VI

Information Extraction

VI.1 INTRODUCTION TO INFORMATION EXTRACTION

A mature IE technology would allow rapid creation of extraction systems for new tasks whose performance would approach a human level. Nevertheless, even systems without near perfect recall and precision can be of real value. In such cases, the results of the IE system would need to be fed into an auditing environment to allow auditors to fix the system's precision (an easy task) and recall (much harder) errors. These types of systems would also be of value in cases in which the information is too vast for the users to be able to read all of it; hence, even a partially correct IE system would be preferable to the alternative of not obtaining any potentially relevant information. In general, IE systems are useful if the following conditions are met:

- The information to be extracted is specified explicitly and no further inference is needed.
- A small number of templates are sufficient to summarize the relevant parts of the document.
- The needed information is expressed relatively locally in the text (check Bagga and Biermann 2000).

As a first step in tagging documents for text mining systems, each document is processed to find (i.e., extract) *entities* and *relationships* that are likely to be meaningful and content-bearing. The term *relationships* here denotes *facts* or *event*s involving certain *entities*.

By way of example, a possible *event* might be a company's entering into a joint venture to develop a new drug. An example of a *fact* would be the knowledge that a gene causes a certain disease. *Facts* are static and usually do not change; events are more dynamic and generally have a specific time stamp associated with them. The extracted information provides more concise and precise data for the mining process than the more naive word-based approaches such as those used for text categorization, and the information tends to represent concepts and relationships that are more meaningful and relate directly to the examined document's domain.

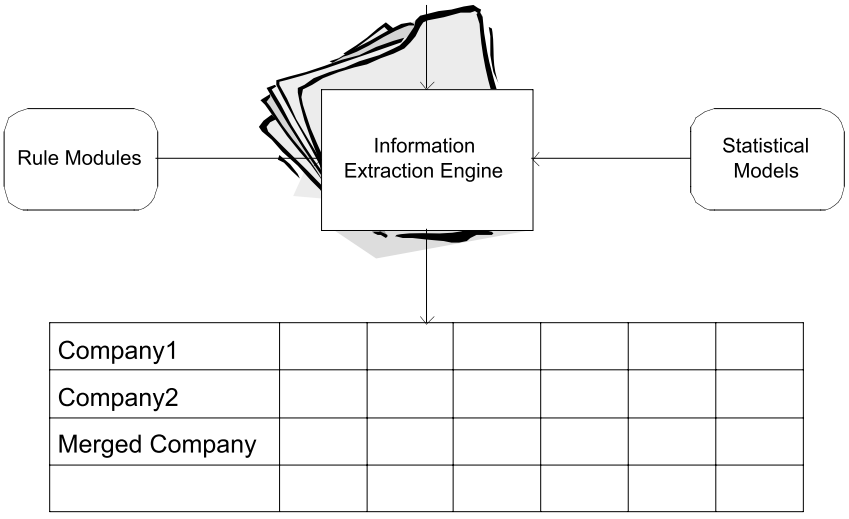

Figure VI.1. Schematic view of the information extraction process.

Consequently, IE methods allow for mining of the actual information present within the text rather than the limited set of tags associated with the documents. The IE process makes the number of different relevant *entities* and *relationships* on which the text mining is performed unbounded – typically thousands or even millions, which would be far beyond the number of tags any automated categorization system could handle. Thus, preprocessing techniques involving IE tend to create more rich and flexible representation models for documents in text mining systems.

IE can be seen as a limited form of "complete text comprehension." No attempt is made to understand the document at hand fully. Instead, one defines a priori the types of semantic information to be extracted from the document. IE represents documents as sets of entities and frames that are another way of formally describing the relationships between the entities.

The set of all possible entities and frames is usually open and very big compared with the set of categorization keywords. It cannot be created manually. Instead, the features are extracted directly from the text. The hierarchy relation between the entities and frames is usually a simple tree. The root has several children – the entity types (e.g., "*Company*," "*Person*," "*Gene*," etc.) under which the actual entities are automatically added as they are being discovered.

The frames constitute structured objects, and so they cannot be directly used as features for text mining. Instead, the frame attributes and its label are used for features. The frame itself, however, may bypass the regular text mining operations and may be fed directly to the querying and visualization components.

The simplest kind of information extraction is called *term extraction*. There are no frames, and there is only one entity type – simply "*term*."

Figure VI.1 gives a schematic view of the IE process. At the heart of the process we have the IE engine that takes a set of documents as input. The engine works by using a statistical model, a rule module, or a mix of both.

The output of the engine is a set of annotated frames extracted from the documents. The frames actually populate a table in which the fields of the frame are the rows of the table.

VI.1.1 Elements That Can Be Extracted from Text

There are four basic types of elements that can, at present, be extracted from text.

- ***Entities.*** Entities are the basic building blocks that can be found in text documents. Examples include people, companies, locations, genes, and drugs.
- ***Attributes.*** Attributes are features of the extracted entities. Some examples of attributes are the title of a person, the age of a person, and the type of an organization.
- ***Facts.*** Facts are the relations that exist between entities. Some examples are an employment relationship between a person and a company or phosphorylation between two proteins.
- ***Events.*** An event is an activity or occurrence of interest in which entities participate such as a terrorist act, a merger between two companies, a birthday and so on.

Figure VI.2 shows a full news article that demonstrates several tagged entities and relationships.

VI.2 HISTORICAL EVOLUTION OF IE: THE MESSAGE UNDERSTANDING CONFERENCES AND TIPSTER

The Defense Advanced Research Project Agency (DARPA) has been sponsoring efforts to codify and expand IE tasks, and the most comprehensive work has arisen from MUC-6 (Message Understanding Conference) and MUC-7 conferences. We now describe the various tasks introduced during the MUC conferences.

VI.2.1 Named Entity Recognition

The named entity recognition (NE, sometimes denoted also as NER) phase is the basic task-oriented phase of any IE system. During this phase the system tries to identify all mentions of proper names and quantities in the text such as the following types taken from MUC-7:

- People names, geographic locations, and organizations;
- Dates and times; and
- Monetary amounts and percentages.

The accuracy (F1) of the extraction results obtained on the NE task is usually quite high, and the best systems manage to get even up to 95-percent breakeven between precision and recall.

The NE task is weakly domain dependent – that is, changing the domain of the texts being analyzed may or may not induce degradation of the performance levels. Performance will mainly depend on the level of generalization used while developing the NE engine and on the similarity between the domains.

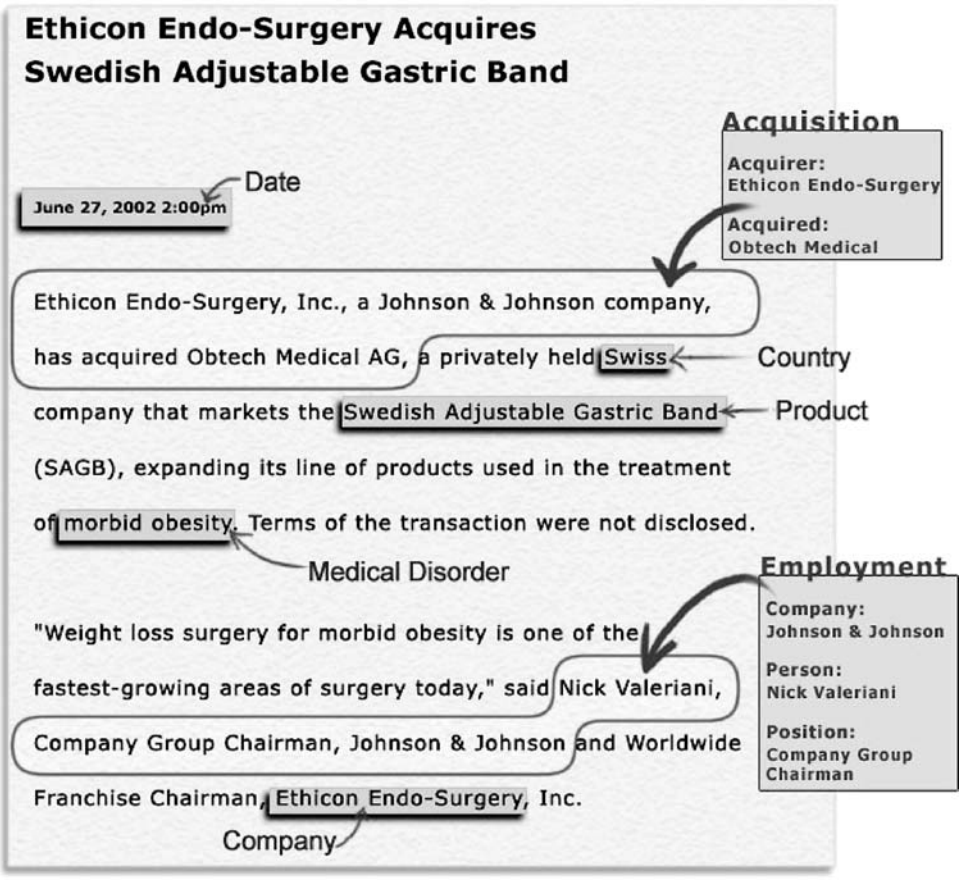

Figure VI.2. A tagged news article.

Proper names usually account for 70 percent of the named entities in the MUC corpuses, dates and times account for 25 percent, and monetary amounts and percentages account for less than 5 percent of the total named entities. Out of the named entities, about 45–50 percent are organization names, 12–32 percent are location tags, and 23–39 percent are people tags.

The MUC committee stipulated that the following types of noun phrases should not be extracted because they do not refer to any specific entity:

- Artifacts (e.g., *Wall Street Journal*, MTV, etc.),
- Common nouns used in anaphoric reference (such as the plane, the company, etc.),
- Names of groups of people and laws named after people (e.g., Republicans, "Gramm–Rudman amendment," "the Nobel Prize," etc.),
- Adjectival forms of location names (e.g., "American," "Japanese," etc.), and
- Miscellaneous uses of numbers that are not specifically currency or percentages.

VI.2.2 Template Element Task

Template element tasks (TEs) are independent or neutral with respect to scenario or domain. Each TE consists of a generic object and some attributes that describe it. This enables separating domain-independent from domain-dependent aspects of extraction.

The TE following types were included in MUC-7:

- Person
- Organization
- Location (airport, city, country, province, region, water)
- Artifact.

Here are examples of TEs. A typical paragraph of text from a press release is as follows below (taken from <http://www.itl.nist.gov/iaui/894.02/related_projects/muc/>):

> Fletcher Maddox, former Dean of the UCSD Business School, announced the formation of La Jolla Genomatics together with his two sons. La Jolla Genomatics will release its product Geninfo in June 1999. L.J.G. is headquartered in the Maddox family's hometown of La Jolla, CA.

One can extract various entities and descriptors. For instance, some of the entities and descriptors that can be automatically extracted from this paragraph by using information extraction algorithms include the following:

```
entity {
ID = 1,
NAME = "Fletcher Maddox"
DESCRIPTOR = "Former Dean of USCD Business School"
CATEGORY = person
}
entity {
ID = 2
NAME = "La Jolla Genomatics"
ALIAS = "LJG"
DESCRIPTOR = ""
CATEGORY = organization
}
entity {
ID = 3
NAME = "La Jolla"
DESCRIPTOR = "the Maddox family hometown"
CATEGORY = location
}
```

VI.2.3 Template Relationship (TR) Task

The Template relationship task (TR) expresses a domain-independent relationship between entities as compared with TEs, which just identify entities themselves. The goal of the TR task is to find the relationships that exist between the template elements extracted from the text (during the TE task). Just like the definition of an entity, entity attributes depend on the problem and the nature of the texts being analyzed; the relationships that may exist between template elements is domain dependent too. For example, persons and companies may be related by employee_of relation, companies and locations may be related by located_of relations, and companies may be interrelated by subdivision_of relations.

The following TRs were extracted from the sample text:

employee_of (Fletcher Maddox, UCSD Business School)
employee_of (Fletcher Maddox, La Jolla Genomatics)
product_of (Geninfo, La Jolla Genomatics)
location_of (La Jolla, La Jolla Genomatics)
location_of (CA, La Jolla Genomatics)

VI.2.4 Scenario Template (ST)

Scenario templates (STs) express domain and task-specific entities and relations. The main purpose of the ST tasks is to test portability to new extraction problems quickly. This task gives advantage to technologies that are not so labor intensive and hence can port the extraction engine to a new domain in a short time (couple of weeks).

Here are a few events that were extracted from the sample text:

```
company-formation-event {
PRINCIPAL = "Fletcher Maddox"
DATE = ""
CAPITAL = ""
}
product-release-event {
COMPANY = "La Jolla Genomatics"
PRODUCS = "Geninfo"
DATE = "June 1999"
COST = ""
}
```

VI.2.5 Coreference Task (CO)

The coreference task (CO) captures information on coreferring expressions (e.g., pronouns or any other mentions of a given entity), including those tagged in the NE, TE tasks. This CO focuses on the IDENTITY (IDENT) relation, which is symmetrical and transitive. It creates equivalence classes (or coreference chains) used for scoring. The task is to mark nouns, noun phrases, and pronouns.

```
"It's a chance to think about first-level
questions",said Ms. <enamex type=
"PERSON">Cohn</enamex>, a partner in the <enamex type=
"ORGANIZATION"> McGlashan & Sarrail</enamex> firm in
<enamex type= "LOCATION">San Mateo</enamex>, Senamex
type= "LOCATION">Calif.</enamex>
```

Figure VI.3. MUC-style annotation.

Consider the following sentence:

David$_1$ came home from school, and saw his$_1$ mother$_2$,
Rachel$_2$. She$_2$ told him$_1$ that his$_1$ father will be late.

The correctly identified pronominal coreference chains are (**David**$_1$, **his**$_1$, **him**$_1$, **his**$_1$) and (**mother**$_2$, **Rachel**$_2$, **She**$_2$).

This is not a high-accuracy task for IE systems but properly resolving some kinds of coreference is usually difficult even for humans annotators, who achieved about 80 percent.

An MUC-style tagging is shown in Figure VI.3, and a sample template extracted from that text fragment is shown in Figure VI.4.

VI.2.6 Some Notes about IE Evaluation

We follow here the discussion of Lavelli et al. (2004) about various problems in the common evaluation methodology of information extraction. The main problem is that it is very hard to compare different IE experiments without comparing the exact settings of each experiment. In particular the following problems were raised:

- The exact split between the training set and test set: considering both the proportions between the two sets (e.g., a 50/50 versus a 90/10 split) and the repetition procedure adopted in the experiment (e.g., a single specific split between training and test versus *n* repeated random splits versus *n*-fold cross-validations).
- Determining the test set: the test set for each point on the learning curve can be the same (hold-out set) or be different and based on the exact split.
- What constitutes an exact match: how to treat an extraneous or a missing comma – that is, should it be counted as a mistake or is it close enough and does not miss any critical information.
- Feature Selection: many different types of features can be used, including orthographic features, linguistic features (such as POS, stemming, etc.), and semantic features based on external ontologies. In order to compare any two algorithms properly they must operate on the same set of features.

```
<ORGANIZATION-9303020074-1> :=
    ORG_NAME: "McGlashan & Sarrail"
    ORG_ALIAS: "M & S"
    ORG_LEADER: <PERSON-9303020074-57>
    ORG_TYPE: COMPANY
```

Figure VI.4. MUC-style templates.

Counting the Correct Results

- Exact Matches: Instances generated by the extractor that perfectly match actual instances annotated by the domain expert.
- Contained Matches: Instances generated by the extractor that contain actual instances annotated by the domain expert and some padding from both sides.
- Overlapped Matches: Instances generated by the extractor that overlap actual instances annotated by the domain expert (at least one word is in the intersection of the instances).

Another aspect is how to treat entities that appear multiple times in a document. One option is to extract all of them, and then any omission will result in lower recall. Another option is to extract each entity just once; hence, it is enough just to identify one occurrence of each entity. There are situations in which the latter option will actually make sense if we are just interested in knowing which entities appear in each document (and we do not care how many times it appears).

VI.3 IE EXAMPLES

This section provides several real-world examples of input documents and the results obtained by performing information extraction on them. The examples have been culled from a variety of domains and demonstrate a broad range tagging standards to give the reader an exposure to the different ways to approach coding the information exaction process.

VI.3.1 Case 1: Simplistic Tagging, News Domain

Consider a system that extracts business events from news articles. Such a system is useful for business analysts or even casual users interested in keeping abreast of the current business events. Consider the following text fragment:

> "TeliaSonera, the Nordic region's largest telecoms operator, was formed in 2002 from the cross-border merger between Telia and Finland's Sonera,"

One can extract the following frame from it:

FrameName: Merger
Company1: Telia
Company2: Sonera
New Company: TeliaSonera

This frame actually provides a concise summary of the previous text fragment. The following cases will show the types of summary information that can be extracted from other text fragments.

VI.3.2 Case 2: Natural Disasters Domain

> **4 Apr Dallas** – Early last evening, a tornado swept through an area northwest of Dallas, causing extensive damage. Witnesses confirm that the twister occurred without warning at approximately 7:15 p.m. and destroyed the mobile homes.

The Texaco station, at 102 Main Street, Farmers Branch, TX, was severely damaged, but no injuries were reported. Total property damages are estimated at $350,000.

Event:	tornado
Date:	4/3/97
Time:	19:15
Location:	Farmers Branch : "northwest of Dallas" : TX : USA
Damage:	mobile homes
	Texaco station
Estimated Losses:	$350,000
Injuries:	none

VI.3.3 Case 3: Terror-Related Article, MUC-4

19 March – a bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no causalities have been reported. According to unofficial sources, the bomb – allegedly detonated by urban guerrilla commandos – blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).

Incident Type:	Bombing
Date:	March 19th
Location:	El Salvador: San Salvador (City)
Perpetrator:	urban guerrilla commandos
Physical Target:	power tower
Human Target:	–
Effect of Physical Target:	destroyed
Effect on Human Target:	no injury or death
Instrument	bomb

VI.3.4 Technology-Related Article, TIPSTER-Style Tagging

Here is an article from the MUC-5 evaluation dealing with microelectronics.

```
<doc>
<REFNO> 000019641 </REFNO>
<DOCNO> 3560177 </DOCNO>
<DD> November 25, 1991 </DD>
<SO> News Release </SO>
<TXT>
Applied Materials, Inc. today announced its newest source technology, called
the Durasource, for the Endura(TM) 5500 PVD system. This enhanced source
includes new magnet configurations, giving the industry's most advanced
```

sputtered aluminum step coverage in sub-micron contacts, and a new one piece target that more than doubles target life to approximately 8000 microns of deposition compared to conventional two-piece "bonded" targets. The Durasource enhancement is fully retrofittable to installed Endura 5500 PVD systems. The Durasource technology has been specially designed for 200 mm wafer applications, although it is also available for 125 mm and 1s0mm wafer sizes. For example, step coverage symmetry is maintained within 3% between the inner and outer walls of contacts across a 200 mm wafer. Film thickness uniformity averages 3% (3 sigma) over the life of the target.
</TXT>
</doc>

<TEMPLATE-3560177-1> :=
DOC NR: 3560177
DOC DATE: 251192
DOCUMENT SOURCE: "News Release"
CONTENT:
<MICROELECTRONICS_CAPABILITY-3560177-1>
DATE TEMPLATE COMPLETED: 021292
EXTRACTION TIME: 5
COMMENT: "article focuses on nonreportable target source but reportable info available"
/"TOOL_VERSION: LOCKE.3.4"
/"FILLRULES_VERSION: EME.4.0"

<MICROELECTRONICS_CAPABILITY-3560177-1> :=
PROCESS: <LAYERING-3560177-1>
MANUFACTURER: <ENTITY-3560177-1>

<ENTITY-3560177-1> :=
NAME: Applied Materials, INC
TYPE: COMPANY

<LAYERING-3560177-1> :=
TYPE: SPUTTERING
FILM: ALUMINUM
EQUIPMENT: <EQUIPMENT-3560177-1>

<EQUIPMENT-3560177-1> :=
NAME_OR_MODEL: "Endura(TM) 5500"
MANUFACTURER: <ENTITY-3560177-1>
EQUIPMENT_TYPE: PVD_SYSTEM
STATUS: IN_USE
WAFER_SIZE: (200 MM)
(125 MM)
COMMENT: "actually three wafer sizes, third is error 1s0mm"

VI.3.5 Case 5: Comprehensive Stage-by-Stage Example

- ***Original Sentence*:** Mr. Eskew was Vice President of Worldwide Sales for Sandpiper Networks, which was recently acquired by Digital Island where he created the worldwide sales strategy.
- ***After Part of Speech Tagging*:**
  <Prop>Mr. Eskew</Prop> <Verb>was</Verb> <Prop>Vice President</Prop> <Prep>of</Prep> <Prop>Worldwide Sales</Prop> <Prep>for</Prep> <Prop>Sandpiper Networks</Prop> which <Verb>was</Verb> <Adv>recently</Adv> <Verb>acquired</Verb> <Prep>by</Prep> <Prop>Digital Island</Prop> where <Pron>he</Pron> <Verb>created</Verb> <Det>the</Det> <Adj>worldwide</Adj> <Nn>sales strategy.</Nn>
- ***After Shallow Parsing*:**
  NP:{Mr. Eskew} was NP:{Vice President of Worldwide Sales} for NP:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by NP:{Digital Island} where NP:{he} V:{created} NP:{the worldwide sales strategy}
- ***After Named Entity Recognition*:**
  Person:{Mr. Eskew} was Position:{Vice President of Worldwide Sales} for Company:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by Company:{Digital Island} where Person:{he} V:{created} NP:{the worldwide sales strategy}
- ***After Merging (Anaphora Resolution)*:**
  Person:{Mr. Eskew} was Position:{Vice President of Worldwide Sales} for Company:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by Company:{Digital Island} where Person:{Mr. Eskew} V:{created} NP:{the worldwide sales strategy}
- ***Frames Extracted*:**
  Frame Type: **Acquisition**
  Acquiring Company: Digital Island
  Acquired Company: Sandpiper Networks
  Acquisition Status: Historic

  FrameType: **PersonPositionCompany**
  Person: Mr. Eskew
  Position: Vice President of Worldwide Sales
  Company: Sandpiper Networks
  Status: Past

VI.4 ARCHITECTURE OF IE SYSTEMS

Figure VI.5 shows the generalized architecture for a basic IE system of the type that would be used for text mining preprocessing activities. The subcomponents are colored according to their necessity within the full system.

A typical general-use IE system has three to four major components. The first component is a tokenization or *zoning* module, which splits an input document into its basic building blocks. The typical building blocks are words,

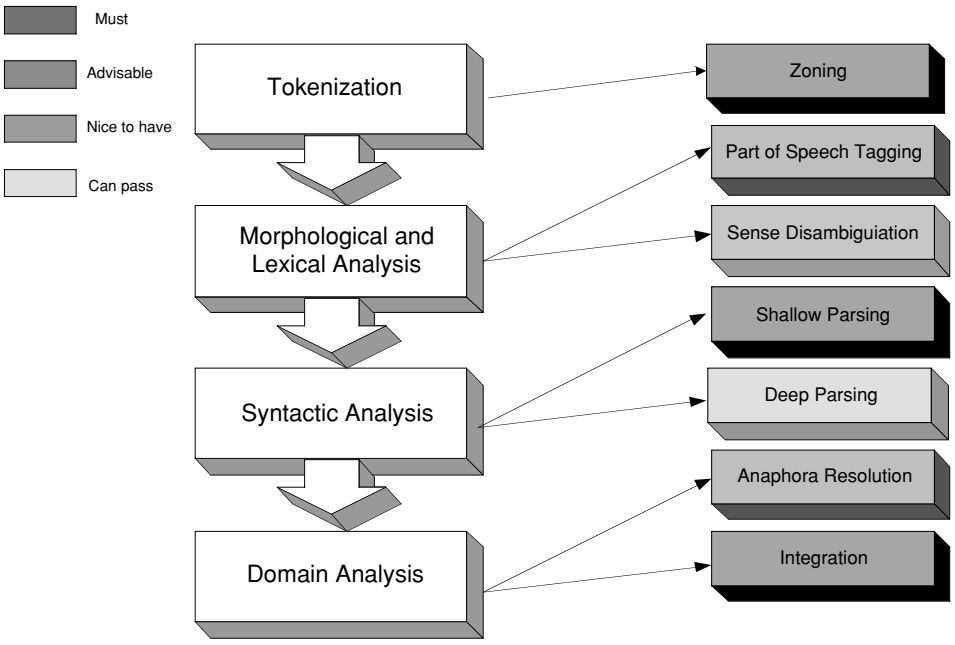

Figure VI.5. Architecture of a typical information extraction system.

sentences and paragraphs. Rarely we may have higher building blocks like sections and chapters.

The second component is a module for performing *morphological and lexical analysis*. This module handles activities such as assigning POS tags to the document's various words, creating basic phrases (like noun phrases and verb phrases), and disambiguating the sense of ambiguous words and phrases.

The third component is a module for *syntactic analysis*. This part of an IE system establishes the connection between the different parts of each sentence. This is done either by doing full parsing or shallow parsing.

A fourth and increasingly more common component in IE systems performs what might be termed *domain analysis*, which is a function in which the system combines all the information collected from the previous components and creates complete frames that describe relationships between entities. Advanced domain analysis modules also possess an anaphora resolution component. Anaphora resolution concerns itself with resolving indirect (and usually pronomic) references for entities that may appear in sentences other than the one containing the primary direct reference to an entity.

VI.4.1 Information Flow in an IE System

Most information extraction systems use a deterministic bottom-up approach to analyzing the document. Initially, one identifies the low-level elements and then identifies higher level features that are based on the low-level features identified in the previous phases.

Processing the Initial Lexical Content: Tokenization and Lexical Analysis

The first two phases of an IE system really both concern themselves with processing a document to identify various elements of its basic lexical content. As a first pass, a document is divided into tokens, sentences, and possibly paragraphs. Then, each word is tagged by its part of speech and lemma.

In addition, an IE system can use specialized dictionaries and gazetteers to tag words that appear in those word lists. Typical dictionaries include names of countries, cities, people's first names, public companies, company suffixes, common titles in companies, and so on. Dictionary support during initial tagging creates richer document representations. For example, using appropriate dictioniaries, the word "Robert" would be tagged as a "first name," "IBM" would be tagged as a company, and the acronym "spa" could be tagged as "company suffix."

Proper Name Identification

Commonly, the next phase is proper name identification. After an IE system performs the basic lexical analysis, it is typically designed to try to identify a variety of simple entity types such as dates, times, e-mail address, organizations, people names, locations, and so on. The entities are identified by using regular expressions that utilize the context around the proper names to identify their type. The regular expressions can use POS tags, syntactic features, and orthographic features such as capitalization.

Proper name identification is performed by scanning the words in the sentence while trying to match one of the patterns in the predefined set of regular expressions. Each proper name type has its associated set of regular expressions. All patterns are attempted for each word. If more than one pattern is matched, the IE system picks the pattern that matches the longest word sequence. If there is a tie, the IE system usually just uses the first pattern. If no pattern matches, the IE system moves to the next word and reapplies the entire set of patterns. The process continues until the end of the sentence is reached.

To illustrate how such regular expressions are constructed, we present several regular expressions for identifying people names below.

1. @Honorific CapitalizedWord CapitalizedWord
   a. @Honorific is a list of honorific titles such as {Dr., Prof., Mr., Ms., Mrs. etc.)
   b. Example: Mr. John Edwards
2. @FirstNames CapitalizedWord
   a. @FirstNames is a list of common first names collected from sites like the U.S. census and other relevant sites
   b. Example: Bill Hellman
3. CapitalizedWord CapitalizedWord [,] @PersonSuffix
   a. @PersonSuffix is a list of common suffixes such as {Jr., Sr., II, III, etc.}
   b. Example: Mark Green, Jr.
4. CapitalizedWord CapitalLetter [.] CapitalizedWord
   a. CapitalLetter followed by an optional period is a middle initial of a person and a strong indicator that this is a person name
   b. Example: Nancy M. Goldberg

Element	Grammatical Function	Type
E1	NP	Company
E2	VG	
E3	NP	Person
E4	NP	Position
E5	NP	Position
E6	NP	Company
E7	NP	Location
E8	VG	
E9	NP	Position

Figure VI.6. Identifying a text element's grammatic function and type.

5. CapitalizedWord CapitalLetter @PersonVerbs
   a. @PersonVerbs is a list of common verbs that are strongly associated with people such as {said, met, walked, etc.}

A more expansive treatment of the topic of manual rule writing and pattern development is offered in Appendix A.

Shallow Parsing

After identifying the basic entities, an IE system moves to shallow parsing and identification of noun and verb groups. These elements will be used as building blocks for the next phase that identifies relations between these elements. As an example, consider the following annotated text fragment:

> Associated Builders and Contractors (ABC)$_{\mathbf{E1}}$ today announced$_{\mathbf{E2}}$ that Bob Piper$_{\mathbf{E3}}$, co-owner$_{\mathbf{E4}}$ and vice president of corporate operations$_{\mathbf{E5}}$, Piper Electric Co., Inc.$_{\mathbf{E6}}$, Arvada, Colo.$_{\mathbf{E7}}$, has been named$_{\mathbf{E8}}$ vice president of workforce development$_{\mathbf{E9}}$.

Essentially, at this point, an IE system focuses on creating a comprehensive listing of the types of elements found in such a text fragment in the manner shown in Figure VI.6.

The next step performed by an IE system is the construction of noun groups based on the noun phrases (NPs) that were constructed before. The construction is based on common patterns developed manually. Essentially, this works in the following manner. On the basis of a few typical patterns such as

1. Position and Position, Company
2. Company, Location,

one can construct the following noun groups (NGs):

1. co-owner$_{\mathbf{E4}}$ and vice president of corporate operations$_{\mathbf{E5}}$, Piper Electric Co., Inc.$_{\mathbf{E6}}$
2. Piper Electric Co., Inc.$_{\mathbf{E6}}$, Arvada, Colo.$_{\mathbf{E7}}$.

Already, even at the conclusion of the initial tokenization and lexical analysis stages of the IE system's preprocessing operations, a relatively rich amount of structure has been created to represent the text of a particular document. This structure

will be useful as a building block for further phases of IE-related preprocessing operations.

Building Relations

The construction of relations between entities is done by using domain-specific patterns. The generality of the patterns depends on the depth of the linguistic analysis performed at the sentence level. If one just performs this analysis against individual noun or verb phrases, or both, then one will need to develop five to six times more patterns than if simply the subject, verb, and object of each sentence were identified. To extract an executive appointment event from the text fragment above, one could use the following pattern:

> Company [Temporal] @Announce Connector Person PersonDetails @Appoint Position

This pattern can be broken down in the following way:

- ***Temporal*** is a phrase indicating a specific date and/or time such as {yesterday, today, tomorrow, last week, an hour ago}
- ***@Announce*** is a set of phrases that correspond to the activity of making a public announcement like {announced, notified, etc.}
- ***Connector*** is a set of connecting words like {that, . . . }
- ***PersonDetails*** is a phrase describing some fact about a person (such as his or her age, current position, etc.); it will be usually surrounded by commas
- ***@Appoint*** is a set of phrases that correspond to the activity of appointing a person to a position like {appointed, named, nominated, etc.}

One of the main tasks during the relation extraction is coreference resolution, which is introduced in Section VI.5. We expand on coreference resolution in Section VI.5.

Inferencing

In many cases, an IE system has to resort to some kind of common sense reasoning and infer missing values to complete the identification of events. The inference rules are written as a formalism similar to Prolog clauses. Common examples include family relations, management changes, spatial relations, and so on.

Below are two examples, one related to location of a person and the other to the position a person is going to fill. The first example is a simple two-sentence text fragment.

> **Example 1:** John Edgar was reported to live with Nancy Leroy. His Address is 101 Forest Rd., Bethlehem, PA.

From this, it is possible to extract the following entities and events:

1. person(John Edgar)
2. person(Nancy Leroy)
3. livetogether(John Edgar, Nancy Leroy)
4. address(John Edgar, 101 Forest Rd., Bethlehem, PA)

Using the following rule, one can infer that Nancy Leroy lives at 101 Forest Rd., Bethlehem, PA.

address(P1,A) :- person(P1), person(P2), livetogether(P1,P2), address(P1,A).

The second example is also a two-sentence text fragment.

Example 2: RedCarpet Inc. announced that its President, JayGoldman, has resigned. The company appointed Fred Robbins to the position.

From this one can extract the following entities and events:

1. company(RedCarpet)
2. person(Jay Goldman)
3. personLeftPosition(Jay Goldman, RedCarpet, President)
4. personReplacesPerson(Fred Robbins, Jay Goldman)

Using the following rule in this second example, one can infer that Fred Robbins is the new President of RedCarpet:

newposition(P2,Pos) :- person(P1), person(P2), company(C1), personLeftPosition(P1,C1,Pos), personReplacesPerson (P2,P1).

VI.5 ANAPHORA RESOLUTION

Anaphora or *coreference resolution* is the process of matching pairs of NLP expressions that refer to the same entity in the real world. It is a process that is critical to the proper function of advanced text mining preprocessing systems.

Below is an example of an annotated text fragment that includes chains of coreffering phrases. We can see here two chains referring to a person (#1, #5), one chain referring to an incident (#2), one chain referring to groups of people (#4), two chains referring to locations (#3, #7), and one chain referring to an organization (#6).

HADERA, Israel$_3$ (AP) – A Palestinian gunman$_1$ walked into a wedding hall in northern Israel$_3$ late Thursday and opened fire, killing six people and injuring 30$_2$, police$_6$ said.... Police$_6$ earlier said the attacker$_1$ threw hand grenades but witnesses and later police$_6$ accounts said the attacker$_1$ opened fire with an M-16 and was$_1$ stopped before he$_1$ could throw a grenade. The Al Aqsa Brigades$_4$, a militia$_4$ linked to Yasser Arafat's Fatah claimed responsibility. The group$_4$ said that Abed Hassouna$_1$ from a village$_7$ near the Palestinian town of Nablus carried out the attack$_2$ to avenge the death of Raed Karmi$_5$, (the militia)$_4$'s leader$_5$ in the town of Tulkarem. Hassouna$_1$ had been a Palestinian policeman$_1$ but left$_1$ the force two years ago, residents of his$_1$ village$_7$ said.

There are two main approaches to anaphora resolution. One is a knowledge-based approach based on linguistic analysis of the sentences and is coded as a rigid algorithm. The other approach is a machine learning approach based on an annotated corpus.

VI.5.1 Pronominal Anaphora

Pronominal anaphora deals with resolving pronouns such as he, she, and they, It is the most common type of coreference. There are three types of pronouns:

- ***Reflexive pronouns***: himself, herself
- ***Personal pronouns***: he, him, you
- ***Possessive pronouns***: her, his, hers

It should be pointed out that not all pronouns in English are anaphoric. For instance, "it" can often be nonanaphoric as in the case of the previous sentence. Other examples of nonanaphoric "it" include expressions such as "It is important," "It is necessary," or "It has to be taken into account." A nonanaphoric "it" is described as *pleonastic*.

VI.5.2 Proper Names Coreference

The task here is to link together all the variations of a proper name (person, organization, location) that are observed in text. For example,

> Former President Bush$_1$ defended the U.S. military Thursday during a speech at one of the nation's largest Army posts, where one private accused of abusing Iraqi prisoners awaits a court-martial. "These are difficult times for the Army as the actions of a handful in Iraq violate the soldier's code," said George H. W. Bush$_1$.

Additional examples can be observed in the example above.

VI.5.3 Apposition

Appositives are used to provide auxiliary information for a named entity. This information is separated from the entity by a comma and either precedes it or comes directly after it as in the following example:

> said George H. W. Bush$_1$, the father of President Bush$_1$. the father of President Bush$_1$, George H. W. Bush$_1$ said. . . .

A necessary condition for an appositional phrase to corefer to a named entity is that they occur in different noun phrases. If the apposition is a modifier of the named entity within the same noun phrase, then they are not considered coreferring as in the following phrase "Former President Bush." In this case *Former President* is not coreferring to Bush.

VI.5.4 Predicate Nominative

A predicate nominative occurs after a copulative verb (is, seems, looks like, appears, etc.) and completes a reference to the subject of a clause.

An example follows:

> Bill Gates$_1$ is the Chairman of Microsoft Corporation$_1$
> Subject: Bill Gates
> Predicate Nominative: the Chairman of Microsoft Corporation

A predicate nominative is a candidate for coreference only if it is stated in a firm way. If it is stated in a speculative or negative way, then it is not a candidate for coreference.

VI.5.5 Identical Sets

In this type of coreference the anaphor and the antecedent both refer to sets that are identical or to identical types. In the following example, "The Al Aqsa Brigades," "a militia," and "The group" all refer to the same set of people.

> The Al Aqsa Brigades$_4$, a militia$_4$ linked to Yasser Arafat's Fatah claimed responsibility. The group$_4$ said that Abed Hassouna$_1$ from a village$_7$ near the Palestinian town of Nablus carried out the attack$_2$ to avenge the death of Raed Karmi$_5$, (the militia)$_4$'s leader$_5$

Identifying identical sets is usually extremely difficult because deep knowledge about the domain is needed.

If we have a lexical dictionary such as WordNet that include hyponyms and hypernyms, we may be able to identify identical sets. We can deduce, for instance, that "militia" is a kind of "group."

VI.5.6 Function–Value Coreference

A function–value coreference is characterized by phrases that have a function–value relationship. Typically, the function will be descriptive and the value will be numeric.

In the following text there are two function–value pairs:

> Evolved Digital Systems's Revenues$_1$ were \$4.1M$_1$ for the quarter, up 61% compared to the first quarter of 2003. Net Loss$_2$ declined by 34% to \$5.6M$_2$.

Function: Evolved Digital Systems's Revenues
Value: \$4.1M
Function: Net Loss
Value: \$5.6M

VI.5.7 Ordinal Anaphora

Ordinal anaphora involves a cardinal number like *first* or *second* or an adjective such as *former* or *latter* as in the following example:

> IBM and Microsoft$_1$ were the final candidates, but the agency preferred the latter company$_1$.

VI.5.8 One-Anaphora

A one-anaphora consists of an anaphoric expression realized by a noun phrase containing the word "one" as in the following:

> If you cannot attend a tutorial$_1$ in the morning, you can go for an afternoon one$_1$.

VI.5.9 Part–Whole Coreference

Part–whole coreference occurs when the anaphor refers to a part of the antecedent as in the following:

John has bought a new car$_1$. The indicators$_1$ use the latest laser technology.

As in the case of identifying identical sets discussed in Section VI.5.5, a lexical resource such WordNet is needed. In particular, WordNet includes the meronymy–holonymy relationship, which can help us identify that indicators are a part of a car.

VI.5.10 Approaches to Anaphora Resolution

Most of the work on coreference resolution focuses on pronominal resolution because that is the most common type of coreference and is also one of the easier types to resolve.

Most of the approaches to pronominal resolution share a common overall structure:

- Identify the relevant paragraphs (or sentences) around the pronoun in which one will search for candidates antecedents.
- Using a set of consistency checks, delete the candidates that to do meet any of the checks.
- Assign salience values to each of the surviving candidates according to a set of predefined rules.
- Pick the candidate with the highest salience value.

Some of these approaches require heavy preprocessing and rely on full parsers, whereas others are fairly knowledge-poor and rely on shallow parsing. The focus here will be on approaches that do not require full parsing of the sentences because doing this is too time-consuming and hence prohibitive in a real-world IE system.

VI.5.10.1 Hobbs Algorithm

The most simplistic algorithm is the *Hobbs algorithm*, which is also called the *Naive algorithm* (Hobbs 1986). This algorithm works by specifying a total order on noun phrases in the prior discourse and comparing each noun phrase against a set of constraints imposed by the features of the anaphor (i.e., gender, number). The first antecedent to satisfy all the constraints is chosen.

A few points to note about this algorithm are as follows:

- For two candidate antecedents a and b, if a is encountered before b in the search space, then a is preferred over b.
- The salience given to the candidate antecedents imposes a total ordering on the antecedents – that is, no two antecedents will have the same salience.
- The algorithm can not handle ambiguity and will resolve a pronoun as if there were at least one possible antecedent.

VI.5.11 CogNIAC (Baldwin 1995)

CogNIAC is a pronoun resolution engine designed around the assumption that there is a subclass of anaphora that does not require general purpose reasoning. Among the kinds of information CogNIAC does require are POS tagging, simple noun phrase recognition, and basic semantic category information like gender and number.

The system is based on a set of high-confidence rules that are successively applied over the pronoun under consideration. The rules are ordered according to their importance and relevance to anaphora resolution. The processing of a pronoun stops when one rule is satisfied. Below are listed the six rules used by the system. For each of them, the ***sentence prefix of anaphor*** is defined as the text portion of the sentence from the beginning of the sentence to the position of the anaphor.

1. ***Unique Antecedent.***
   Condition*:** If there is a single valid antecedent ***A in the relevant discourse.
   Action*: *A is the selected antecedent.
2. ***Reflexive Pronoun.***
   ***Condition*:** If the anaphor is a reflexive pronoun.
   ***Action*:** Pick the nearest valid antecedent in the anaphor prefix of the current sentence.
3. ***Unique in Current + Preceding.***
   ***Condition*:** If there is a single valid antecedent A in the preceding sentence and anaphor prefix of the current sentence.
   Action*: *A is the selected antecedent.
   ***Example*: Rupert Murdock**'s News Corp. confirmed his interest in buying back the ailing New York Post. But analysts said that if **he** winds up bidding for the paper, ...
4. ***Possessive Pronoun.***
   ***Condition*:** If the anaphor is a possessive pronoun and there is a single exact copy of the possessive phrase in the previous sentence.
   ***Action*:** The antecedent of the latter copy is the same as the former.
   ***Example*:** After he was dry, Joe carefully laid out the damp towel in front of **his locker**. Travis went over to **his locker**, took out a towel and started to dry off.
5. ***Unique in Current Sentence.***
   Condition*:** If there is a single valid antecedent ***A in the anaphor-prefix of the current sentence
   Action*: *A is the selected antecedent.
6. ***Unique Subject–Subject Pronoun.***
   ***Condition*:** If the subject of the previous sentence is a valid antecedent *A* and the anaphor is the subject of the current sentence.
   Action*: *A is the selected antecedent.

VI.5.11.1 Kennedy and Boguraev

This approach is based on Lappin and Leass's (1994) method but without the need for full parsing. This algorithm was used to resolve personal pronouns, reflexives, and possessives. The algorithm works by constructing coreference equivalence classes.

Each such class has a salience that is computed based on a set of 10 factors. Each pronoun is resolved to the antecedent that belongs to the class with the highest salience.

Here are the factors used by the salience algorithm. All the conditions refer to the current candidate for which we want to assign salience. *GFUN* is the grammatical function of the candidate

SENT-S: 100 iff in the current sentence
CNTX-S: 50 iff in the current context
SUBJ-S: 80 iff GFUN = *subject*
EXST-S: 70 iff in an existential construction
POSS-S: 65 iff GFUN = *possessive*
ACC-S: 50 iff GFUN = *direct object*
DAT-S: 40 iff GFUN = *indirect object*
OBLQ-S: 30 iff the complement of a preposition
HEAD-S: 80 iff EMBED = NIL
ARG-S: 50 iff ADJUNCT = NIL

As an example of the salience assignment, consider the following text fragment:

> Sun's prototype Internet access device uses a 110-Mhz MicroSPARCprocessor, and is diskless. Its dimensions are 5.5 inches × 9 inches × 2 inches.

Anaphors and candidates are represented using their offset in the text (from the beginning of the document), their grammatical function, and several other syntactic features.

The structure of each candidate is Element: Offset/Salience

ANAPHOR:	Its : 347
CANDIDATES:	
Internet access device:	335/180 (=50+85+50)
MicroSPARCprocessor:	341/165 (=50+65+50)
Sun's:	333/140 (=50+40+50)

The first sentence in this fragment includes three candidates with different grammatical functions. The second sentence, which includes that anaphor, does not include any candidate satisfying the basic constraints. The three candidates in the first sentence are ranked according to their salience.

The main factor determining the salience is the grammatical function of each candidate. *Internet access device* is the subject of the sentence and hence satisfies the SUBJ-S condition, is the optimal candidate, and is selected as the antecedent of Its.

VI.5.11.2 Mitkov

In contrast to the previous approaches that use mostly positive rules, Mitkov's approach (Mitkov 1998) is based on a set of boosting and impeding indicators applied to each candidate. The approach takes as an input the output of a text processed by a part-of-speech tagger, identifies the noun phrases that precede the anaphor within a distance of two-sentences, checks them for gender and number agreement with the anaphor, and then applies the genre-specific antecedent indicators to the remaining candidates.

The boosting indicators assign a positive score to a matching candidate, reflecting a positive likelihood that it is the antecedent of the current pronoun. In contrast, the impeding indicators apply a negative score to the matching candidate, reflecting a lack of confidence that it is the antecedent of the current pronoun. The candidate with the highest combined score is selected.

Here are some of the indicators used by Mitkov:

- ***Definiteness.*** Definite noun phrases in previous sentences are more likely antecedents of pronominal anaphors than indefinite ones (definite noun phrases score 0 and indefinite ones are penalized by −1).
- ***Givenness.*** Noun phrases in previous sentences representing the "given information" are deemed good candidates for antecedents and score 1 (candidates not representing the theme score 0). The given information is usually the first noun phrase in a nonimperative sentence.
- ***Indicating Verbs.*** If a verb in the sentence has a stem that is a member of {discuss, present, illustrate, identify, summarize, examine, describe, define, show, check, develop, review, report, outline, consider, investigate, explore, assess, analyze, synthesize, study, survey, deal, cover}, then the first NP following the verb is the preferred antecedent.
- ***Lexical Reiteration.*** Lexically reiterated noun phrases are preferred as candidates for antecedent (an NP scores 2 if is repeated within the same paragraph twice or more, 1 if repeated once, and 0 if it is not repeated). The matching is done in a loose way such that synonyms and NPs sharing the same head are considered identical.
- ***Section Heading Preference.*** If a noun phrase occurs in the heading of the section containing the current sentence, then we consider it the preferred candidate.
- "***Nonprepositional*" *Noun Phrases.*** A "nonprepositional" noun phrase is given a higher preference than a noun phrase that is part of a prepositional phrase (0, −1). Example: `Insert the cassette into the VCR making sure it is suitable for the length of recording`. Here VCR is penalized for being part of a prepositional phrase and is resolved to the cassette.
- ***Collocation Pattern Preference.*** This preference is given to candidates having an identical verb collocation pattern with a pronoun of the pattern "noun phrase (pronoun), verb" and "verb, noun phrase (pronoun)." *Example*: `Press the key down and turn the volume up... Press it again`. Here key is preferred antecedent because it shares the same verb (press) with the pronoun ("it").
- ***Immediate Reference.*** Given a pattern of the form "... You? V1 NP... con you? V2 it (*con* you? V3 it)", where *con* ∈ {and/or/before/after...}, the noun phrase immediately after V1 is a very likely candidate for the antecedent of the pronoun "it" immediately following V2 and is therefore given preference (scores 2 and 0). Example:

> To print the paper$_1$, you can stand the printer$_2$ up or lay it$_2$ flat. To turn on the printer$_2$, press the Power button$_3$ and hold it$_3$ down for a moment. Unwrap the the paper$_1$, form it$_1$ and align it$_1$ then load it$_1$ into the drawer.

- ***Referential distance.*** In complex sentences, noun phrases receive the following scores based on how close they are to the anaphor:
  - previous clause: 2
  - previous sentence: 1
  - 2 sentences: 0
  - 3 sentences further back: −1

  In simple sentences, the scores are as follows:
  - previous sentence: 1
  - 2 sentences: 0
  - 3 sentences further back: −1
- ***Domain Terminology Preference.*** NPs representing domain terms are more likely to be the antecedent (score 1 if the NP is a term and 0 if not).

VI.5.11.3 Evaluation of Knowledge-Poor Approaches

For many years, one of the main problems in contrasting the performance of the various systems and algorithms had been that there was no common ground on which such a comparison could reasonably be made. Each algorithm used a different set of documents and made different types of assumptions.

To solve this problem, Barbu (Barbu and Mitkov 2001) proposed the idea of the "evaluation workbench" – an open-ended architecture that allows the incorporation of different algorithms and their comparison on the basis of the same preprocessing tools and data. The three algorithms just described were all implemented and compared using the same workbench.

The three algorithms implemented receive as input the same representation of the input file. This representation is generated by running an XML parser over the file resulting from the preprocessing phase. Each noun phrase receives the following list of features:

- the original word form
- the lemma of the word or of the head of the noun phrase
- the starting and ending position in the text
- the part of speech
- the grammatical function (subject, object . . .)
- the index of the sentence that contains the referent
- the index of the verb related to the referent.

In addition, two definitions should be highlighted as follows:

- ***Precision*** = number of correctly resolved anaphors / number of anaphors attempted to be resolved
- ***Success Rate*** = number of correctly resolved anaphors / number of all anaphors.

The overall results as reported in Mitkov are summarized in the following table:

	K&B	Cogniac	Mitkov
Precision	52.84%	42.65%	48.81%
Success	61.6%	49.72%	56.9%

VI.5.11.4 Machine Learning Approaches

One of the learning approaches (Soon et al. 2001) is based on building a classifier based on the training examples in the annotated corpus. This classifier will be able to take any pair of NLP elements and return true if they refer to the same real-world entity and false otherwise. The NLP elements can be nouns, noun phrases, or pronouns and will be called *markables*.

The markables are derived from the document by using the regular NLP preprocessing steps as outlined in the previous section (tokenization, zoning, part of speech tagging, noun phrase extraction and entity extraction). In addition to deriving the markables, the preprocessing steps make it possible to create a set of features for each of the markables. These features are used by the classifier to determine if any two markables have a coreference relation.

Some Definitions

- ***Indefinite Noun Phrase.*** An indefinite noun phrase is a phrase that is used to introduce a specific object or set of objects believed to be new to the addressee (e.g., a new automobile, some sheep, and five accountants).
- ***Definite Noun Phrase.*** This is a noun phrase that starts with the article "the."
- ***Demonstrative Noun Phrase.*** This is a noun phrase that starts with "this," "that," "those," or "these."

Features of Each Pair of Markables

- Sentence Distance: 0 if the markables are in the same sentence.
- Pronoun: 1 if one of the markables is a pronoun; 0 otherwise.
- Exact Match: 1 if the two markables are identical; 0 otherwise.
- Definite Noun Phrase: 1 if one of the markables if a definite noun phrase; 0 otherwise.
- Demonstrative Noun Phrase: 1 if one of the markables is a demonstrative noun phrase.
- Number Agreement: 1 if the both markables are singular or plural; 0 otherwise.
- Semantic Agreement: 1 if the markables belong to the same semantic class (based on the entity extraction component).
- Gender Agreement: 1 if the two markables have the same gender (male, female), 0 if not, and 2 if it is unknown.
- Proper Name: 1 if both markables are proper names; 0 otherwise.
- Alias: 1 if one markable is an alias of the other entity (like GE and General Motors).

Generating Training Examples

- ***Positive Examples.*** Assume that in a given document we have found four markables that refer to the same real-world entity, {M1,M2,M3,M4}. For each adjacent pair of markables we will generate a positive example. In this case, we will have three positive examples – namely {M1,M2}, {M2,M3} and {M3,M4}.
- ***Negative Examples.*** Assume that markables a,b,c appear between M1 and M2; then, we generate three negative examples {a,M2}, {b,M2}, {c,M2}.

The Algorithm

```
Identify all markables
For each anaphor A
    Let M1 to Mn be all markables from the
beginning of the document till A
     For i=n;i=1;i - -
        if PairClassifier(A,Mi)=true then
              A, Mi is an anaphoric pair
              exit
           end if
        end for
     end for
```

Evaluation

Training on 30 documents yielded a classifier that was able to achieve precision of 68 percent and recall of 52 percent (F1 = 58.9%).

Ng and Cardie (Ng and Cardie 2002) have suggested two types of extensions to the Soon et al. corpus-based approach. First, they applied three extralinguistic modifications to the machine learning framework, which together provided substantial and statistically significant gains in coreference resolution precision. Second, they expanded the Soon et al. feature set from 12 features to an arguably deeper set of 53.

Ng and Cardie have also proposed additional lexical, semantic, and knowledge-based features – most notably, 26 additional grammatical features that include a variety of linguistic constraints and preferences. The main modifications that were suggested by Ng and Cardie are as follows:

- ***Best-first Clustering.*** Rather than a right-to-left search from each anaphoric NP for the first coreferent NP, a right-to-left search for a *highly likely antecedent* was performed. As a result, the coreference clustering algorithm was modified to select as the antecedent of NP the NP with the highest coreference likelihood value from among preceding NPs with coreference class values above 0.5.
- ***Training Set Creation.*** Rather than generate a positive training example for each anaphoric NP and its **closest** antecedent, a positive training example was generated for its **most confident** antecedent. For a nonpronominal NP, the closest **nonpronominal** preceding antecedent was selected as the most confident antecedent. For pronouns, the closest preceding antecedent was selected as the most confident antecedent.
- ***String Match Feature.*** Soon's string match feature (SOON STR) tests whether the two NPs under consideration are the same string after removing determiners from each. Rather than using the same string match for all types of anaphors, finer granularity is used. Exact string match is likely to be a better coreference predictor for proper names than it is for pronouns, for example. Specifically, the SOON STR feature is replaced by three features – PRO STR, PN STR, and WORDS STR – that restrict the application of string matching to pronouns, proper names, and nonpronominal NPs, respectively.

Overall, the learning framework and linguistic knowledge source modifications boost performance of Soon's learning-based coreference resolution approach from an F-measure of 62.6 to 70.4 on MUC-6 and from 60.4 to 63.4 on MUC-7.

VI.6 INDUCTIVE ALGORITHMS FOR IE

Rule Induction algorithms produce symbolic IE rules based on a corpus of annotated documents.

VI.6.1 WHISK

WHISK is a supervised learning algorithm that uses hand-tagged examples for learning information extraction rules. WHISK learns regular expressions for each of the fields it is trying to extract. The algorithm enables the integration of user-defined semantic classes. Such classes enable the system to adjust to the specific jargon of a given domain. As an example, consider the domain of apartment rental ads. We want to accommodate all types of spellings of bedroom, and hence we introduce the following semantic class: Bdrm = (brs |br |bds | bdrm | bd| bedroom| bed). WHISK learns the regular expressions by using an example-covering algorithm that tries to cover as many positive examples while not covering any negative example. The algorithm begins learning a single rule by starting with an empty rule; then we add one term at a time until either no negative examples are covered by the rule or the prepruning criterion has been satisfied. Each time we add the term that minimizes the Laplacian, which is $(e + 1)/(n + 1)$, where e is the number of negative examples covered by the rule as a result of the addition of the term and n is the number of positive examples covered by the rule as a result of the term addition. The process of adding rules repeats until the set of learned rules covers all the positive training instances. Finally, postpruning removes some of the rules to prevent overfitting.

Here is an example of a WHISK rule:

ID::1
Input:: * (Digit) 'BR' * '$' (number)
Output:: Rental {Bedrooms $1} {Price $2}

For instance, from the text "3 BR, upper flr of turn of ctry. Incl gar, grt N. Hill loc 995$. (206)-999-9999," the rule would extract the frame Bedrooms – 3, Price – 995.

The "*" char in the pattern will match any number of characters (unlimited jump). Patterns enclosed in parentheses become numbered elements in the output pattern, and hence (Digit) is $1 and (number) is $2.

VI.6.2 BWI

The BWI (boosted wrapper induction) is a system that utilizes wrapper induction techniques for traditional Information Extraction. IE is treated as a classification problem that entails trying to approximate two boundary functions $X_{\text{begin}}(i)$ and $X_{\text{end}}(i)$. $X_{\text{begin}}(i)$ is equal to 1 if the ith token starts a field that is part of the frame to be extracted and 0 otherwise. $X_{\text{end}}(i)$ is defined in a similar way for tokens that

end a field. The learning algorithm approximates each X function by taking a set of pairs of the form $\{i, X\}(i)$ as training data. Each field is extracted by a wrapper $W = <F, A, H>$ where

F is a set of begin boundary detectors
A is a set of end boundary detectors
$H(k)$ is the probability that the field has length k

A boundary detector is just a sequence of tokens with wild cards (some kind of a regular expression).

$$W(i, j) = \begin{cases} 1 & \text{if} \quad F(i)A(j)H(j - i + 1) > \sigma \\ 0 & \text{otherwise} \end{cases}$$

$$F(i) = \sum_k C_{F_k} F_k(i), \quad A(i) = \sum_k C_{A_k} A_k(i).$$

$W(i, j)$ is a nave Bayesian approximation of the probability that we have a field between token i and j with uniform priors. Clearly, as σ is set to be higher we get better precision and lower recall, and if we set σ to be 0 we get the highest recall but compromise precision.

The BWI algorithm learns two detectors by using a greedy algorithm that extends the prefix and suffix patterns while there is an improvement in the accuracy. The sets $F(i)$ and $A(i)$ are generated from the detectors by using the AdaBoost algorithm. The detector pattern can include specific words and regular expressions that work on a set of wildcards such as <num>, <Cap>, <LowerCase>, <Punctuation> and <Alpha>.

When the BWI algorithm was evaluated on the acquisition relations from the Reuters news collection, it achieved the following results compared with HMM:

Slot	BWI	HMM
Acquiring Company	34.1%	30.9%
Dollar Amount	50.9%	55.5%

VI.6.3 The (LP)^2 Algorithm

The (LP)^2 algorithm learns from an annotated corpus and induces two sets of rules: tagging rules generated by a bottom-up generalization process and correction rules that correct mistakes and omissions done by the tagging rules.

A tagging rule is a pattern that contains conditions on words preceding the place where a tag is to be inserted and conditions on the words that follow the tag. Conditions can be either words, lemmas, lexical categories (such as digit, noun, verb, etc), case (lower or upper), and semantic categories (such as time-id, cities, etc).

The $(LP)^2$ algorithm is a covering algorithm that tries to cover all training examples. The initial tagging rules are generalized by dropping conditions.

A sample rule for tagging the stime (start time of a seminar) is shown below.

	Condition					
Word Index	**word**	**lemma**	**LexCat**	**Case**	**SemCat**	**Tag Inserted**
3		at				<stime>
4			digit			
5					time-id	

The correction rules take care of incorrect boundaries set for the tags and shift them to fix the errors. An example is "at <stime> 4 </stime> pm," where the </stime> tag should be shifted one token to the right. The correction rules learn from the mistakes of the tagging processing on the training corpus. The action taken by a correction rule is just to shift the tag rather than introduce a new tag. The same covering algorithm used for learning the tagging rules is used for learning the correction rules.

$(LP)^2$ was also tested on extracting information from financial news articles and managed to obtain the following results:

Tag	**F1**	**Tag**	**F1**
Location	70%	Organization	86%
Currency	85%	Stock Name	85%
Stock Exchange Name	91%	Stock Category	86%
Stock Exchange Index	97%	Stock Type	92%

These results are not on par with the results achieved by the probabilistic extraction algorithms such as HMM, CRF, and MEMM. It seems that the inductive algorithms are suitable for semistructured domains, where the rules are fairly simple, whereas when dealing with free text documents (such as news articles) the probabilistic algorithms perform much better.

VI.6.4 Experimental Evaluation

All four algorithms were evaluated on the CMU seminar announcement database and achieved the following results (F1results):

Slot	**BWI**	**HMM**	**$(LP)^2$**	**WHISK**
Speaker	67.7%	76.6%	77.6	18.3%
Location	76.7%	78.6%	75.0%	66.4%
Start Time	99.6%	98.5%	99.0%	92.6%
End Time	93.9%	62.1%	95.5%	86%

VI.7 STRUCTURAL IE

VI.7.1 Introduction to Structural IE

Most text mining systems simplify the structure of the documents they process by ignoring much of the structural or visual characteristics of the text (e.g., font type, size, location, etc.) and process the text either as a linear sequence or as a bag of words. This allows the algorithms to focus on the semantic aspects of the document. However, valuable information is lost in these approaches, which ignore information contained in the visual elements of a document.

Consider, for example, an article in a journal. The title is readily identifiable based on its special font and location but less so based on its semantic content alone, which may be similar to the section headings. This holds true in the same way for the author names, section headings, running title,and so on. Thus, much important information is provided by the visual layout of the document – information that is ignored by most text mining and other document analysis systems.

One can, however, leverage preprocessing techniques that do not focus on the semantic content of the text but instead on the visual layout alone in an effort to extract the information contained in layout elements. These type of techniques entail an IE task in which one is provided a document and seeks to discover specific fields of the document (e.g., the title, author names, publication date, figure captions, bibliographical citings, etc.). Such techniques have been termed *structural* or *visual information extraction*.

Of course, it goes without saying that, within the overall context of text mining preprocessing, a structural or visual IE approach is not aimed at replacing the semantic one. Instead, the structural IE approach can be used to complement other more conventional text mining preprocessing processes.

This section describes a recently developed general algorithm that allows the IE task to be performed based on the visual layout of the document. The algorithm employs a machine learning approach whereby the system is first provided with a set of training documents in which the desired fields are manually tagged. On the basis of these training examples, the system automatically learns how to find the corresponding fields in future documents.

VI.7.2 Overall Problem Definition

A document D is a set of primitive elements $D = \{e_1, \ldots, e_n\}$. A primitive element can be a character, a line, or any other visual object as determined by the document format. A primitive element can have any number of visual attributes such as font size and type, physical location, and so on. The *bounding box* attribute, which provides the size and location of the bounding box of the element, is assumed to be available for all primitive elements. We define an *object* in the document to be any set of primitive elements.

The *visual information extraction (VIE)* task is as follows. We are provided with a set of *target fields* $F = \{f_1, \ldots, f_k\}$ to be extracted and a set of *training documents* $T = \{T_1, \ldots, T_m\}$ wherein all occurrences of the target fields are annotated. Specifically, for each target field f and training document T, we are provided with the object

$f(T)$ of T that is of type f ($f(T) = 0$ if f does not appear in T). The goal, when presented with an unannotated query document Q, is to annotate the occurrences of target fields that exist in Q (not all target fields need be present in each document).

Practically, the VIE task can be decomposed into two subtasks. First, for each document (both training and query) one must group the primitive elements into meaningful objects (e.g., lines, paragraphs, etc.) and establish the hierarchical structure among these objects. Then, in the second stage, the structure of the query document is compared with the structures of the training documents to find the objects corresponding to the target fields.

It has also proven possible to enhance the results by introducing the notion of *templates*, which are groups of training documents with a similar layout (e.g., articles from the same journal). Using templates, one can identify the essential features of the page layout, ignoring particularities of any specific document. Templates are discussed in detail in the sections that follow.

A brief examination is also made of a real-world system that was implemented for a typical VIE task involving a set of documents containing financial analyst reports. The documents were in PDF format. Target fields included the title, authors, publication dates, and others.

VI.7.3 The Visual Elements Perceptual Grouping Subtask

Recall that a document is a set of primitive elements such as characters, figures, and so on. The *objects* of a document are sets of primitive elements. Target fields, in general, are objects.

Thus, the first step in the visual IE task is to group the primitive elements of the documents into higher level objects. The grouping should provide the conceptually meaningful objects of the document such as paragraphs, headings, and footnotes.

For humans, the grouping process is easy and is generally performed unconsciously based on the visual structure of the document. As with other types of IE perceptual grouping requirements, the goal is to mimic the human perceptual grouping process.

VI.7.4 Problem Formulation for the Perceptual Grouping Subtask

One can model the structure of the objects of a document as a tree, of which the leaves are primitive elements and the internal nodes are (composite) objects. This structure is called the object tree or *O-Tree* of the document.

The O-Tree structure creates a hierarchal structure among objects in which higher level objects consist of groups of lower level objects. This hierarchal structure reflects the conceptual structure of documents in which objects such as columns are groups of paragraphs, which, in turn, are groups of lines, and so on. The exact levels and objects represented in the O-Tree are application and format dependent.

For an HTML document, for example, the O-Tree may include objects representing tables, menus, the text body, and other elements, whereas for PDF documents the O-Tree may include objects representing paragraphs, columns, lines, and so on. Accordingly, for each file format and application we define the *object hierarchy, H*,

which determines the set of possible *object types*, and a hierarchy among these objects. Any object hierarchy must contain an object of type document, which is at the root of the hierarchy. When constructing an O-Tree for a document, each object is labeled by one of the *object types* defined in the object hierarchy, and the tree structure must correspond to the hierarchical structure defined in the hierarchy.

Formally, an object hierarchy H is a rooted DAG that satisfies the following:

- The leaf nodes are labeled by primitive element types.
- Internal nodes are labeled by objects types.
- The root node is labeled by the document object type.
- For object types x and y, type y is a child of x if an object of type x can (directly) contain an object type y.

For a document $D = \{e_1, \ldots, e_n\}$ and an object hierarchy H, an *O-Tree of D* according to H is a tree O with the following characteristics:

- The leaves of O consist of all primitive elements of D.
- Internal nodes of O are objects of D.
- If X and X' are nodes of O (objects or primitive elements) and $X \subset X'$, then X' is an ancestor (or parent) of X.
- Each node X is labeled by a label from H denoted *label*(X).
- If X' is a parent of X in T, then *label*(X') is a parent of *label*(X) in H.
- *label*(*root*) = Document.

VI.7.5 Algorithm for Constructing a Document O-Tree

Given a document, one constructs an O-Tree for the document. In doing so, the aim is to construct objects best reflecting the true grouping of the elements into "meaningful" objects (e.g., paragraphs, columns, etc.). When constructing an object we see to it that the following requirements are met:

- The elements of the objects are within the same physical area of the page. Specifically, each object must be *connected* – that is, an object X cannot be decomposed into two separate objects $X1$ and $X2$ such that any line connecting $X1$ and $X2$ necessarily crosses an element in a different object.
- The elements of the object have similar characteristics (e.g., similar font type, similar font size, etc.). Specifically, one must assume a *fitness function fit*(.,.) such that for any two objects X and Y, where *label*(Y) is child of *label*(X), *fit*(Y,X) provides a measure of how fit Y is as an additional member to X (e.g., if X is a paragraph and Y a line, then how similar is Y the other lines in X). One adds Y to X only if *fit*(Y;X) is above some threshold value, $\circ$. The exact nature of the function *fit*(ϕ; ϕ) and the threshold value are format and domain dependent.

Given these two criteria, the O-Tree can be constructed in a greedy fashion, from the bottom up, layer by layer. In doing so, one should always prefer to enlarge existing objects of the layer, starting with the largest object. If no existing object can be enlarged, and there are still "free" objects of the previous layer, a new object is created. The procedure terminates when the root object, labeled Document, is

completed. A description of the algorithm is provided in the following pseudocode algorithm:

Input: D - Document
Output: O-Tree for D

1. For each type $t \in H$ do
   let *level(t)* be the length of the longest path from t to a leaf
2. Let h = *level*(Document)
3. *Objects*(0) $\leftarrow D$
4. For i = 1 to h do
5. *Objects(i)* $\leftarrow$ 0
6. *free* $\leftarrow$ *Objects(i* $-$ 1)
7. *While free* $\neq$ 0 do
8. For each X $\in$ *Objects(i)* in order of descending size do
9. For each $Y \in$ *free* in order of increasing distance from X do
10. If Y is a neighbor of X and *fit(Y, X)* $\geq \gamma$ then
11. $X \leftarrow X \cup Y$
12. *make Y a child of X*
13. Remove Y from *free*
14. Break (go to line 7)
15. For each $t \in H$ such that *level(t)* = i do
16. if *Objects(i)* does not include an empty object of type t
17. Add an empty set of type t to *Objects(i)*
18. end while
19. Remove empty objects from *Objects*(i)
20. end for
21. return resulting O-Tree

VI.7.6 Structural Mapping

Given a Visual Information Extraction task, one first constructs an O-Tree for each of the training documents as well as for the query document, as described in the previous section. Once all the documents have been structured as O-Trees, it is necessary to find the objects of Q (the query document) that correspond to the target fields. This is done by comparing the O-Tree of Q, and the objects therein, to those of the training documents.

This comparison is performed in two stages. First, the training document that is visually most similar to the query document is found. Then, one maps between the objects of the two documents to discover the targets fields in the query document.

VI.7.6.1 Basic Algorithm

- ***Document Similarity.*** Consider a query document Q and training documents $T = \{T_1, \ldots, T_n\}$. We seek to find the training document T_{opt} that is *visually*

most similar to the query document. We do so by comparing the O-Trees of the documents. In the comparison we only concentrate on similarities between the top levels of the O-Tree. The reason is that even similar documents may still differ in the details.

Let $O(Q)$ and $O(T_1), \ldots, O(T_n)$ be the O-Trees, the query document and the training documents, respectively, and let H be the type hierarchy. We define a subgraph of H, which we call the *Signature Hierarchy*, consisting of the types in H that determine features of the global layout of the page (e.g., columns, tables). The types that are included in the signature is implementation dependent, but generally the signature would include the top one or two levels of the type hierarchy. For determining the similarity between the objects we assume the existence of a similarity function $sim(X,Y)$, which provides a measure of similarity between objects of the same type based on object characteristics such as size, location, and fonts ($sim(X,Y)$ is implementation dependent).

Given a query document Q and a training document T, for each object X in the signature of T we find the object $X0$ of Q (of the same type as X) that is most similar to X. We then compute the average similarity for all objects in the signature of T to obtain the overall similarity score between the documents. We choose the document with the highest similarity score. A description of the procedure is provided below.

Input: Q, $T_1 \ldots, T_n$., and their respective O-Trees
Output: T_{opt} (training document most similar to query document)
1 for i = 1 to n do
1. $total \leftarrow 0$
2. $count \leftarrow 0$
3. For each $t \in S$ do
4. For each $X \in O(Ti)$ of type t do
5. $s(X) \leftarrow \max \{sim(X,X') \mid X' \in O(Q), X' \text{ of type } t\}$
6. $total \leftarrow total + s(X)$
7. $count \leftarrow count + 1$
8. $score\ (i) \leftarrow total = count$
9. end for
10. $opt \leftarrow \text{argmax}\{score\ (i)\ g\}$
11. return T_{opt}

■ ***Finding the Target Fields.*** Once the most similar training document T_{opt} has been determined, it remains to find the objects of Q that correspond to the target fields, as annotated in the document T_{opt}. One does so by finding, for each target field f, the object within $O(Q)$ that is most similar to $f(T_{opt})$ (the object in $O(T_{opt})$ annotated as f). Finding this object is done in an exhaustive manner, going over all objects of $O(Q)$. One also makes sure that the similarity between this object and the corresponding object of T_{opt} is beyond a certain threshold α, or else one decides that the field f has not been found in Q (either because it is not there or we have failed to find it).

A description of the procedure for finding the target fields is provided in the algorithm below. Note that the annotation of the training documents is performed before (and independent) of the construction of the O-Trees. Thus, annotated objects need not appear in the O-Tree. If this is the case, line 2 sees to it that one takes the minimal object of the $O(T_{opt})$ that fully contains the annotated object.

Input:
- Q, T_{opt} (and their respective O-Trees)
- $\{f(T_{opt}) \mid f \in F\}$(target fields in T_{opt})

Output: $\{f(Q) \mid f \in F\}$(Target fields in Q)

1 *For each* $f \in F$ do

1. Let $f(T_{opt}) \in O(T_{opt})$ be minimal such that $f(T_{opt}) \subseteq {}^{-}f(T_{opt})$
2. $f(Q) \leftarrow \text{argmax}\{sim(f(T_{opt}), X) \mid X \in O(Q); X \text{ of type } t\}$
3. if $sim({}^{-}f(T_{opt}), f(Q)) < \alpha$ then $f(Q) \leftarrow 0$

VI.7.7 Templates

The preceding algorithm is based on finding the single most similar document to the query document and then extracting all the target fields based on this document alone. Although this provides good results in most cases, there is the danger that particularities of any single document may reduce the effectiveness of the algorithm.

To overcome this problem, the notion of *templates* has been introduced; these templates permit comparison of the query document with a *collection* of similar documents. A *template* is a set of training documents that have the same general visual layout (e.g., articles from the same journal, Web pages from the same site, etc.). The documents in each template may be different in details but share the same overall structure.

Using templates, one finds the template most similar to the query document (rather than the document most similar). This is accomplished by – for each template – averaging the similarity scores between the query document and all documents in the template.

One then picks the template with the highest average similarity. Once the most similar template is determined, the target fields are provided by finding the object of Q most similar to a target field in *any* of the documents in the template. A description of the process by which one can find target fields through the use of templates is provided below.

Input:
- Q and its O-Tree
- $T_{opt} = \{T_1, \ldots, T_k.\}$(most similar template) and respective O-Trees
- $\{f(T) \mid f \in F, T \in T_{opt}\}$(target fields in T_{opt})

Output: $\{f(Q) \mid f \in F\}$(Target fields in Q)

1 For each $f \in F$ do

1. For each $T \in T_{opt}$ do

2. Let $^{-}f(T) \in O(T)$ be minimal such that $f(T) \subseteq {}^{-}f(T)$
3. $X_T \leftarrow (\text{argmax}\{sim(^{-}f(T), X) \mid X \in O(Q), X \text{ of type } t\}$
4. $s(T) \leftarrow sim(^{-}f(T), X_T)$
5. if $\max\{s(T) \mid T \in T_{opt}\} \geq \alpha$ then
6. $f(Q) \leftarrow (\text{argmax}\{s(X_T) \mid T \in T_{opt}\}$
7. else $f(Q) \leftarrow 0$

VI.7.8 Experimental Results

A system for Visual Information Extraction, as described above, was recently implemented on documents that are analyst reports from several leading investment banks. The training data consisted of 130 analyst reports from leading investment banks: 38 from Bear Stearns, 14 from CSFB, 15 from Dresdner, and 63 from Morgan Stanley. All the documents were in PDF format. The training documents were divided into a total of 30 templates: 7 from the Bear Stearns data, 4 from the CSFB data, 5 from the Dresdner data, and 14 in the Morgan Stanley data. All the training documents were manually annotated for the target fields. The target fields included the following fields: Author, Title, Subtitle, Company, Ticker, Exchange, Date, Geography, Industry Info. Not all documents included all target fields, but within each template documents had the same target fields.

The system was tested on 255 query documents: 33 from Bear Stearns, 12 from CSFB, 14 from Dresdner, and 196 from Morgan Stanley. With regard to the implementation, the type hierarchy (H) used in the system is provided in Figure VI.7. The signature hierarchy contained the objects of type column and paragraph. The implementation of the fitness function *fit*(.,.) (for the fitness of one object within the other) takes into account the distance between the objects and the similarity of font type.

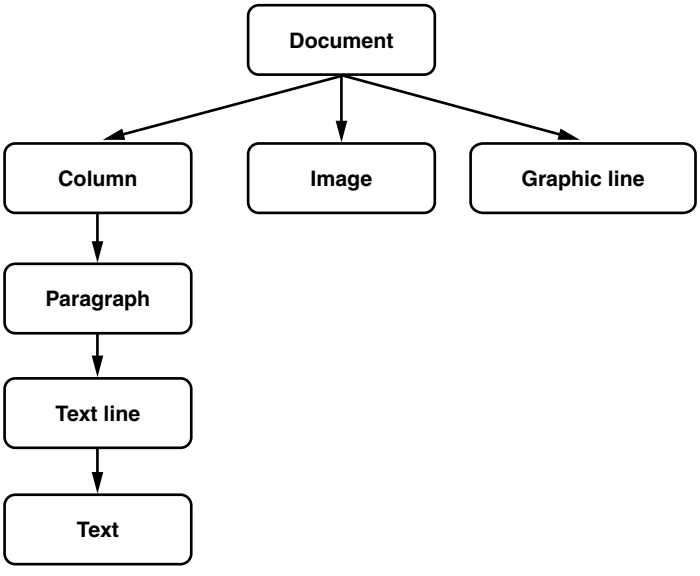

Figure VI.7. The type hierarchy.

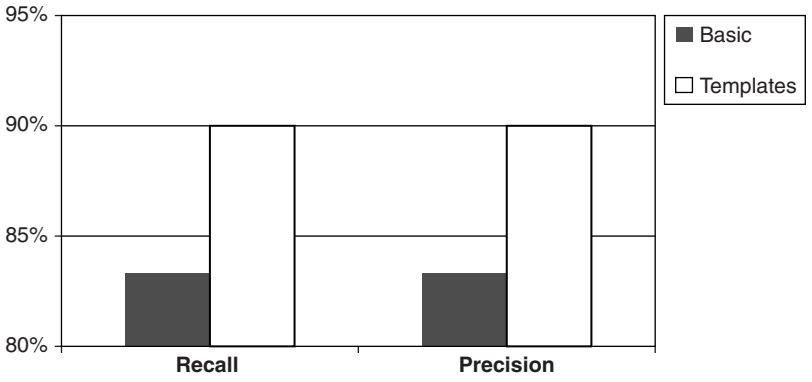

Figure VI.8. Overall recall and precision rates for basic algorithm and for algorithm using templates.

For the fitness of a line within an existing paragraph, it also takes into account the distance between lines. The similarity function *sim*(.,.), which measures the similarity between objects in different documents, is primarily based on similarity between the sizes and locations of the respective objects.

With an emphasis on the results, the performance of the system was measured with the basic algorithm and the use of templates. The overall average recall and precision values are given in Figure VI.8.

On the whole the introduction of templates improved the performance of the algorithm, increasing the average accuracy from 83 to 90 percent. Note that for both algorithms the recall and precision values are essentially identical. The reason for this is that, for any target field *f*, on the one hand, each document contains only one object of type *f*, and, on the other hand, the algorithm marks one object as being of type *f*. Thus, for every recall error there is a corresponding precision error. The slight difference that does exist between the recall and precision marks is due to the cases in which the algorithm decided not to mark any element, signifying a recall error but not a precision error.

Some target fields are harder to detect than others. It is interesting to note that, although the introduction of templates improves accuracy in most cases, there are some target fields for which it reduces accuracy. Understanding the exact reasons for this and how to overcome such problems is a topic for further research.

VI.8 FURTHER READING

Section VI.1

For a full list of definitions related to information extraction, see <http://www.itl.nist.gov/iaui/894.02/related˙projects/muc/info/definitions.html>.

Section VI.4

Descriptions of rule-based information extraction systems can be found in Hobbs et al. (1991); Appelt, Hobbs, Bear, Israel, and Tyson (1993); Grishman (1996); Freitag (1997); Grishman (1997); Wilks (1997); and Ciravegna et al. (1999).

Section VI.5

Algorithms on anaphora resolutions can be found in Rich and LuperFoy (1988); Lappin and Leass (1994); McCarthy and Lehnert (1995); Humphreys, Gaizauskas, and Azzam (1997); Kehler (1997); Kennedy and Boguraev (1997); Barbu and Mitkov (2001); Klebanov and Wiemer-Hastings (2002); Ng and Cardie (2002); and Ng and Cardie (2003).

Discussions about evaluation of anaphora resolution algorithms can be found in Aone and Bennett (1995) and Azzam, Humphreys, and Gaizauskas (1998).

Section VI.6

More details about the whisk algorithm can be found in Soderland (1999). The description of the BWI algorithm can be found in Freitag and Kushmerick (2000). The $(LP)^2$ algorithm is described in Ciravegna (2001).