2 Information Extraction from an Historical Perspective

With Rik De Busser

2.1 Introduction

This chapter presents an historical overview of information extraction spanning more than three decades. It explains also the evolution from systems that use symbolic, handcrafted knowledge towards systems that train from labeled and eventually unlabeled examples. The historical overview allows us also to introduce the most common information extraction tasks and the common architecture of an extraction system. The most important algorithms will be discussed in detail in Chap. 3, which focuses on systems that use symbolic, handcrafted knowledge, and in Chaps. 4, 5 and 6, which discuss the machine learning approaches.

2.2 An Historical Overview

2.2.1 Early Origins

At the end of the sixties, Roger C. Schank introduced a revolutionary model to parse natural language texts into formal semantic representations (Schank, 1972; Shank, 1975)[1] and very soon his Conceptual Dependency

[1] An exposition of an early version of his theory can be found Shank (1972). The basic principles of CD theory as it is known among computational linguists today are explained in Schank (1975).

Theory (CDT) gained an enormous popularity. Schank's basic assumption is that "there exists a conceptual base that is interlingual, onto which linguistic structures in a given language map during the understanding process and out of which such structures are created during generation" (Schank, 1972, p. 553 ff.). This implies that any two linguistic structures that have the same meaning ought to be represented by identical conceptual structures (even when they are of a different language). These conceptual structures or *conceptualizations* – as Schank called them – are composed of primary concepts, the interconnections of which are governed by a closed set of universal *conceptual syntax rules* and a larger set of concept specific *conceptual semantic rules*. For the first time, a more or less comprehensive model had been developed that not only made it possible to semantically analyze entire texts, but that was also fit for practical implementation into artificial intelligence systems for the extraction of semantic information, something that would later be called information extraction.

In its infancy, Conceptual Dependency Theory mainly aimed to extract semantic information about individual events from sentences at a conceptual level. The main categories of concepts are PPs (i.e., picture producers, in other words, concrete nouns) and actions. Relations between concepts are dependencies. The main conceptualization of a clause is a two-way dependency between a PP (the actor) and an action. Schank defined natural language words in terms of conceptual primitives or predicates. The syntax of the conceptual level is described by a set of rules which specify which type of concepts can depend on which other type, as well as the different kinds of dependency relationships between concepts. Soon after its design, the theory developed into a fairly comprehensive system to embed these event analyses into full scenarios. Schank's theory had and still has an enormous influence on information extraction technologies.

Schank's unconventional theory mothered a research group at Yale University in the late 1970s and early 1980s, which developed several prototypes of information extraction tools that were able to extract information from texts with a previously unseen accuracy – be it in a very limited domain.

One of the early systems developed by the Yale School, called SAM, parsed a text into its full CD structure. In a first stage, a natural language analyzer maps the input text into conceptual dependency structures sentence by sentence, while filling out all implicit classes of concepts by using expectation routines. Using the output of the language analyzer, a second module – the script applier – tries to understand the story offered by the text. It matches a given input with a script in its database and it uses the script to predict which information is likely to follow in the input string and to fill out any conceptualizations left implicit. The script applier is assisted by a conceptual parser, which parses the input into individual conceptualizations and disambiguates word senses, and a memory module, which maps information about physical objects onto references to a real world model. At the end of the analysis, the script applier outputs a fully instantiated conceptual dependency network of the text, which can be accessed by the post-processing modules. Schank and Abelson (1977) mention a summarizer and a questioning-answering module.

Conceptual Dependency Theory has been implemented in numerous other applications, most of which were not developed beyond the test stage. The majority of these systems did not use fully developed CD scripts (which are supposed to contain all information about any event which could possibly occur in a given situation), but so-called *sketchy scripts*, which only contain the most crucial or most relevant conceptualizations. This approach allows that a text is only partially analyzed, while the rest is ignored or skipped, or in other words that only certain information is extracted. Such an analysis is commonly referred to as *partial parsing*.

One of the most typical systems using sketchy scripts is undoubtedly FRUMP. The *Fast Reading Understanding and Memory Program* was developed at Yale University for skimming newspaper articles (DeJong, 1977). At the time of writing of DeJong's 1982 article, sketchy scripts for the interpretation modules of FRUMP had been manually constructed for 60 different situations. The interpretation of texts takes place in two modules, which DeJong calls the predictor and the substantiator. On the basis of the current context, the predictor predicts which conceptualizations or parts of conceptualizations are likely to follow in the partly analyzed text and passes its results on to the substantiator. The latter will try to fill out the predicted structures; either by finding a word or phrase from the input text matching a slot filler specification proposed in one of the predictions; or by drawing an inference based on the input text and the relevant CD structure proposed by the predictor. When the substantiator succeeds in verifying one of the predictions, the predictor adds it to the current context. If none of the predictions can be verified, the predictor backtracks and makes new suggestions. In this way the entire text is processed sentence by sentence. Since FRUMP's routines are expectation driven – as all algorithms based on conceptual dependency are – the system needs to be initialized: At the start of the analysis it has not built up a context on which to base its predictions yet. It somehow has to be able to activate one or more relevant scripts to allow the substantiator to create an initial context or when the substantiator does not succeed in verifying the predictions.

Therefore, FRUMP provides activation routines for sketchy scripts. A script is triggered by words or phrases in the text, by events already detected in the texts and by a related script that is already active. After processing the text, most – but not all – empty slots in the script will be filled. FRUMP stores the partly instantiated script containing the information about the article and when encountering a new article related to the same situation, it will use it to update the partly instantiated script.

Computer understanding of human narratives is a classic challenge for natural language processing and artificial intelligence. The understanding regards the extraction of the chronology and the plot. Rumelhart (1975 1977) has proposed the idea of so-called story grammars for understanding and summarizing text. He analyzed stories into hierarchical structures. The system of Lehnert (1982) built a plot unit connectivity graph for narrative text. Plot units have the form of propositions and are composed of affect states (e.g., positive events, negative events, mental states) that are linked by four types of relations (motivation, actualization, termination and equivalence). The recognition of affect states is based on a large predictive knowledge base containing knowledge about plans, goals and themes. The analysis of the story in terms of plot units results in a complex network where some units are subordinated to others. These graph type representations of a story structure still have their influence today (Mani, 2003).

Interesting also to mention is The Linguistic String Project (LSP) at New York University, which began in 1965 with funding from the National Science Foundation to develop computer methods for structuring and accessing information in the scientific and technical literature. Document processing was to be based on linguistic principles, first to demonstrate the possibility of computerized grammatical analysis (parsing), then to extend to specialized vocabulary and rules for particular scientific domains. Domain specialization led to an elaboration of the methods of sublanguage analysis (Sager, 1981), in particular as applied to the language of clinical reporting in patient documents and to the extraction of information. The project still has an influence on medical language processing.

2.2.2 Frame Theory

Schank's research inspired many scientists. Though his theory was rarely implemented in its pure form, it boosted research into frame based methods for knowledge representation. The notion of a frame system was explicitly formulated for the first time by Minsky in his groundbreaking paper on knowledge representation structures for artificial intelligence. Minsky defined a frame as follows:

A frame is a data structure for representing a stereotyped situation, like being in a certain type of living room or being at a child's birthday party.

(Minsky, 1975, p. 212)

A frame stores the properties of characteristics of an entity, action or event. It typically consists of a number of slots to refer to the properties named by a frame, each of which contains a value (or is left blank). The number and type of slots will be chosen according to the particular knowledge to be represented. A slot may contain a reference to another frame. Other features of frames have advantages: They include the provision of a default value for a particular slot in all frames of a certain type, the use of more complex methods for inheriting values and properties between frames, and the use of procedural attachments to the frame slots. When frames have mutual relationships, a *semantic net of frames* can represent them.

Minsky's frames pervaded AI research in the second half of the 1970s. They would remain the major data representation structures for information extraction applications up to the late 1990s. Instantiated conceptual frames are often stored in a semantic network that can afterwards be accessed by the questioning answering module.

In the 1980s information extraction became a hot topic. A multitude of algorithms were designed that were influenced by Schank's Conceptual Dependency theory, many of which used frames. Famous systems are TESS that analyzes bank telexes (Young and Hayes, 1985), SCISOR that used to process news articles on corporate mergers and acquisitions from the online information service of the Dow Jones (Jacobs and Rau, 1990), CONSTRUE (Hayes and Weinstein, 1991) and FASTUS (Appelt, Hobbs, Bear, Israel and Tyson, 1993). A nice overview of other systems developed at the end of the 1980s and 1990s is given by Hahn (1989) and Jacobs (1992). Some of the systems first parse the text into its syntactical structure before instantiating the frames and mapping the information to the frame slots. An example is FASTUS, which will be discussed extensively in Chap. 3.

From their introduction in the 1970s on, frames managed to preserve their position in the world of knowledge representation with a remarkable tenacity. As we will further see in this chapter the current FrameNet project (Baker et al., 1998; Fillmore and Baker, 2001) is creating an on line lexical resource for English, based on frame semantics (Fillmore, 1968). An important component of the resource is a Frame database containing descriptions for lexical units for English. The descriptions include the conceptual structure of frames that represents a lexical item and descriptions for the elements (semantic roles), which participate in such structure. The FrameNet database is available in XML format and translated into a DAML + OIL (DARPA Agent Markup Language + Ontology Interface Layer) knowledge representation.

2.2.3 Use of Resources

The many practical information extraction systems that were developed in the 1980s and 1990s showed the need for a number of additional resources when processing texts.

First of all, we need a module for tokenization and sentence segmentation (Palmer, 2000). Tokenization breaks a text into tokens or words. It distinguishes words, components of multi-part words (e.g., splitting the Dutch term **onroerendgoedmarkt** in **markt van onroerend goed,** meaning market of real estate) and multiword expressions (e.g., **in spite of**). In space delimited languages (such as most European languages) a word or token can be defined as a string of characters separated by white space. In unsegmented languages (such as Chinese, Thai and Japanese), you need additional lexical and morphological information that can be found in a word list in the form of a machine-readable dictionary. Alternatively, statistical techniques can be used to learn which characters are most likely to form words based on co-occurrence statistics (e.g., use of the mutual information statistic, chi-square statistic, etc.). During lexical analysis a text is usually split into sentences.

Usually language dependent rules are incorporated, for instance, for the resolution of apostrophes or hyphens. For most languages and texts, punctuation reliably indicates sentence boundaries. However, punctuation marks are sometimes ambiguous. A period, for example, can denote a decimal point or a thousands marker, an abbreviation, the end of a sentence, or even an abbreviation at the end of a sentence.

On overall, normalization is considered to be a necessary processing step in any application that involves textual data. It comprises harmonizing spelling and capitalization and cleaning up unnecessary metadata. For some applications, stemming or lemmatization (i.e., words are restored to respectively their root or dictionary form) can be useful.[2]

Another step is the enrichment of the textual data with linguistic metadata that will be used as features in the extraction process. To this end, a

[2] Splitters are often incorporated in stemmers as affixes sometimes have to be removed.

number of *natural language processing* tools can be used. For most applications, they include *part-of-speech* (POS) *tagging* (i.e., detecting the syntactic word class such as noun, verb, etc.) and *phrase chunking* (i.e., detecting base noun and verb phrases). Syntactic structure is often indicative of the information distribution in a sentence. For many applications, a rudimentary syntactic analysis is sufficient, which is often referred to as *shallow parsing*. Shallow parsing aims to recover fragments of syntactic structures as efficiently and as correctly as possible. It can be implemented in different ways. For example, phrasal analysis can be accomplished by bracketing the output of a part-of-speech tagger (see Church, 1988). In some cases additional *parsing* (i.e., breaking up a sentence into its constituents and building the dependency tree of a sentence), or even full parsing might be desirable. *Full parsing* aims at providing an analysis of the sentence structure as detailed as it is possible. This might include the translation into a canonical structure (e.g., argument structure) in which processes (e.g., as expressed by verbs) of sentences and their arguments are delimited. Sometimes, sentence constituents are classified (e.g., into subject, object, semantic roles).

For a more complete overview of natural language processing in general, we refer the reader to Allen (1995).

A treebank can also be a useful resource. A *treebank* can be defined as a syntactically processed corpus that contains annotations of natural language data at various linguistic levels: Often at word, phrase, clause and sentence levels. A treebank provides mainly the morphosyntactic and syntactic structure of the utterances within the corpus and consists of a *bank of linguistic trees*, thereby its name. The type of annotations differs, however, between treebanks. The descriptions can be based on various linguistic theories, such as a dependency grammar (e.g., the Prague Dependency Treebank for Czech, the Turin University Treebank for Italian, the Turkish treebank METU) or a Head-driven Phrase Structure Grammar (e.g., HPSG-based Syntactic Treebank of Bulgarian, Polish and Verbmobil HPSG Treebanks). One of the most well known treebanks is the Penn treebank. Currently, there are also on going treebank projects for several languages such as for Chinese, Dutch, French, Portuguese, Spanish, Turkish, etc.

Additional lexical resources in machine readable form that offer knowledge of synonymy, hypernymy, hyponymy and meronymy are valuable. *Synonymy* involves the use of different lexical items that express the same or a closely related word sense (e.g., **sound** and **noise**). Strict synonymy almost never occurs, since word forms describing the same concept tend to differentiate their meanings. For instance, **sound** and **noise** refer to the same referent, but they have a different meaning: **Noise** has a slightly negative connotation (e.g., an obnoxious sound) whereas the meaning of

sound is neutral. *Hypernymy* regards describing a term with a more general term (e.g., **tree** is a hypernym of **oak)**. *Hyponymy* describes a term with a more specific term (e.g., **apple** is a hyponym of **fruit)**. *Meronyms* are related through a part-whole relation (e.g., **leg** is a part of **body**). The relations here discussed can be found in a lexico-semantic resource such as WordNet for English (Miller 1990).

Other lexico-semantic resources such as *FrameNet* are valuable. As mentioned above, the Berkeley FrameNet project is creating an on line lexical resource for English, based on frame semantics and supported by corpus evidence (Baker et al., 1998; Fillmore and Baker, 2001). The aim is to document the range of semantic and syntactic combinatory possibilities (valences) of each word in each of its senses, through computer assisted annotation of example sentences and automatic tabulation and display of the annotation results. The major product of this work, the FrameNet lexical database, currently contains more than 8,900 lexical units, more than 6,100 of which are fully annotated, in more than 625 semantic frames, exemplified in more than 135,000 annotated sentences. FrameNet documents the manner in which frame elements (for given words in given meanings) are grammatically instantiated in English sentences based on attested instances of contemporary English and organizes the results of such findings in a systematic way. The FrameNet database can be seen both as a dictionary and as a thesaurus. The former signals, for instance, the definition of a lexical item and gives access to annotated examples illustrating each syntactic pattern found in the corpus and the kinds of semantic information instanced with such patterns. The database acts also as a thesaurus, in that, by being linked to frames, each word is directly connected with other words in its frame(s), and further extensions are provided by working out the ways in which a word's basic frames are connected with other frames through relations of inheritance (possibly multiple inheritance) and composition.

Tools that analyze the discourse structure of a text might be integrated in an extraction system. They include topic segmenters and recognition modules of rhetorical structures. *Topic segmentation* of texts concerns the detection of the overall organization of the text into themes or topics and the identification of text segments that correspond to the general and more specific topics. Existing segmentation algorithms usually produce what is called a *linear segmentation* of the text assuming that the main topics or subtopics are sequentially organized (e.g., Hearst, 1997; Kan et al., 1998). Innovative topic segmentation algorithms allow detecting the hierarchical and sequential topical segments including semantic returns of a topic at different levels of topical detail (Moens, 2006). *Rhetorical Structure Theory* (RST) was developed in the second half of the 1980s at the University of Southern California as a comprehensive linguistic theory for determining the textual coherence structure of monologue discourse (Mann and Thompson, 1987). RST assumes a text to have a hierarchical organization based on asymmetrical, and recursively definable nucleus-satellite relationships. A rhetorical parsing algorithm segments a text into its rhetorical structure tree (Marcu, 2000).

Many information extraction applications will also use *named entity recognition* and *coreference resolution*. Since both detect and/or connect referents of basic semantic entities in text, they are usually considered to be forms of information extraction (see *infra*).

2.2.4 Machine Learning

Notwithstanding the success of the information extraction systems in the 1980s and 1990s, there was a growing concern in making the information extraction systems easily portable to domains other than the one a system was built for and eventually to use information extraction in open domains. Here, the high cost of the manual pattern drafting and the knowledge acquisition involved made researchers investigate the possibilities of machine learning approaches.

The application of machine learning methods to aid the information extraction task go back to work on the learning of verb preferences in the 1980s, which is published by Grishman and Sterling (1992) and Lehnert and Sundheim (1991) in the early 1990s. Other interesting research is early work on lexical knowledge acquisition by Kim and Moldovan (1993) and especially by Riloff and Lehnert (1993) on the famous AutoSlog system (Riloff, 1996). Soderland (1999) has done many information extraction experiments by using Muggleton's ILP (Inductive Logic Programming system) (Muggleton, 1991).

Most of these systems use supervised techniques to learn extraction patterns. The pattern recognizers or classifiers train from a training base of classified examples. The general idea is that a human expert annotates the fragments that should be extracted in a small corpus of training documents, and then the learning system generalizes from these examples to produce a function or rules that can be applied on previously unseen instances. The underlying idea is that it is easier to annotate documents than to write extraction rules, since the later requires some degree of programming expertise and usually relies on the skills of the knowledge engineer to anticipate extra patterns. Although for some applications symbolic, handcrafted knowledge is more convenient, we see a gradually increasing interest in the machine learning techniques from the second half of the 1990s onwards.

Several research experiments have demonstrated that the supervised learning techniques produce very good results compared to systems that use handcrafted patterns. The results approach the ones that depend upon handcrafted rules. The AutoSlog system of Riloff (1996) constructed a dictionary of extraction patterns for the MUC-4 terrorism domain that achieved 98% of the performance of the handcrafted patterns. The research of Soderland (1999) endorses these findings. In recent years the supervised learning techniques have become very popular in information extraction. The more current and the most successful algorithms are discussed in detail in Chap. 5. They include Support Vector Machines, maximum entropy modeling, hidden Markov models, conditional random fields, learning of decision rules and trees, and relational learning.

Since the second half of the 1990s, we see a definite interest in using unsupervised or semi-supervised learning for information extraction. Apparently the cost of annotation is still a major handicap in developing large scale information extraction systems or when porting an existing system to another domain. Many bootstrapping technologies that employ forms of weakly supervised learning were developed, which we will discuss in detail in Chap. 6. The aim is to learn a pattern recognizer from a small number of labeled examples and to improve the classifier by using the unlabeled examples, or at least learn a classifier whose performance is equal to one trained on the full labeled set. In addition, there are approaches that aim at eliminating the need for manual annotation entirely.

Most of the work in machine learning regards the acquisition of patterns for entity classification, entity relation recognition, semantic role classification, and recognition and resolution of temporal expressions. Very little research on automatically learning complete scenarios and scripts exists. An initial impetus is found in the learning of structured patterns by means of kernel methods, hidden Markov models, conditional random fields and relational learning as discussed in Chap. 5.

2.2.5 Some Afterthoughts

One of the most elusive properties of the human mind is without any doubt the ability to relate utterances in a text or conversation to some kind of conceptual model of the world. Despite the more than three decades of research into natural language understanding in general and information extraction in particular, we did not succeed yet in providing computers with this ability. Apart from the obvious computational complexity of the task, there are two main obstacles for putting real understanding into computers. First of all, determining the exact meaning of an utterance requires a considerable amount of knowledge about the text genre, about relevant conventions and implicitly assumed common background knowledge, and of data about the world in general to which the utterance refers. Any sophisticated form of natural language understanding would therefore require a comprehensive implementation of world knowledge (or for some applications domain knowledge) and of textual and conversational models that are necessary to interpret the exact function of an utterance in the flow of reasoning.

A second problem is that the relationships between language and meaning are still surprisingly unclear. This is so because on the one hand the exact processes that a language user employs to encode meaning into language are not as straightforward as relationships between other linguistic strata. On the other hand, there is no agreement at all as to how these relationships, as far as it is known, should be formally implemented in a computational-linguistic framework. For instance, it is relatively easy to design procedures that within a certain error margin unambiguously assign part-of-speech tags (e.g., noun, adjective) to words in an utterance, since the relationship between a word in context and its part-of-speech is relatively straightforward and most researchers in the field will more or less agree on its annotation. It is not so easy to do the same for a word in context and its meaning. Consider for example the following sentence.

The prime minister dissolved the parliament. (2.1)

For instance, the word **parliament** has six distinct definitions in the Merriam-Webster Online Dictionary and an NLP system will have to contain a considerable amount of knowledge about how parliaments are constituted and how they work in order to be able to determine that the definition "the supreme legislative body of a usually major political unit that is a continuing institution comprising a series of individual assemblages" conveys the meaning that is most relevant to sentence (2.1). For a complete understanding at the sentence level, the system would have to know that prime ministers are persons who are related to parliaments (although they are not necessarily a member of it); that in some countries prime ministers have the authority to dissolve or disband a parliament; and that the fact that a prime minister is involved implies that the parliament referred to is neither a medieval institutional body in England, nor a French court of justice that existed before the revolution of 1789. In realistic natural language applications, the world knowledge that would be needed for word sense disambiguation could be partially replaced by a semantic network, in which only abstract semantic links between linguistic entities exist. But even though it would be relatively uncontroversial to define a membership relation between **parliamentarian** and **parliament** or a part-whole relation between **chamber** and **parliament**, there is no such obvious relationship between **prime minister** and **parliament**. For example, in Belgium a prime minister cannot be a member of parliament, has no official authority to dissolve it, but can partake in sessions of one of its chambers and can in particular situations instigate the parliament's dissolution (although only the king can effectively dissolve it). Even when someone should succeed in defining a set of relations that would be both uncontroversial and generally applicable, it remains to be seen whether it is feasible to build a semantic network of the size that will be necessary for real world applications.

When we move from the level of individual semantic entities (which is mainly the domain of lexical semantics) to the level of entities in context (which is primarily concerned with event analysis) the situation is at the same time more complex and more hopeful. Despite the fact that since the dawn of artificial intelligence a multitude of theories have been constructed for *event analysis*, no unified framework currently exists for describing the relationships between textual utterances and conceptualizations of events in a way that is useful in natural language processing. Grossly schematizing the complex field of event semantics, a rough distinction could be made between truth semantics, conceptual semantics, temporal semantics and modal semantics. Temporal and modal semantics are respectively concerned with the temporal allocation and the certainty or necessity of events. Truth semantics deals with the sufficient and necessary conditions for making valid judgments about event descriptions, i.e., its primary aim is making statements about the truth value of linguistic entities referring to events. Truth semantics is inherently conceived as a formalized model for describing event statements and for performing operations on them. Truth semantics does not contain any information about the conceptualization of linguistic entities, i.e., it might be able to say whether a statement is valid or not, but it cannot tell anything about what exactly the statement is about. Therefore a semantic framework is needed that describes the relationships between linguistic entities and a conceptualization of entities and events in the real world. What exactly such a cognitive world model might be, how language users acquire it and how it should be implemented in NLP are all matters of heated dispute, which we will steer away from. From all the competing (and often partially complementary) theories that exist, we will only remark the following. In frame theory, event types are encoded as semantic frames, each frame consisting of a number of attribute-value pairs, which are called *frame elements*. Despite its obvious potential for natural language processing and information extraction in particular, extraction systems that rely on frame theory have often been developed in an ad hoc fashion and built to cover a very limited subject domain. The introduction of a more generic theoretical linguistic framework when defining the frame semantics will be more advantageous as the portability and the applicability of the semantics is increased. One linguistic theory that could fulfill this task is systemic-functional grammar.

Systemic-functional grammar (Halliday, 1994; Halliday and Matthiessen, 1999; Butler, 2003) starts from the hypothesis that humans perceive reality through a mediating set of fundamental conceptual categories, which are reflected in the lexico-grammatical constructs of a language. These categories reflect that human observers primarily conceive the world around them as a never-ending series of (consecutive and parallel) actions and states. As a consequence, any linguistic expression can be analyzed in terms of the events that it describes, the entities that somehow are part of that event, and its worldly setting. Any linguistic description of a single event is centered around a *process* of a particular type. A process in its turn consists of a number of semantic roles: the *process role* itself, which describes an event in the real world; a restricted number of *participant roles*, which describe the real world entities partaking in that event; and an – in theory – unrestricted number of *circumstantial roles*, which describe the general setting of the process. *Participants* and *circumstances* characterize the "Who, what, when and how?" (or the "Who did what to whom, when and how?") expressed in sentences or phrases. Examples of process categories are *Material*, *Verbal*, *Mental*, *Behavioral*, *Existential* and *Relational.* The participant and circumstantial roles are process-category specific (e.g., **Mary** (Material_Actor) **gave** (Material_Process) **John** (Material_ Beneficiary) **a book** (Material_Goal) or **Mary** (Sayer) **said:** (Verbal_Process) **"Hi, John"** (Verbiage)). Systemic Functional process categories imply certain properties of the participants involved in them, e.g., Behavioral processes entail certain person-like experiences (i.e., mental experience). These implications (or entailments) have grammatical reflexes. Behavioral processes are analyzed as distinct from, for instance, Material processes in that they do not (without added syntactic machinery) have ergative constructions such as **Mary smiled John**. Circumstantial elements simply express temporal, locative, durative, etc. semantic content. Systemic-functional grammar offers very general semantic roles that might be more specified when building information extraction systems, but that can make up for the lack of semantic understanding of the text when certain extraction patterns for the specific semantic characterizations are lacking. In addition their classification contributes to the disambiguation of the meaning of an utterance or lexical item.

We conclude this historical framework with a final remark that is important if we want to build information extraction systems and incorporate them in information processing systems. *Cognitive linguistics* adheres to the belief that, rather than existing independently of meaning, grammar is symbolic in nature and inherently meaningful (Lanacker, 1999). It is claimed that all elements validly posited in grammatical description reside in the pairings of conceptualizations and ways of symbolizing them. Among these elements are grammatical markers, categories, relations, roles and constructions. This means that, if we are building information extraction patterns, grammar is important.

In the rest of this chapter we will define the typical information extraction tasks that have been implemented in working systems in information extraction and define a general architecture of an information extraction system.

2.3 The Common Extraction Process

2.3.1 The Architecture of an Information Extraction System

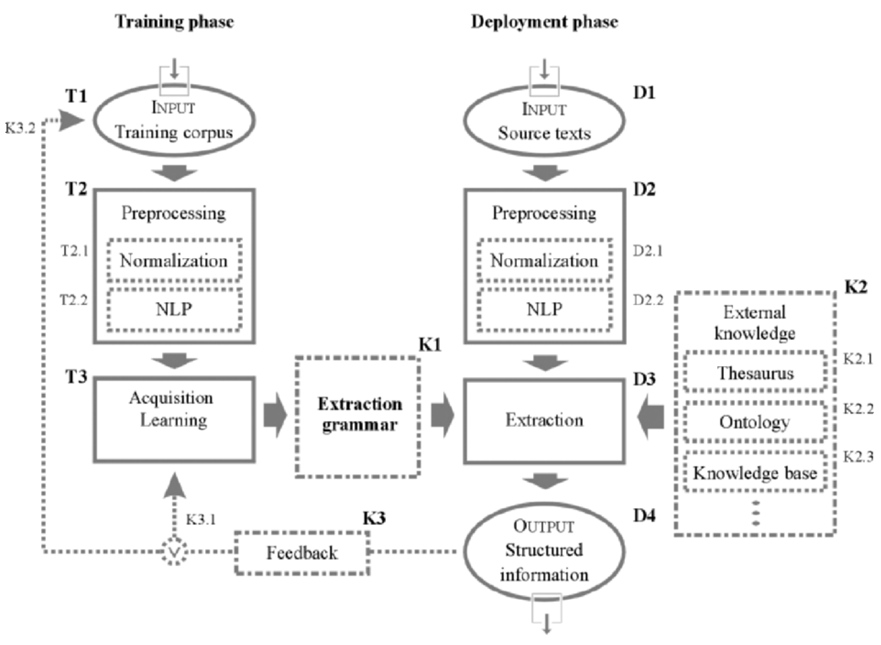
In this section, we will go deeper into the typical components of an information extraction system, the types of information it can extract, and what the theoretical foundations are for assuming that it is possible to extract these kinds of information.

Figure 2.1 *s*hows that the architecture of an operational information extraction system typically has two distinct phases: A training phase and a deployment phase. In the *training phase*, the knowledge engineer or the system acquires the necessary extraction patterns, the latter referring to the use of machine learning. In a first step, a text corpus is selected that is representative for the task the system is intended for (Fig. 2.1, T1).

Before the texts can be used for extrapolating extraction rules from them, they usually go through a preprocessing phase (T2) in which their formal characteristics are normalized. Another step belonging to the preprocessing phase is the enrichment of the textual data with linguistic metadata that will be used as parameters in the acquisition process (T2.2). To this end, a number of *natural language processing* tools can be used (see Sect. 2.2.3).

In the manual approach, an information specialist will use the preprocessed training corpus during the learning phase (T3) as a basis for writing an *extraction grammar.* In case of a machine learning approach, the training corpus is usually first manually annotated to indicate which elements in the texts are relevant for the extraction task and the machine learning module will use these annotations in the learning phase (T3) to automatically induce the extraction grammar from the corpus. The extraction grammar can here be in the form of a mathematical function that will predict the class of an example. It is also possible that the training corpus is not manually labeled or only partially annotated referring to respectively unsupervised and weakly supervised techniques.

In the deployment phase (D1-4), the information extraction system identifies and classifies relevant semantic information in new texts, i.e., texts that were not included in the training corpus. The preprocessing component in the deployment phase (D2) is as similar as possible to that in the learning phase. After preprocessing the input texts are passed on to the



Legend	
T	component of training phase
D	component of deployment phase
K	knowledge component

Fig. 2.1. A typical information extraction system.

extraction phase (D3), which uses the extraction grammar (K1) as it was learned in the learning step and possibly some additional knowledge (K2) to determine which elements in the input texts are relevant for the extraction task and how they relate to certain semantic classes. It extracts these textual elements from the texts, classifies them and outputs them in a structured format (D4). Some systems also have a feedback mechanism (K3) in which the final output of the system is corrected and used for retraining the learning component (incremental learning). Existing literature usually does not focus on the real world implementation of information extraction, but on development and testing, and as a consequence the deployment phase is often called the evaluation or testing phase.

It is obvious that the main task of an information extraction system is the extraction of semantic information from texts, and we already mentioned that this information is defined in advance of the extraction process. Different kinds of semantic information can be extracted from any one text, depending on the size of the linguistic units that are targeted in the extraction process and the linguistic context that is covered by the system. As seen above we will call the former the extraction unit or text region, and the latter the linguistic context.

2.3.2 Some Information Extraction Tasks

There are a number of typical information extraction tasks that lately have been extensively researched with regard to open domain information extraction. They include named entity recognition, noun phrase coreference resolution, semantic role recognition, entity relation recognition and timeline recognition. This is not an exhaustive listing of extraction tasks. We will often use these example tasks to illustrate the extraction algorithms discussed in the following chapters.

Named Entity Recognition

Named entity recognition recognizes and classifies named expressions in text (such as person, company, location or protein names).

Example:

$$\begin{tikzcd} \textbf{John Smith} \arrow[d] & \text{works for} & \textbf{IBM.} \arrow[d] \\ \textit{Person} & & \textit{Company} \end{tikzcd} \tag{2.2}$$

Noun Phrase Coreference Resolution

Two or more noun phrases are coreferent, when they refer to the same situation described in the text. Many references in a text are encoded as phoric references, i.e., linguistic elements that rather than directly encode the meaning of an entity, refer to a direct description of the entity earlier or later in the text. They are respectively called anaphoric and cataphoric references.

Example:

> **Bill Clinton** went to **New York**, where **he** was invited
> for a keynote speech. The former president ... (2.3)

Bill Clinton, he and **the former president** refer in this text to the same entity. **He** refers to an anaphoric reference.

Semantic Role Recognition

Semantic role recognition regards the assignment of semantic roles to the (syntactic) constituents of a sentence. They regard certain actions or states, their participants and their circumstances. Semantic roles can be very generally defined (e.g., the roles defined by the theory of systemic-functional grammar: cf. p. 37 ff.) or be more specific (e.g., as found in the FrameNet database and the actors and circumstance shown in the example below).

Example:

$$\underset{\substack{\downarrow\\ \textit{Agent}}}{\textbf{She}}\ \textbf{clapped}\ \underset{\substack{\downarrow\\ \textit{Body part}}}{\textbf{her hands}}\ \underset{\substack{\downarrow\\ \textit{Cause}}}{\textbf{in inspiration.}} \qquad (2.4)$$

Entity Relation Recognition

The relation between two or more entities is detected and the relation possibly is typed with a semantic role.

Example:

$$\begin{tikzcd} \textbf{John Smith} \arrow[d] & \textbf{works for} \arrow[d] & \textbf{IBM.} \arrow[d] \\ \textit{Person} & \textit{Relation works for} & \textit{Company} \end{tikzcd} \qquad (2.5)$$

Timex and Time Line Recognition

A first task is timex (i.e., temporal expression) detection and recognition in text. Temporal expressions to be marked include both absolute expressions (**July 17, 1999**, **12:00**, **the summer of ’69**) and relative expressions (**yesterday**, **last week**, **the next millennium**). Also noteworthy are durations (**one hour**, **two weeks**), event anchored expressions (**two days before departure**), and sets of times (**every week**). From the recognized timexes, the time line of different events can be reconstructed. Basic temporal relations are: X before Y, X equals Y, X meets Y, X overlaps Y, X during Y, X starts Y, X finishes Y. Recognizing a time line involves sophisticated forms of temporal reasoning.

Example:

On **April 16, 2005** I passed my final exam. **The three weeks before** I studied a lot. (2.6)

March 26, 2005-> April 15, 2005: Study
April 16, 2005: Exam

Table 2.1 shows the selected number of information extraction tasks. Note that the extraction unit for these particular tasks, i.e., the unit to be semantically classified or structured, is quite small and spans several word groups at most. The linguistic contexts used in the classification and the eventual goals of the tasks differ. The linguistic context enlarges as the goal of the understanding grows in scope.

The above extraction tasks are rather domain independent. But, they allow already identifying many of the details of an event (e.g., time, location). Domain dependent extraction tasks can be defined to complement an event description (e.g., the number of victims of a terrorist attack, the symptoms

Table 2.1. Examples of information extraction tasks, their respective extraction units and linguistic contexts, and the eventual goal of the information extraction.

Information Extraction task	**Extraction unit**	**Linguistic context**	**Eventual goal**
Named entity recognition	Word/ Word group	Sentence/text	Entity understanding
Noun phrase coreference resolution	Word/ Word group	Sentence/text/ Multiple texts	Entity understanding
Semantic role recognition	Word/ Word group	Sentence	Sentence understanding
Entity relation recognition	Words/ Word groups	Sentence/text/ multiple texts	(Multi-text) discourse/story understanding
Timeline extraction	Words/ Word groups	Sentence/text/ multiple texts	(Multi-text) discourse/story understanding

of a disease of a patient). At this level, information extraction is mainly interested in the extraction of information about individual events (and states), the status of participants in these events and their spatial, temporal, causal, … setting.

Events in the real world never exist in isolation, but rather are part of more complex events that are causally linked to each other. Humans recognize these linked events as event complexes because they stereotypically occur in a certain order. We call these stereotyped event complexes *scripts* or *scenarios*. The eventual goal of information extraction at a textual level is to recognize scenarios and to link them to abstract models that reflect complex events in the real world. In some cases, the analysis of texts into scenarios might not be really meaningful. Policy reports or court decisions, for instance, might not contain a real event structure.

Complex events are not the largest semantic structures that can be found in textual data. They are often part of multi-event structures that can be ordered chronologically to represent an entire story. These structures usually span multiple texts and are to be distinguished from scenarios in that causality is often not as important as chronology. Eventually, it should be possible to extract complex chronologies of events from entire text corpora and to locate scenarios, single events and the entities participating in these events in their temporal and causal setting.

2.4 A Cascade of Tasks

Many of the extraction tasks rely on the results of other information extraction tasks. Typically a bottom up analysis is performed in several stages. For instance, it is valuable to perform named entity recognition before noun phrase coreferent resolution (e.g., in the above example (2.3) of noun phrase coreferent resolution it is valuable to know that **Bill Clinton** is a person, before resolving the anaphor **he**). Defining the semantic roles of a sentence's constituent can be performed prior to the classification of relations between entities. It is also impossible to determine the scenario underlying a text without first being able to identify individual events, since that is exactly what a scenario is: A chain of events that is ordered in a meaningful, structured way.

Future information extraction systems may well be cascaded systems in which the output of one information extraction system is used as input of another information extraction system (e.g., the results of one information extraction task are used as features for training another information extraction system that more globally understands the text) (see Fig. 2.2). Actually, the foundations for such an approach were already set by the FASTUS information extraction system, which we will discuss in detail in the next chapter.

2.5 Conclusions

In this chapter we outlined the history of information extraction. The historical perspective allowed to smoothly introduce some important – and mainly domain independent – information extraction tasks. The architecture of a classical information extraction system was explained together with some possible future improvements. In the next chapters we discuss the most important information extraction algorithms. Chap. 3 explains the techniques that rely on symbolic, handcrafted extraction patterns.

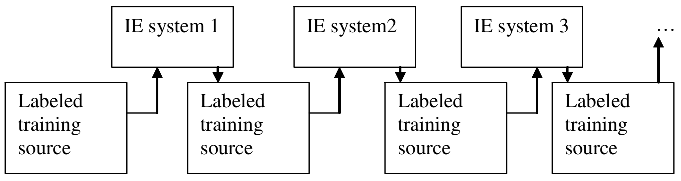


Fig. 2.2. A cascaded information extraction system.

2.6 Bibliography

Allen, James (1995). *Natural Language Understanding*. Redwood City: Benjamin/Cummings.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David J. Israel and Mabry Tyson (1993). FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1172-1178). San Mateo, CA: Morgan Kaufmann.

Baker, Collin F., Charles J. Fillmore and John B. Lowe (1998). The Berkeley FrameNet project. In *Proceedings of the COLING-ACL '98 Joint Conference* (pp. 86-90). San Francisco, CA: Morgan Kaufmann.

Butler, Christopher S. (2003). *Structure and Function: A Guide to Three Major Structural-Functional Theories.* Amsterdam, The Netherlands: John Benjamins.

Church, Kenneth (1988). A stochastic parts program and noun phrase parser for unrestricted texts. In *Proceedings of the Second Conference on Applied Natural Language Processing*. Austin, Texas.

DeJong, Gerald (1977). Skimming newspaper stories by computer. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (p. 16). Cambridge, MA: William Kaufmann.

DeJong, Gerald (1982). An overview of the FRUMP system. In Wendy G. Lehnert and Martin H. Ringle (Eds.), *Strategies for Natural Language Processing* (pp. 149-176). Hillsdale, NJ: Lawrence Erlbaum.

Fillmore, Charles J. (1968). The case for case. In Emmon Bach and Robert T. Harms (Eds.), *Universals in Linguistic Theory* (pp. 1-88). New York, NY: Holt, Rinehart, and Winston.

Fillmore, Charles J. and Collin F. Baker (2001). Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop.*

Hahn, Udo (1989). Making understanders out of parsers: Semantically driven parsing as a key concept for realistic text understanding applications. *International Journal of Intelligent Systems*, 4, 345-393.

Halliday, Michael A.K. (1994). *An Introduction to Functional Grammar*. London: Arnold.

Halliday, Michael A.K. and Christian M.I.M. Matthiessen (1999). *Construing Experience Through Meaning: A Language-based Approach to Cognition.* London: Cassell.

Hayes, Philip J. and Steven P. Weinstein (1991). CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In *2nd Annual Conference on Innovative Applications of Artificial Intelligence* (pp. 49-64). Menlo Park, CA: AAAI Press.

Hearst, Marti A. (1997). TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics,* 23 (1), 33-64.

Grishman, Ralph and John Sterling (1992). Acquisition of selectional patterns. In *Proceedings of the 14th International Conference on Computational Linguistics* (*COLING*) (pp. 658-664). Morristown, NJ: ACL.

Jacobs, Paul S. (Ed.) (1992). *Text-based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval.* Hillsdale, NJ: Lawrence Erlbaum.

Jacobs, Paul S. and Lisa F. Rau (1990). “SCISOR”: Extracting information from on-line news. *Communications of the ACM*, 33 (11), 88-97.

Kan, Min-Yen, Judith L. Klavans and Kathy R. McKeown (1998). Linear segmentation and segment relevance. In *Proceedings of 6th International Workshop of Very Large Corpora (WVLC-6), Montréal, Québec, Canada: August 1998* (pp. 197-205).

Kim, Jun-Tae and Dan I. Moldovan (1993). PALKA: A system for lexical knowledge acquisition. In *Proceedings CIKM 93, Proceedings of the Second International Conference on Information and Knowledge Management* (pp. 124-131). New York: ACM.

Langacker, Ronald W. (1999). *Grammar and Conceptualization.* Berlin: Walter De Gruyter.

Lehnert, Wendy G. (1982). Plot units: A narrative summarization strategy. In Wendy G. Lehnert and Martin H. Ringle (Eds.), *Strategies for Natural Language Processing* (pp. 375-412). Hillsdale, NJ: Lawrence Erlbaum.

Lehnert, Wendy, Cardie Claire, David Fisher, Joseph McCarthy and Ellen Riloff (1992). Description of the CIRCUS system as used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference MUC-4* (pp. 282-288). San Francisco, CA: Morgan Kaufmann.

Lehnert, Wendy and Beth Sundheim (1991). An evaluation of text analysis techniques. *AI Magazine,* 12 (3), 81-94.

Mani, Inderjeet (2003). Recent developments in temporal information extraction. In *Proceedings of Recent Advances in Natural Language Processing, Borovets Bulgaria,10-12 September 2003* (pp. 45-60).

Mann, William C. and Sandra A. Thompson (1987). *Rhetorical Structure Theory: A Theory of Text Classification.* ISI Report ISI/RS-87-190. Marina del Rey, CA: Information Sciences Institute.

Marcu, Daniel (2000). *The Theory and Practice of Discourse Parsing and Summarization.* Cambridge, MA: The MIT Press.

Miller, George A. (Ed.) (1990). Special issue: WordNet: An on-line lexical database. *International Journal of Lexicography*, 3 (4).

Minsky, Marvin (1975). A framework for representing knowledge. In P.H. Winston (Ed.), *The Psychology of Computer Vision* (pp. 211-277). New York: McGraw-Hill.

Moens, Marie-Francine (2006). Using patterns of thematic progression for building a table of contents of a text. *Journal of Natural Language Engineering* (forthcoming).

Muggleton, Stephen H. (1991). Inductive logic programming. *New Generation Computing,* (4), 295-318.

Palmer, David D. (2000). Tokenisation and sentence segmentation. In Robert Dale, Herman Moisl and Harold Somers (Eds.), *Handbook of Natural Language Processing* (pp. 11-35). New York, NY: Marcel Dekker.

Riloff, Ellen (1996). An empirical study for automated dictionary construction for information extraction in three domains. *Artificial Intelligence,* 85, 101-134.

Riloff, Ellen and Wendy Lehnert. Automated dictionary construction for information extraction from text. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications* (pp. 93-99). Los Alamitos, CA: IEEE Computer Society Press.

Rumelhart, David E. (1975). Notes on a schema for stories. In D.G. Bobrow and A. Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science* (pp. 211-236). New York, NY: Academic Press.

Rumelhart, David E. (1977). *Introduction to Human Information Processing.* New York, NY: John Wiley and Sons.

Sager, Naomi (1981). *Natural Language Information Processing: A Computer Grammar of English and Its Applications.* Reading, MA: Addison-Wesley.

Schank, Roger C. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology,* 3 (4), 532-631.

Schank, Roger C. (1975). *Conceptual Information Processing.* Amsterdam: North Holland.

Schank, Roger C. and Robert P. Abelson (1977). *Scripts, Plans, Goals and Understanding. An Inquiry into Human Knowledge Structures*. Hillsdale, NY: Lawrence Erlbaum.

Soderland, S. (1999). Learning information extraction rules from semi-structured and free text. *Machine Learning*, 34 (1/3), 233-272.

Young, Sheryl R. and Philip J. Hayes (1985). Automatic classification and summarization of banking telexes. In *The Second Conference on Artificial Intelligence Applications: The Engineering of Knowledge Based Systems* (pp. 402-408). Washington, DC: IEEE Computer Society Press.