TUPAC AMARU REVOLUTIONARY MOVEMENT ( MRT
A )"
                                       "THE SHINING PATH"
7.  PERPETRATOR: CONFIDENCE            POSSIBLE: "SHINING PATH"
                                       POSSIBLE: "TUPAC AMARU REVOLUTIONARY MOVE
MENT ( MRTA )"
                                       POSSIBLE: "THE SHINING PATH"
8.  PHYSICAL TARGET: ID(S)             "THE EMBASSIES OF THE PRC AND THE SOVIET
UNION"
9.  PHYSICAL TARGET: TOTAL NUM         1
10. PHYSICAL TARGET: TYPE(S)           DIPLOMAT OFFICE OR RESIDENCE: "THE EMBASS
IES OF THE PRC AND THE SOVIET UNION"
11. HUMAN TARGET: ID(S)                -
12. HUMAN TARGET: TOTAL NUM            -
13. HUMAN TARGET: TYPE(S)              -
14. TARGET: FOREIGN NATION(S)          PRC: "THE EMBASSIES OF THE PRC AND THE SO
VIET UNION"
15. INSTRUMENT: TYPE(S)                *
16. LOCATION OF INCIDENT               PERU: SAN ISIDRO (TOWN): LIMA (DISTRICT)
17. EFFECT ON PHYSICAL TARGET(S)       -
18. EFFECT ON HUMAN TARGET(S)          -
```

6.1 Hand-crafted Patterns

Event extraction can be considered a combination of two problems: identifying the phrases which fill individual slots (slot filling) and then figuring out which slots belong to the same event or template (consolidation).

Slot Filling Events can be captured through patterns, much like relations, except that a variable number of slots may be involved. For each slot, the semantic type constraint on the slot is specified. For example

$company_1$ appointed $person_2$ as $title_3$

Associated with this pattern would be a template such as

event type: start_job		
person	position	company
2	3	1

The considerations in writing patterns for event slot extraction are similar to those for writing relation patterns: being able to capture syntactic and semantic paraphrase. Syntactic paraphrase can be addressed in part through chunking and allowance for certain classes of optional modifiers, as in the FASTUS system [AHB+93]. It can also be achieved through the use of dependency parses or more regularized structures.

Consolidation Hand-crafted patterns will typically be accompanied by task-specific rules for consolidation. The amount of consolidation required depends on the granularity of the rules: if individual patterns capture several slots, as in the example above, little consolidation may be required; if individual patterns capture but a single slot, consolidation will play a larger role.

6.2 Supervised Extraction

For the ACE event task, the general approach has been to train multiple classifiers, first to identify the trigger and then to identify the arguments. [Ahn06, CJ09]. Trigger classification involves deciding whether the word is a trigger and, if so, its event type. This can be done in a single stage or two successive classifiers. If a trigger has been identified, an argument classifier is then applied to a pair consisting of the trigger and an entity in the same sentence. Again, this can be done in a single stage or a binary classifier (argument / non-argument) can be followed by a classifier for role assignment.

Enhancements: using non-local information The performance of the extraction engine is limited by the amount of training data, but also by the fact that the classifiers (in particular, the trigger classifiers) rely on local clues ... the trigger word itself, the syntactic dependents of the trigger, possibly other entities in the sentence. In some cases this is not enough information to determine the type of event. For example,

> He left Microsoft that afternoon.

could mean that he retired or just went on a trip;

> His attack that afternoon was successful.

could refer to a military manuever or a tennis game.

Two general strategies have been tried to improve extraction performance. One strategy looks for indicators of a relevant event in a wider context (the entire sentence or the document). The other looks for other reports of the same event which might be less ambiguous.

Patwardhan and Riloff [PR09] combined sentence-level and local evidence and demonstrated some improvement on MUC04 (terrorist incident) extraction. Liao showed how document-level topic information could be used to improve ACE event identification, either by making use of statistics about event co-occurrence within a document [LG10b], or by building an LDA topic model of a corpus [LG11].

Ji [JG08] relied on a cross-document approach combining information from multiple documents on the same topic or event. Given a document to be analyzed, and a large collection of unannotated documents, the procedure builds a cluster of documents similar to the original document, analyzes each document in the cluster, and uses consistency constraints over the extracted information to infer additional information about the original document.

Consolidation For the ACE event task, where events are atomic (not composed into larger structures) consolidation is a matter of event coreference: determining when two event mentions refer to the same event. Taking an approach similar to that for entity coreference, one can train a classifier to decide whether two event mentions are coreferential, based on the event type, triggers, distance, argument overlap or conflict, and attributes of the two events. Both Ahn [Ahn06] and the CUNY group [CJH09] found that such an approach is quite effective at making coreference decisions when applied to the event mentions in the reference (hand-annotated) corpus. However, they both found that, at the current state-of-the-art, system generated event mentions are so noisy as to render the same coreference procedures almost useless. It's not clear if it would be possible to create a procedure more robust to such noise.

The problem of assembling templates for larger scenarios, which may involve different event types, has received less attention. In cases where the information for a single template (mostly) appears consecutively in a text, this can be reduced to a text segmentation problem, where all the individual events or slot fills within a single segment go into a single template. Feng *et al.* has shown (for some biomedical texts) how this could be implemented as a sequential modeling problem at the level of sentences, labeling each sentence as B (beginning of segment), I (continuation of segment), or O (not part of a segment) [FBH07].

6.3 Semi-supervised Models

The relations in ACE relation extraction are very general and appear across a wide range of documents. On the other hand, many of the events types involved in event extraction and scenario template extraction are associated with specific topics. This provides an opportunity for semi-supervised training.

The observation was first made by Riloff [Ril96] in connection with the MUC-3 task (terrorist incidents) that if one takes a collection of relevant documents and a collection of irrelevant documents (those not about terrorism) and computes the frequency of particular predicates in the two collections, those predicates which appear with a relatively higher frequency in the relevant documents are (in general) predicates relevant to the task. Using the scoring function

$$score(P) = (RelFreq/Freq) \times \log(Freq)$$

(where $Freq$ is the freqency of the pattern and $RelFreq$ is the frequency in relevant documents) applied to about 750 terrorism-relevant and 750 irrelevant articles, the top-ranked patterns were

(subj) exploded
murder of (np)
assassination of (np)
(subj) was killed
(subj) was kidnapped

Yangarber [YGTH00] extended this to a semi-supervised (bootstrapping) procedure: a seed set of patterns were used to identify some relevant documents; these documents were used to identify a few additional patterns, etc. The initial procedure, which was applied to the MUC-6 (management succession) task, did not have any natural stopping point. By training competing learners concurrently for different event types, it was possible to improve the learners for individual topics and reach a natural stopping point where all documents were assigned to a topic [Yan03].

Stevenson and Greenwood [SG05] demonstrated an alternate approach to generalizing event extraction patterns using WordNet, again applied to the MUC-6 task. Liao [LG10a] combined these two approaches, using WordNet to expand patterns but filtering them using document relevance criteria, showing gains both on MUC-6 and three of the ACE event types.

6.4 Evaluation

As with any IE task, evaluating event or scenario templates begins with preparation of an annotated reference corpus. An evaluation is only as good as its reference corpus, and the quality of the reference corpus depends in turn on the precision with which the events are defined. In some technical domains, events and their arguments can be precisely defined. This is also true for some everyday events – being born, married, divorced, dying, starting a job, being convicted of a crime. On the other hand, some ACE event classes, such as *attack* and *meet*, have fuzzy boundaries, so the inter-annotator agreement is quite low.

Devising an intuitive and effective metric for comparing a system output with the reference annotation is difficult. The system output and reference consist of multiple events / templates; each event will have multiple arguments, and each argument will have a role. There may be missing or spurious events, missing or spurious arguments, and correct arguments with incorrect roles.

For both MUC and ACE there was a need to have a single event/template score. This was obtained, roughly speaking, by searching for the best possible alignment of templates and template slots (MUC) or events and event arguments (ACE). ACE provided a different penalty for each type of error (wrong event type, missing argument, spurious argument, incorrect role, ...). The resulting scores are not always easy to understand, especially when small changes in output alter the optimal alignment.

The alternative is to have two (or more) separate evaluation metrics. For MUC, a fair amount of the research focused on slot filling and did not address the consolidation task. Accordingly, the evaluation metric counted the [slot_name, slot_value] pairs, classified them as correct, missing, and spurious, and computed recall and precision in the usual way. For ACE, separate measurements can be made of event classification and argument extraction. Event classification is judged based on finding correct [trigger word, event type] pairs. Argument extraction (similar to MUC slot filling) is based on finding correct [event type, role, event argument] triples. In judging correct event arguments, we either use reference entities or align the system entities with reference entities.

Chapter 7

Unsupervised Information Extraction

In all the methods we have discussed up to now, we have assumed that we knew in advance what semantic classes of entities, relations, or events we wanted to extract. This information was either manually incorporated into rules or expressed through supervision – either fully supervised or semi-supervised learning.

In this chapter we shall consider what we can learn from the texts themselves, without any supervision. Given some texts and some entity types, can we identify the relations, or at least the most common relations, between these types of entities? Going a step further, can we identify both a set of relations and a set of types of entities (word classes) which appear as the arguments of these relations?

If we consider the general language (of news, magazines, literature) the word classes will be correspondingly general (people, animates, places, ...) and have fuzzy boundaries (because of metaphorical usages, for example). However, if we narrow the domain and in particular focus on scientific and technical domains, the word classes will be more specialized and the constraints sharper. Harris [Har68] dubbed these *science sublanguages* and observed that just as the general language has syntactic word class constraints

> The cat is sleeping.
> * The cat are sleeping.

(where the second sentence is ungrammatical in the general language) the sublanguage has sublanguage word constraints

> The potassium entered the cell.
> * The cell entered the potassium.

(so the second sentence is ungrammatical in the sublanguage). Ideally, unsupervised IE should be able to discover such argument constraints and thus discover the information structures of the domain.

7.1 Discovering Relations

Most of the work has focussed on discovering relations. The basic idea is fairly simple: we gather a large set of triples (potential relation instances), each consisting of a pair of arguments (entities) and a context – either the word sequence between the arguments or the dependency path connecting the arguments. The arguments must appear in the same sentence, at most a fixed distance apart; typically only examples with named arguments are collected. Our goal is to group these triples into sets which represent the same semantic relation.

To this end, we define a similarity measure between triples, based on whether the arguments match and on how similar the contexts are. Linking triples which involve the same pair of arguments captures the notion that if the same pair of names appear together in two different contexts they are likely to represent the same relation (this same notion was the basis for semi-supervised relation extraction). Using the similarity measure, we gather the triples into clusters, using some standard clustering procedure. If the procedure has been successful, the triples which fall into the same cluster will convey (approximately) the same relation.

If the procedure is provided with a standard set of entity types and an entity tagger, we will treat each pair of argument types separately. Using the standard MUC types, for example, we would gather PERSON-PERSON relations, PERSON-ORGANIZATION relations, etc. separately.

Hasegawa et al. [HSG04] report an early experiment of this type. They collected name pairs (PERSON-GPE and COMPANY-COMPANY) from one year of newspaper articles using rather stringent criteria: the names could be at most 5 words apart, and each pair had to appear at least 30 times. This yielded a small number of pairs with extensive evidence about each pair, and made possible a relatively simple clustering strategy. An initial cluster was built by taking all the contexts for a particular name pair; this assumes that each pair is only connected by a single relation. Then clusters with any context words in common were merged. This produced a small number of high-quality clusters, such as ones representing merger and parent relations between companies. Chen et al. [CJTN05] refined several aspects of this approach, automatically selecting weights for words in the context and choosing the number of clusters to generate.

Zhang et al. [ZSW$^+$05] observed for the same news corpus that 10 to 25% of the name pairs (depending on the frequency of the pairs) were connected by more than one relation. For example, a story about A acquring B might be followed by a story in which A was the corporate parent of B. Accordingly they clustered instances of name pairs separately, allowing distinct instances of the same pair to appear in separate clusters. By doing so, and by using a parse-tree-based measure of context similarities (rather than a simple bag of words) they obtained some performance improvements.

Rosenfeld and Feldman [RF07], also using a corpus of news stories, demonstrated that this clustering could be performed without classifying types of names. This should make it easier to port the procedure to new domains which may have new entity types. Using a rich set of features based on matching parts of the context token sequence, they compared the effectiveness of different clustering procedures.

The task becomes somewhat more challenging when we move from single news sources (used in all the experiments just described) to the Web. The scale is larger, and there is a greater variety in entity names and relation contexts. The larger scale requires more efficient clustering algorithms. Typical simple clustering algorithms, such as hierarchical agglomerative clustering, require time $O(n^2)$ to cluster n items, since each item must be compared (at least once) to every other item. This is manageable for small collections – thousands of name pairs – but not for the Web (millions or billions of pairs. However, if the connections (shared features) between items are sparse (argument pairs share only a few relation phrases, for example) and the final clusters will be relatively small (compared to the total number of clusters), it is possible to build a more efficient procedure by iterating over the shared features.

Yates and Etzioni [YE07] described such a procedure, which takes time $O(kn \log n)$ where k is the maximum cluster size. They applied it to a set of 10,000 frequently appearing relation phrases collected from the Web. To address the problem of name variants in Web data, they also clustered argument strings based on string similarity and shared contexts, thus identifying many synonymous names. The two clustering phases are mutually beneficial: identifying synonymous arguments means that we have more shared contexts with which to identify synonymous relation phrases, and vice versa.

7.2 Discovering Relations and Entity Classes

As we have noted, new domains are likely to have new entity types as well as new relation types. It is possible to consider entity type and relation type discovery as separate problems, but there are natural benefits to linking these tasks: if one can generalize from the observed arguments of a relation context to an argument class, that may help in clustering relation contexts.

Kok and Domingos [KD08] implemented such a joint clustering procedure using Markov Logic Networks for about 10,000 relation phrases taken from the Web, and demonstrated substantial gains over procedures which cluster the arguments and relation phrases separately. A very different type of model, based on Latent Dirichlet Allocation [LDA – a technique typically used for document topic models], has also been used to create combined argument and relation phrase clusters [YHRM11] from news sources.

7.3 Discovering Larger Semantic Structures: Events

Just as the supervised training of event extractors is more difficult than relation extractors, the unsupervised discovery of multi-argument structures such as event templates is more challenging than relation discovery. Not every description of an event will provide information on all the event arguments, and the arguments which are present may be scattered over several sentences, so we may need to build a separate set of contexts for each argument.

One property of news stories that we can take advantage off is that each article, or at least the first few sentences of each article, generally describe a single event, so the names in (the beginning of) an article correspond to (a subset of) the event arguments.

Shinyama and Sekine [SS06] addressed the problem of missing arguments in a single article by taking 12 parallel feeds from different news sources, using bag of words overlap (including names) to identify stories about the same event, and putting these articles into a cluster. Names which appeared early and often in multiple articles were likely to be the primary arguments of the event. Then they built *metaclusters* out of clusters representing the same *type* of event. Two event clusters were placed in a metacluster based on finding corresponding names in the two clusters with common contexts; these then define the arguments and extraction patterns for this type of event.

7.4 Evaluation

Evaluating unsupervised extraction is inherently problematic. When we developed hand-coded, supervised, or semi-supervised extraction systems, we declared what entity and relation classes we wanted to extract; if something different was extracted, that counted as an error. When we perform unsupervised extraction, we in effect ask the data to tell us what the good classes are. If we independently (through manual data analysis) come up with a different set of classes for the same data, it may not be easy to say who is right, or whether there are multiple correct analyses.

Nonetheless, an approach based on a separate gold standard can be useful. [CJTN05], for example, tested on an ACE corpus and evaluated by aligning the resulting classes against the ACE relation types. [RF07] and [KD08] created their own keys by hand from subsets of their corpora.

More indirect evaluations are also possible. To evaluate its argument clusters, [KD08] measured how well they aligned with WordNet synsets. [YHRM11] used the relation phrase clusters as features for a relation classifier trained using distant supervision, and reported some performance improvements.

Chapter 8

Other Domains

What sort of texts are good candidates for information extraction? Basically, domains in which there are large volumes of text which express a common set of semantic relations, and where there is a strong incentive for being able to search, build data bases, collect statistics, or otherwise mine this information.

News. The examples we have used so far involve general news, including political, international, and business news. The Web has made such news available from thousands of sources in large volume, and the ability to search or react rapidly to such information is of interest to many large businesses and governments.

Quite a number of such systems have been deployed. For example, the Europe Media Monitor's NewsExplorer[1] gathers news from across Europe, clusters related news stories, and extracts names, locations, general person-person relations, and event types. OpenCalais from Thomson Reuters[2] extracts a range of entity, relation, and event types from general and business news. The GATE system from the University of Sheffield has been used in a number of business intelligence applications [MYKK05]. For the most part these deployed applications have used hand-crafted lists of terms and regular expressions, rather than corpus-trained approaches.

Two other domains which meet the criteria for IE are medical records and the scientific literature.

Medical records have been a target area for information extraction for several decades [SFLmotLSP87]. Hospitals produce very large amounts of patient data, and a significant portion of this is in text form. IE could improve access to crucial medical information in time-critical situations. Furthermore, medical research and monitoring of overall patient care require analysis of this data, in order to observe relationships between diagnosis and treatment, or between treatment and outcome, and this in turn has required a manual review of this textual data, and in many cases the manual assignment of standardized diagnosis and treatment codes. Automatically transforming this text into standardized fields and categories based on medical criteria can greatly reduce the manual effort required. The push for electronic health records (EHR) has increased both the need for and the potential impact of medical text analysis.

A number of implemented systems have already demonstrated the feasibility of such applications for specialized medical reports [MSKSH08, FSLH04]. However, progress in clinical text analysis has been slower than for other IE tasks [CNH^{+}11], for a number of reasons. Medical records are sensitive and have to be carefully anonymized; this has made it difficult to obtain large amounts of data or to share data between sites. Few standard test sets are available for clinical data. More generally, until recently much of the work on EHR has been done locally by individual medical centers, leading to a lack of standardization of EHR.

[1]http://emm.newsexplorer.eu
[2]http://www.opencalais.com

Only in the past few years have shared evaluations of IE for clinical data developed along the lines of MUC and ACE. Several of these Challenges in NLP for Clinical Data have been organized in connection with Informatics for Integrating Biology and the Bedside.[3] The specific tasks are quite different from year to year. For example, the 2009 task involved the extraction of information on medication; the 2010 task was quite general, involving the extraction of problems, tests, and treatments from discharge summaries (the extended reports prepared at the end of a patient's hospital stay).

Biomedical literature. In our introduction, we noted Zellig Harris's vision for using a process similar to fine-grained information extraction to index scientific journal articles [Har58]. As robust extraction technology has caught up in the last few years with this vision, there has been renewed interest in extracting information from the scientific literature. One particular area has been biomedicine and genomics, where the very rapid growth of the field has overwhelmed the researcher seeking to keep current with the literature. The goal for NLP has been to automatically identify the basic entities (genes and proteins) and reports of their interaction, and build a data base to index the literature. To address this goal, a number of annotated corpora have been developed.[4] These have been used in turn for open, multi-site evaluations of biomedical named entity and relation extraction.

[3]See www.i2b2.org/NLP

[4]See for example the resources of the GENIA project, http://www-tsujii.is.s.u-tokyo.ac.jp/ genia/topics/Corpus/

Bibliography

[Abn08] Steven Abney. *Semisupervised Learning for Computational Linguists.* Chapman and Hall, 2008.

[AG00] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, New York, NY, USA, 2000. ACM.

[AHB$^+$93] Douglas Appelt, Jerry Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI-93*, pages 1172–1178, Chambery, France, August 1993.

[Ahn06] David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia, July 2006. Association for Computational Linguistics.

[BHAG05] Markus Becker, Ben Hachey, Beatrice Alex, and Claire Grover. Optimising selective sampling for bootstrapping named entity recognition. In *Proceedings of the ICML-2005 Workshop on Learning with Multiple Views*, 2005.

[BM05] Razval Bunescu and Raymond Mooney. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, Vancouver, BC, December 2005.

[BMSW97] Daniel Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C., 1997.

[Bor99] Andrew Borthwick. *A Maximum Entropy Approach to Named Entity Recognition.* PhD thesis, Dept. of Computer Science, New York University, 1999.

[Bri98] Sergey Brin. Extracting patterns and relations from the world-wide web. In *Proceedings of the 1998 International Workshop on the Web and Databases at the 6th International Conference on Extending Database Technology, EDBT 98*, pages 172–183, 1998.

[BSAG98] Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, Canada, 1998.

[CA05] Massimiliano Ciaramita and Yasemin Altun. Named-entity recognition in novel domains with external lexical knowledge. In *Advances in Structured Learning for Text and Speech Processing Workshop*, 2005.

[CJ09] Zheng Chen and Heng Ji. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[CJH09] Zheng Chen, Heng Ji, and Robert Haralick. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*, pages 17–22, Borovets, Bulgaria, September 2009. Association for Computational Linguistics.

[CJTN05] Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. Unsupervised feature selection for relation extraction. In *Second International Joint Conference on Natural Language Processing – Companion Volume*, pages 262–267, Jeju Island, Republic of Korea, October 2005.

[CK99] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intel ligent Systems for Molecular Biology (ISMB-99*, pages 77–86. AAAI Press, 1999.

[CNH$^+$11] Wendy Chapman, Prakash Nadkarni, Lynette Hirschman, Leonard D'Avolio, Guergana Savova, and Ozlen Uzuner. Overcoming barriers to nlp for clinical text: the role of shared tasks and the need for additional creative solutions. *Journal of the American Medical Informatics Association*, 18:540–543, 2011.

[CS99] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[DeJ82] Gerald DeJong. An overview of the FRUMP system. In Wendy Lehnert and Martin Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum, Hillsdale, NJ, 1982.

[FBH07] Donghui Feng, Gully Burns, and Eduard Hovy. Extracting data records from unstructured biomedical full text. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 837–846, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[FSLH04] C. Friedman, L. Shagina, Y. Lussier, and G. Hripcsak. Automated encoding of clinical documents based on natural language processing. *Journal of the American Medical Informatics Association*, 11:392–402, 2004.

[GS96] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th International Conference on Computational linguistics*, pages 466–471, Copenhagen, Denmark, 1996.

[Har58] Zellig Harris. Linguistic transformations for information retrieval. In *Proceedings of the International Conference on Scientific Information*, Washington, D.C., 1958. National Academy of Sciences-National Research Council.

[Har68] Zellig Harris. *Mathematical Structures of Language.* Interscience, 1968.

[HSG04] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 415–422, Barcelona, Spain, July 2004.

[JG08] Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[JG11] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[JZ07] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, Rochester, New York, April 2007. Association for Computational Linguistics.

[Kam04] Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 178–181, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[KD08] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the Nineteenth European Conference on Machine Learning*, pages 624–639, Antwerp, Belgium, 2008. Springer.

[LG10a] Shasha Liao and Ralph Grishman. Filtered ranking for bootstrapping in event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 680–688, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[LG10b] Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[LG11] Shasha Liao and Ralph Grishman. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, Hissar, Bulgaria, September 2011.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[MC08] Tara McIntosh and James R. Curran. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Workshop*, Hobart, Australia, 2008.

[McI10] Tara McIntosh. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365, Cambridge, MA, October 2010. Association for Computational Linguistics.

[MGM98] Andrei Mikheev, Claire Grover, and Marc Moens. Description of the LTG system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[MSKSH08] S.M. Meystre, G.K. Savova, K.C. Kipper-Schuler, and J.F. Hurdle. Extracting information from textual documents in the electronic health record: a review of recent research. In *Yearbook of Medical Informatics*, pages 128–144. Schattauer, Stuttgart, 2008.

[MYKK05] Diana Maynard, Milena Yankova, Alexandros Kourakis, and Antonis Kokossis. Ontology-based information extraction for market monitoring and technology watch. In *Proceedings of the Workshop on End User Aspects of the Semantic Web, 2nd European Semantic Web Conference*, Heraklion, Crete, 2005.

[NM11] Truc Vien T. Nguyen and Alessandro Moschitti. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 277–282, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[NS07] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[PR09] Siddharth Patwardhan and Ellen Riloff. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore, August 2009. Association for Computational Linguistics.

[RF07] Benjamin Rosenfeld and Ronen Feldman. Clustering for unsupervised relation identification. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, pages 411–418, 2007.

[Ril96] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, 1996.

[RR09] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[SDW01] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Mining the web with active hidden markov models. In *Data Mining, IEEE International Conference on*, pages 645–646, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[SFLmotLSP87] Naomi Sager, Carol Friedman, Margaret Lyman, and members of the Linguistic String Project. *Medical Language Processing: Computer Management of Narrative Data.* Addison-Wesley Pub. Co., Reading, Mass., 1987.

[SG05] Mark Stevenson and Mark A. Greenwood. A semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, MI, June 2005. Association for Computational Linguistics.

[SG10] Ang Sun and Ralph Grishman. Semi-supervised semantic pattern discovery with guidance from unsupervised pattern clusters. In *Coling 2010: Posters*, pages 1194–1202, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[SN04] Satoshi Sekine and Chikashi Nobata. Definition, dictionary and tagger for extended named entities. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004.

[SS06] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June 2006. Association for Computational Linguistics.

[SZS$^+$04] Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 589–596, Barcelona, Spain, July 2004.

[TKS03] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Sixth Conference on Natural Language Learning 2002*, pages 142–147, 2003.

[TKSDM03] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[Yan03] Roman Yangarber. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[YE07] Alexander Yates and Oren Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of HLT-NAACL*, 2007.

[YGTH00] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Saarbrcken, Germany, August 2000.

[YHRM11] Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[YLG02] Roman Yangarber, Winston Lin, and Ralph Grishman. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, 2002.

[ZAR03] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Machine Learning Research*, 3:1083–1106, 2003.

[ZG05] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 419–426, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[ZSW$^+$05] Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Second International Joint Conference on Natural Language Processing*, pages 378–389, Jeju Island, Republic of Korea, October 2005.

[ZSZZ05] GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[ZZJZ07] GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736, 2007.

[ZZSZ06] Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832, Sydney, Australia, July 2006. Association for Computational Linguistics.