An Overview and Classification of Adaptive Approaches to Information Extraction

Christian Siefkes[1,2] and Peter Siniakov[1]

[1] Database and Information Systems Group, Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany
siefkes@inf.fu-berlin.de, siniakov@inf.fu-berlin.de
[2] Berlin-Brandenburg Graduate School in Distributed Information Systems*

Abstract. Most of the information stored in digital form is hidden in natural language texts. Extracting and storing it in a formal representation (e.g. in form of relations in databases) allows efficient querying, easy administration and further automatic processing of the extracted data. The area of information extraction (IE) comprises techniques, algorithms and methods performing two important tasks: finding (identifying) the desired, relevant data and storing it in appropriate form for future use.

The rapidly increasing number and diversity of IE systems are the evidence of continuous activity and growing attention to this field. At the same time it is becoming more and more difficult to overview the scope of IE, to see advantages of certain approaches and differences to others. In this paper we identify and describe promising approaches to IE. Our focus is adaptive systems that can be customized for new domains through training or the use of external knowledge sources. Based on the observed origins and requirements of the examined IE techniques a classification of different types of adaptive IE systems is established.

1 Introduction

1.1 Information Extraction

There are things unique to humans that astonish and fascinate at the same time. One of them is the human language, admirable for its richness, complexity and ability to adapt to different cultural and social environments. But as valuable from cultural and aesthetic point of view human language is as challenging it is to grasp it scientifically, to formalize and make it manifest for computer processing. *Information extraction* (IE) builds the bridge between the evolutionary aspects of language development and the algorithmic approach to language. IE is one of the most promising efforts to exploit computational capabilities, accurateness and correctness of machines for accomplishing elaborate, often tedious task of searching for, analyzing and identifying desired information.

* The work of this author is supported by the German Research Society (DFG grant no. GRK 316).

S. Spaccapietra et al. (Eds.): Journal on Data Semantics IV, LNCS 3730, pp. 172–212, 2005.

Most of the information stored in digital form is hidden in natural language texts. Extracting and storing it in a formal representation (e.g. in form of relations in databases) allows efficient querying and easy administration of the extracted data. Moreover, information stored and queried in a canonical way can be processed and interpreted by computers without human interaction; it can serve for establishing ontologies, creation of knowledge bases and data analysis.

The area of IE comprises techniques, algorithms and methods performing two important tasks: finding (identifying) the desired, relevant data and storing it in appropriate form for future use. The notion of *fact extraction* is often used interchangeably with the notion of IE. The goals of fact extraction, however, are typically more specific and according to them fact extraction can be defined as the transformation of facts expressed in natural language to a given, formal, properly defined target structure. Fact extraction can therefore be regarded as a subset of IE extraction focusing on more rigidly structured representation forms.

1.2 Related Areas

A precursor of information extraction was the field of *text understanding* (or message understanding) which had the more ambitious aim of completely representing the contents of texts. To stimulate research in this area was the original goal of the *Message Understanding Conferences* (MUC) held from 1987 through 1998 under the auspices of the US government (ARPA/DARPA).

The term *text mining* (TM) is sometimes used almost synonymously to IE. It also denotes the application of data mining techniques to text with the goal of generating new knowledge by finding unknown patterns. TM in this second meaning aims farther than IE, which does not try to generate new knowledge, but only to represent facts explicitly expressed in a text in a more formal structure. But IE can be used as a first step in text mining, by extracting facts from the unstructured text to a database or other structured representation. In a second step, usual data mining techniques can be applied to the resulting database structure to discover interesting relationships in the data. This approach is utilized by [39].

IE and the better established field of *information retrieval* (IR) which locates relevant texts can be combined in various ways. IR can be used to select relevant documents for further analysis by IE. On the other hand, the structure filled by IE can also be utilized for more flexible IR (using a structured query language like SQL). Thus IE might be useful as a preparatory step for information retrieval as well as for postprocessing.

1.3 Goals and Evaluation Criteria

The history of IE as an independent field of research began in the 1980s as a number of academic and industrial research institutions were working on extracting information from naval messages in projects sponsored by the U.S. navy. After establishing of the Message Understanding Conference for comparing the performance of IE systems, information extraction experienced rapid growth extending

its applications to new domains and employing diverse new techniques. Originally consisting of handcrafted rule-based systems, the spectrum of IE methods has been continuously enriched by systems applying statistical and learning methods. Now it ranges from classical pattern-oriented systems over numerous combinations of different AI and statistical methods to rather new approaches such as wrapper induction.

The rapidly increasing number and diversity of IE systems are the evidence of continuous activity and growing attention to this field. At the same time it is becoming more and more difficult to overview the scope of IE, to see advantages of certain approaches and differences to others. Furthermore it is hard to estimate what the development perspective of IE is, what ideas are promising and where the focus of IE will be in the future.

In this paper we identify and describe promising approaches to IE. Our focus is adaptive systems that can be customized for new domains by training or the use of external knowledge sources. Handcrafted systems that can only be adapted by elaborate rewriting are not considered. According to the observed origins and requirements of the examined IE techniques, a classification of different types of adaptive IE systems is established. The classification is significantly based on the essential methods and resources used for extraction such as learning techniques and models and central features. Therefore the approaches that belong to different classes are not necessarily completely orthogonal to each other since some techniques and features are not exclusive to an approach (e.g. rule-based approaches may use some statistical techniques for solving some subtasks in the extraction algorithm).

Since the number of existing systems is considerably large it will not be possible to provide a detailed description of each system. Instead we select systems that distinctly represent directions of research without focusing on details of the systems. However, features of systems are identified that are common for the approach they pursue. Table 1 lists the regarded systems and the approaches they represent.

We distinguish three main classes: rule learning, knowledge-based and statistical approaches. In Sec. 3–5 the approaches are presented according to the classification so as related subclasses are discussed in the common context. To make the analysis of different approaches more systematic and establish a common base for their comparison and correlation we consider several qualitative criteria.

Used methods and algorithms: We focus on how relevant content is identified in texts and what techniques are used to match it to the target structure. Learning capabilities, learning models, the amount and role of human interaction are analyzed to infer advantages and weaknesses of the approach. These aspects form the basis of our classification and are mainly discussed in Sec. 3–5 where the different types of approaches are presented.

Input and output features: Input characteristics involve the prerequisites that the processed texts should fulfill and requirements on used resources. These characteristics affect the domains where approaches can be employed

Table 1. Overview of the Selected Approaches and Systems

Approach	System(s)	Section
Rule Learners		
Pattern & Template Creation	[40] [11] [44]	3.1
Covering Algorithms	Crystal [56, 52] Whisk [54] $(LP)^2$ [9, 10]	3.2
Relational	Rapier [4, 5] SRV [21]	3.3
Case-based	[6]	3.4
Wrapper Induction	Stalker [37, 38] BWI [22]	3.5
Hybrid (Decision Trees)	IE^2 [1]	3.6
Knowledge-based Approaches		
Horn Clauses	TANKA/MaLTe [13]	4.1
Ontology-based	[15]	4.2
Thesaurus-based	TIMES [2, 7]	4.3
Statistical Approaches		
Probabilistic Parsing	SIFT [35, 36]	5.1
Hidden Markov Models	Active HMMs [48, 49] Stoch. Optimization [23, 24] (C)HHMMs [51]	5.2
Conditional Markov Models & Random Fields	MEMMs [33] CRF [28, 34]	5.3
Token Classification	MaxEnt [8] MBL [59] TIE [50] ELIE [18, 19]	5.4
Fragment Classification & Bayesian Networks	SNoW-IE [46, 47] BIEN [41]	5.5

(application range) and how easily they can be adapted to new domains and resources (adaptability). It is examined how much preparatory work and linguistic preprocessing is necessary, whether morphological and syntactic analysis is presupposed etc. Another important factor is whether the approaches rely on external resources such as semantic resources (e.g. thesauri or ontologies).

Output features define the accomplished tasks—which IE tasks have been solved completely or partially. We consider whether single attributes of target structure can be identified in text (single slot extraction) or complex facts consisting of several attributes (template unification) can be found. A résumé over these characteristics is given in Sec. 6.

The description of a single approach features its analysis with respect to the proposed criteria, which is summarized at the end of the description. Sometimes

criteria are omitted if they are not applicable. The universally applicable criteria concerning input requirements and considered features, learning characteristics and accomplished tasks are used for comparison of different approaches in Sec. 6 and allow conclusions about important differences between the identified classes of approaches in IE.

Considering the quantitative metrics *precision*, *recall* and *F-measure* isolated from the testing environment may be misleading since they depend a lot on the complexity of the target structure and training texts. Quantitative parameters are meaningful only if the systems are tested in comparable environments, with the same text corpus and target structure. Section 6.5 discusses quantitative comparisons and evaluation results on two standard corpora.

2 Architecture of a Typical IE System

A typical trainable IE system follows a pipeline architecture that comprises linguistic preprocessing, learning and application stage and, during the application phase, semantic postprocessing as the three main blocks (Fig. 1). Each of them handles a subset of steps that are particularly relevant for a pursued approach.

A text corpus including texts of the application domain and a target structure defining what the relevant information is constitute the minimum input for an IE system. Besides, it can be supported by additional semantic resources provided by a human.

Preprocessing of Input Texts: Text corpora often consist of unstructured, "raw" natural language texts. A big part of the relevant information can be distinguished by some regularity found in the linguistic properties of texts. Thus linguistic analysis can give helpful hints and determine important features for identifying relevant content. Following linguistic components proved to be useful for information extraction:

Tokenization: Starting with a sequence of characters the goal is to identify the elementary parts of natural language: words, punctuation marks and separators. The resulting sequence of meaningful tokens is a base for further linguistic and any text processing.

Sentence Splitting: Sentences are one of the most important elements of the natural language for structured representation of the written content. Binding interrelated information they are the smallest units for expression of completed thoughts or events. The correct recognition of the sentence borders is therefore crucial for many IE approaches. The task would be trivial if the punctuation marks were not ambiguously used. Correct representation of a text as a sequence of sentences is utilized for syntactic parsing.

Morphological Analysis: Certain facts are typically expressed by certain parts of speech (e.g. names). Determining parts of speech of tokens is known as POS tagging. Statistical systems can use POS tags as classification features, rule-based systems as elements of extraction rules.

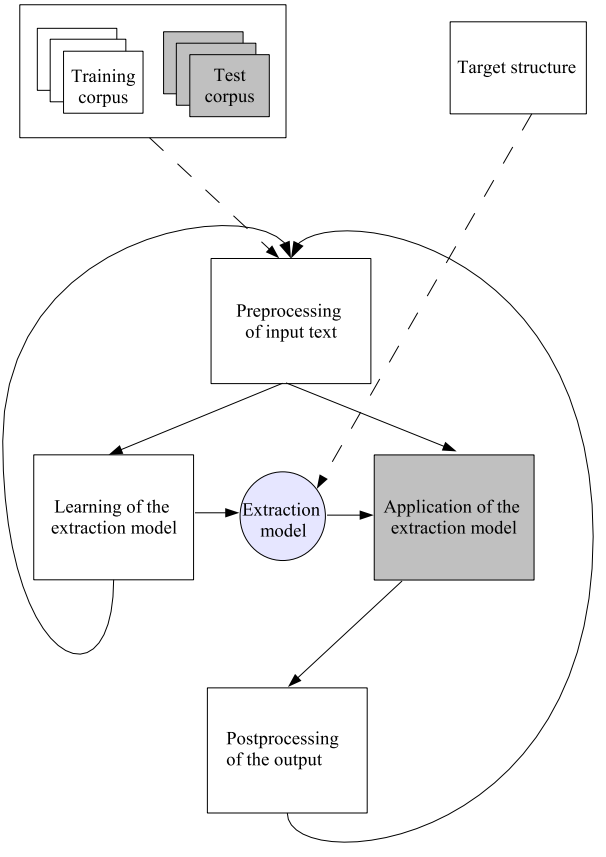

Fig. 1. Architecture of a Typical IE System

Segmentation of compounds, recognition of flection forms and consecutive normalization disclose further important morphological features.

(Chunk) Parsing: While full sentence parsing is preferred by knowledge-based systems, some statistical approaches rely on chunk parsing—shallow syntactic analysis of the sentence fragments performed on phrasal level. It is justified by the fact that the extracted information is often completely included in a noun, verb or prepositional phrase that build the most relevant context for its recognition.

Named Entity Recognition, Coreference Resolution: Named entities are one of the most often extracted types of tokens. Some approaches use a simple lookup in predefined lists (e.g. of geographic locations, company names), some utilize trainable Hidden Markov Models to identify named entities and their type. Coreference resolution finds multiple references to the same object in a text. This is especially important because relevant content may be expressed by pronouns and designators ("she held a seminar", "The company announced"). Both

tasks require deeper semantic analysis and are not as reliable as other linguistic components.

While for knowledge-based and some rule-based systems linguistic preprocessing is an element of the core system, for statistical and other rule-based approaches it is optional but can have a serious impact on the quality of extraction.

Learning and Application of the Extraction Model: The application range of today's IE systems is intended to be as wide as possible. The features of a concrete domain cannot be hardwired in a system since the adaptation effort to other domains is too high. Modern systems use a learning component to reduce the dependence on specific domains and to decrease the amount of resources provided by human. An extraction model is defined according to the pursued approach and its parameters are "learned" (optimized) by a learning procedure. Statistical approaches learn, for example, relevant classification features, probabilities, state sequences, rule-based approaches learn a set of extraction rules and knowledge-based approaches acquire structures to augment and interpret their knowledge for extraction. The challenge is to find an extraction model that allows learning all relevant domain parameters using the same extraction framework for each application domain.

Considering the problems and complexity of IE, supervised learning appears to be the most appropriate and is the most widely used learning technique. The majority of approaches prefer annotated training corpora albeit some rely on human supervision during the learning stage. To assess the quality of an approach the training text corpus is created by annotating text fragments that contain relevant content and divided into two parts. One part, the training set, is used for training (learning the parameters of the extraction model) and another, the test set, is used to test the ability of the model to correctly extract new information it was not trained on. The test results can also be used to improve the extraction model to perform better on new domain texts when applied to real domain texts.

Some approaches allow further refinement of an extraction model based on the human feedback about extractions during the application. The new evaluated extractions can be incorporated as new training instances and the model can be retrained.

The learning component is crucial for an IE system, because it comprises the algorithms for identification of relevant text parts and transferring them according to the target structure.

Postprocessing of Output: The main motivation for IE is the structured representation of information that enables formal queries and automatic processing. One of the possibilities to structure the extracted data is to model the target structure as a database relation. After the relevant information has been found by application of the extraction model the identified text fragments are assigned to the corresponding attributes of the target structure. They can normalized according to the expected format (e.g. representation of dates and numbers). Some identified facts may appear in text more than

once or already exist in the database. In this case, different instances could be merged (instance unification). Finally, the identified, normalized and unified information is stored at the appropriate relation in the database. Most current trainable systems do not yet perform much preprocessing, leaving such tasks as future work.

3 Rule-Learning Approaches

3.1 Automatic Pattern and Template Creation

To overcome some serious limitations of classical rule-based approaches, alternative techniques have been developed that reduce the manual effort and the amount of human knowledge used for the creation of extraction rules. In the optimal case the rules are determined automatically after the information about data to be extracted has been provided. Automatic acquisition of linguistic patterns and templates partially performs this task constructing the left-hand side of the rule and the target structure respectively. It is noteworthy that this approach does not presuppose a fix given target structure, in fact, the target structure is determined dynamically using provided semantic information. Therefore methods described below do not belong to the scope of fact extraction, but certainly comprise one substantial direction of IE. As representatives for this approach Nobata and Sekine's system for pattern acquisition [40], a method for template creation proposed by Collier [11] and a successor of *AutoSlog-TS* that accomplishes both tasks [44] are considered.

To compensate the lack of human interaction, syntactic and lexical resources should be provided that sufficiently cover word semantics and disclose necessary domain information. Therefore preclassified text corpora (Riloff) or reliable IR technology (Collier), a thesaurus or keyword list, part-of-speech (POS) tagger (Nobata) and named entity recognizer are required. Moreover shallow parsing is needed for syntactic analysis. Riloff's system is supported by a list of categories with five seed words in each for the creation of semantic lexicon and by a set of template slots (roles) mapped to corresponding categories. In all systems human inspection of intermediary or final results serves for quality assurance and learning purposes.

Pattern acquisition: Methods for automatic pattern acquisition have in common that the acquired patterns have a simple syntactic structure and the final set is selected from a large amount of initial candidate patterns iteratively using heuristics and statistical methods. The accent of Nobata's system lies on finding patterns describing certain events—an overview of the process is given in Fig 2. Actual information extraction is a direct mapping of ordered lexical items matching acquired pattern to fix template slots. Patterns are acquired by consecutive selection of text fragments and final merging of ordered lexical items in similar sentences. Articles are retrieved by scenario relevant keywords from a large untagged corpus, which are filtered by subject line. Selected articles are POS-tagged and named entities (NE) are recognized. Every sentence in the

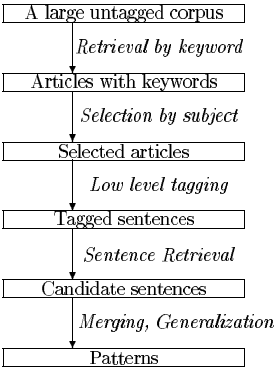

Fig. 2. Overview of Nobata's algorithm (from [40])

selected articles is regarded as initial pattern. Final patterns are acquired by merging lexical items and named entities of two similar sentences. Sentences are considered similar if they have the smallest amount of different items. Merging is facilitated by either ignoring extra items of one of the patterns or creating clusters of different items of both patterns. The merging process is repeated iteratively until the biggest possible generalization is achieved. In a later work [57] a tree representation for patterns is proposed to better account for dependency structures of syntactic patterns and to cope with the problem of free word order.

Riloff's system uses *AutoSlog-TS* for the generation of patterns [43]. It is guided by an assumption that the items to be extracted are comprised by noun phrases, therefore it uses heuristics to create linguistic patterns that represent relevant context for extracting of a given noun phrase (NP). These should be general enough to extract other relevant NPs as well. Typical patterns would be *<subject> exploded; exploded in <noun-phrase>*. They are activated by a keyword and information to be extracted is contained in a syntactic constituent of the pattern clause. In the first stage patterns are generated that collectively extract every noun phrase from the training text. In the second stage their relevance in the examined domain is estimated and a ranking of patterns is produced. The pattern score depends on the number of extracted NPs also found in semantic lexicon for examined domain and on their percentage among all extracted NPs. After human review the best patterns are selected to extract relevant information.

Template generation: The idea of generating target structure automatically may appear somewhat surprising since humans are primarily interested in and determine what should be extracted. However, it is motivated by the fact that the number of templates can be considerably large and therefore difficult to manage. Generation of templates is guided by semantic constraints provided to the system in form of categorization of domain entities in a semantic lexicon or thesaurus.

Riloff's system [44] constructs the semantic lexicon taking only a list of categories with five seed entries as input. Context of the seed words is regarded for

each category and the words in context are scored. The score is basically the conditional probability that the word appears in category context. The top five are added to the category list and the process is continued iteratively. After several iterations a user corrects the list and approves the lexicon. After acquisition of patterns described above and lexicon creation semantic profiles of patterns are established. The correlation between a pattern and a domain category (semantic preference) is expressed based on extracted items. Therefore patterns are applied to relevant domain texts and assigned to categories depending on extracted NPs. These assignments serve later as constraints for the assignment of extracted items to template roles.

Another important source of semantic information provided by the user is the functional dependency between semantic categories and domain roles (template slots). It is based on a reasonable assumption that members of a category can fulfill only one role but one role can be fulfilled by entities of several categories. During extraction the role is assigned according to the semantic preferences (prevalent semantic categories of extracted items) determined earlier. A pattern can extract words of different categories and thus different roles. The presented patterns have a serious limitation extracting only one syntactic constituent and filling therefore only one template slot. To consolidate scattered information patterns that share the same trigger word and compatible syntactic constraints are merged into single pattern. Such a generalized pattern is able to extract items of several roles and creates a multi-slot template as the result of extraction. Hence templates are not fix but can comprise any subset of the set of roles. Giving the system the set of roles only sketches the scope of possible target structure while its actual creation is based on described algorithms.

Collier's approach proposes template creation on a pretty general level using syntactic information and statistical techniques. It identifies three types of information contained in texts, which are relevant for template creation: objects, their interactions and features. Named entities are considered as important objects and the assumption is made that fundamental objects can be found in every relevant domain text. Their identification can be facilitated by existing NE recognizers. Coreferences can be another source for object recognition. Interactions are expressed by verb/subject/object relationships on sentence level. Considering the categories of verbs (obtained from a thesaurus) classes of relationships can be established. Such a class would correspond to a template while features would be mapped to template slots. Relevant features can be found by looking on entities occurring not in every document, consulting thesauri, analyzing n-grams and collocations and using other statistical methods.

The main advantage of automatic creation of patterns and templates is obvious just by looking at the name of this approach. Provided with initial domain and semantic information, the described algorithms solve the problem of IE by generating patterns and templates and partially also extraction mechanisms. The algorithms include no or occasional human interaction and rather small human support involving basically review of obtained results. There is not much pre-processing and additional resources required for this approach, no deep syn-

tactic analysis is necessary. Many subtasks (e.g. creating patterns and semantic lexicon) are solved using robust statistical methods. Generally, presented approach achieves successful results using quite simple comprehensive techniques.

Due to automatic processing researchers restricted the syntactic structure of patterns to be very simple, which is a serious limitation since many relevant facts are expressed in a complex linguistic context with complicated syntactic structure. Facts expressed over multiple sentences remain uncovered. Therefore such patterns cannot be applied to every domain. Since human influence during the runtime is very restricted, the quality of final results depends very much on the quality of semantic information provided at the beginning. Especially the demands on domain specification by categories and roles have to be very detailed and precise. Because of functional dependency between a semantic category and a role wrong slots may be generated, which leads to an adulterated target structure. Another big problem is unknown words (not occurring in the training texts) since heuristics are necessary to decide which role to assign. Additionally, Nobata's system suffers from the choice of sentences based on keywords that may fail because of noise or polysemous keywords. Besides, generalization of patterns is quite limited since, while matching a pattern, clusters of items have to be searched for a matching element, which corresponds to matching against many patterns. Collier's assumption that relevant objects occur in all documents is also very arguable.

Generally, this approach can be applied to domains and languages where desired information is expressed by facts with simple syntactic structure. Although the described techniques are primarily designed for information extraction other areas involved in creation of lexical resources or translation may use them to solve related tasks. This approach is promising because of its main advantage, many weaknesses are not inherent and can be overcome in the future.

3.2 Covering Algorithms

A number of IE systems are based on *covering (separate-and-conquer) algorithms* [25], a special type of *inductive learning*. These systems require a predefined target structure, they do not create it. Except for *Whisk*, which employs active learning, they also require a set of fully tagged training texts where all text fragments that fill a certain slot in the target structure are marked. Based on this input, the systems learn rules that extract the tagged slot fillers. After learning rules that cover a part of the training instances, they remove (separate) these instances from the training set and continue to learn rules that cover (conquer) some of the remaining instances, looping until all or most of the training instances are covered. What is regarded as an instance and which features are considered depends on the system.

Crystal: Crystal [56, 52] builds on a chunk parser that identifies syntactic constituents (subject, verb phrase, direct and indirect object, prepositional phrases) and a domain-specific dictionary that specifies semantic classes for all words. Crystal looks for constituents that fit predefined conceptual types (e.g. *diagno-*

sis, symptom) and subtypes (a diagnosis is either *confirmed, ruled-out, suspected, pre-existing,* or *past*).

The definitions learned to extract subtypes identify a constituent to extract if certain constraints are fulfilled by the surrounding constituents. Constraints may test for word sequences contained in a phrase or for semantic classes of the head noun or a modifier of a phrase. For example, an "absent symptom" is extracted from the direct object if the head of the direct object is of the class *[Sign or Symptom]*, the verb is "denies" in the active voice, and the subject includes the word "patient" and has the head class *[Patient or Disabled Group]*. Negative constraints are not supported.

In a later work [55], the problem of negation is solved by learning different kinds of semantic relations (classes) in a predefined order. Rules for "absent ..." are learned (and applied) first. The examples covered by these rules are removed prior to learning rules for "present ...". Thus specialized rules like *verb group includes "not observed"* can be learned for the absent case, and general rules like *verb group includes "observed"* for the present case (without a predefined order this is not possible, because the general rule covers both cases).

Crystal learns suitable definitions by generalization, i.e. bottom-up: each training instance is used as a highly constrained initial definition. Crystal tries to unify "similar" definitions by relaxing constraints. The similarity metric is based on the number of constraint changes necessary to unify definitions.

Two definitions are unified by finding the most restrictive constraints that cover both. For semantic class constraints the most specific common ancestor in the semantic hierarchy is used; for word constraints the subset of words contained in both constraints is kept. If there is no common ancestor or the subset is empty, the constraint is dropped. The new definition replaces the original ones if the number of false positives it extracts is below a defined error threshold—increasing this threshold results in higher recall at the cost of precision (and vice versa).

For multi-slot extraction Crystal treats each subset of slot combinations as a concept to be learned—this can result in data sparseness. Crystal does not extract exact phrases, it only identifies a constituent to extract from. These are the major limitations of the system.

Whisk: The *Whisk* system developed later by the same author [54] is aimed at handling a larger range of texts, from free texts as found in newspapers and books to semi-structured texts (often ungrammatical or in "telegram style") that are common on the World Wide Web or in advertisements.

Whisk is targeted at multi-slot extraction at the sentence level. The learned rules are a kind of regular expressions. Expression pattern can contain verbatim text, character classes (e.g. digit, number), and wildcards like "*" which lazily skips any characters until the next part of the pattern can match. In addition to the hard-wired character classes, semantic classes of equivalent terms can be defined by the user, e.g. a class *Bdrm* that contains different forms and abbreviations of the term "bedroom." Parentheses indicate a phrase to be extracted. The *Output* part of a rule specifies where to store the extracted phrases. "Pattern:: *

(*Digits*) *Bdrm* * '$' (*Number*). Output:: Rental {Bedrooms $1} {Price $2}." is a rule to extract the number of bedrooms and the price from a rental ad.

When processing free (grammatical) text, each sentence is split into the fields returned by the chunk parser (subject, verb etc.); these fields can be specified in the regular expressions to constrain matching (but still the whole expression must match left to right, so the ordering of fields matters). Additional semantic classes are defined that match the output of a named entity recognizer (*person*, *company* etc.).

Rules are derived top-down (starting with the most general rule) by a covering algorithm. For judging the quality of rules, Whisk uses the Laplacian expected error: $Laplacian = \frac{e+1}{n+2}$, where n is the number of extractions made and e is the number of errors among these. In case of a tie, the more general rule is used. The found rules might not be optimal due to the limitations of hill climbing—each specialization is evaluated in isolation, so if two specializations (adding two terms) must be applied together to yield a better rule (according to the Laplacian), they will not be found.

Whisk incorporates *active learning*, so only a small part of the training corpus needs to be tagged in advance. The system proceeds by selecting three kinds of untagged instances for hand-tagging by the user: instances covered by a rule (which will either increase the support of the rule or force further refinement), "near misses" (to check and adapt the boundaries of rules), and a random sample of instances not covered by any rule (to check whether there are still rules to discover).

A disadvantage is that semantic classes must be predefined by the user, they are not learned by the system. Another drawback is the strict ordering constraints of each rule—different rules must be learned for each possible arrangement of slots.

(LP)²: (LP)² [9] learns rules to add SGML/XML tags to a text. (LP)² is based on *tagging rules* that insert a single (starting or ending) SGML tag into the text. This means that the task of each rule is to recognize the start or the end of a supposed slot filler in the text, not to extract/tag a whole slot filler (or several slot fillers) at once, as in most other systems.

The tagging rules are learned from the hand-tagged training corpus. Rules are learned bottom-up, taking an instance as an initial rule whose constraints are subsequently relaxed (e.g. requiring only a lexical class instead of a specific word) or completely dropped. The k best generalizations of each initial rule found by a beam search are stored in a "best rules pool." As (LP)² is a covering algorithm, the training instances covered by a rule in this pool are removed from the training set.

(LP)² proceeds in four steps:

1. The *tagging rules* from the "best rules pool" are applied.
2. *Contextual rules* are applied to resulting text. These are tagging rules whose overall reliability was not high enough for the best rules pool but that perform better when restrained to the vicinity of tags inserted in the first step

(for example, a rule that inserts an end tag is applied provided that a corresponding start tag occurred some words before).
3. *Correction rules* do not add or delete tags, they only change the position of a tag, moving it some words forward or backward.
4. Finally, invalid markup (unclosed tags etc.) is deleted in a *validation* step.

In the *Amilcare* system, $(LP)^2$ is employed in a "LazyNLP" setting where the amount of utilized linguistic information can be dynamically adjusted [10]. The learner initially induces rules without any linguistic knowledge; then it iterates adding linguistic information (provided by third-party components), stopping when the effectiveness of the generated rules no longer increases. The adequate amount of linguistic input is learned for each type of slot separately (e.g. recognizing a person name might require more NLP input than recognizing a date or time).

$(LP)^2$ is targeted at slot filling and does not perform any template unification. In Amilcare a shallow discourse representation module is added for this purpose [26, Sec. 5]. Slot fillers are unified in templates or subtemplates with the nearest preceding slot fillers of a suitable type. E.g. when describing hotels, *address* data and *room types* (single room, double room) will be attached to the last mentioned *hotel*; *price* information might in turn be attached to the last mentioned *room type*.

3.3 Relational Rule Learners

The basic approach of the systems presented in this section is similar to those of the previous section—indeed, they are based on covering algorithms too. The main difference is that the systems presented here explicitly take relations—especially positional relations—between a (potentially unlimited) number of features into account, while those in the previous section are limited to predefined (finite) combinations of features.

Rapier: The *Rapier* [4, 5] system uses syntactic (POS tags) and semantic (WordNet classes) information to induce rules for slot fillers. Each rule consists in three parts, a pre-filler pattern, a pattern for the actual slot filler and a post-filler pattern. Each pattern contains an ordered list (whose length might be zero for pre/post-fillers) of constraints that restrict the POS tag, the semantic class, and/or the word itself (disjunctions are allowed). Instances are most specific rules with all their constraints set. The pre- and post-filler patterns of instances contain every word from the start resp. to the end of the document, there is no "context window" of limited length.

New rules are created by randomly selecting two rules and creating the least general generalization for the filler pattern. Actually, there are several reasonable generalizations (different values of a constraint can be disjuncted or the constraint can be simply dropped), so each of these generalizations is re-specialized by adding generalized pieces of the pre- and post-filler patterns of the original rules. A list of n best candidates is kept until the best generalization is found, based on the evaluation metric

$$ruleVal = -\log_2\left(\frac{p+1}{p+n+2}\right) + \frac{ruleSize}{p},$$

where p is the number of correct extractions and n the number of erroneous extractions; *ruleSize* is calculated depending on the number of pattern items, lists, and disjuncts in a rule.

Semantic classes are generalized by finding the nearest common ancestor in the WordNet hypernym hierarchy (dropping the constraint if no common ancestor exists). Instances covered by the found best generalization are subsequently ignored and further rules are learned based on the other instances.

Rapier has been extended to use *active learning* [58]. The system is initially trained from a small number of annotated examples. Then it tries to annotate a large number of untagged examples, selecting those examples whose annotation is least certain (certainty-based selective sampling). After the user has annotated the selected examples, the system is incrementally retrained and the process continued. Rapier does not provide probabilities, so the certainty of a rule is estimated based on its coverage: $pos - 5 \times neg$, where *pos* is the number of correct extractions on the training data and *neg* the number of incorrect ones. The active-learning version requires approximately half the examples to reach the performance level of Rapier without active learning.

SRV: SRV [21] considers any combination of simple features (mapping a token to a value, e.g. *word length: 5, character type: alpha, orthography: capitalized, POS tag: noun, semantic class: geographical-place*) and relational features (mapping a token to another token, e.g. *next-token*, *subject-verb*). Feature values can be sets, e.g. all synonyms and hypernyms (superordinate concepts) listed by WordNet are combined in a set for each token. SRV performs only a two-class classification, i.e. different rule sets are learned for classifying each text fragment as an instance or non-instance of a single slot filler—there is no component for template unification or other postprocessing.

The learning algorithm is similar to the relational rule learner FOIL [42]. SRV learns top-down, greedily adding predicates of some predefined types: the number of tokens in the fragment (*length*), whether a condition is matched by one or several (*some*) or by all (*every*) tokens in the fragment; *position* specifies the position of a token in a *some* predicate, *relpos* constrains the ordering and distance between two tokens. The *some* predicate can be constrained by relational features, for example, *some(?A [prev_tok prev_tok] numeric true)* means: there is some token in the fragment preceded by a numeric token two tokens back.

Rules are validated and their accuracy estimated by three-fold cross validation. The three resulting rule sets are then merged. The accuracy estimations are available for each prediction.

An advantage of relational learners is their being able to acquire powerful relational rules that cover a larger and more flexible context than most other rule-learning and statistical approaches. The downside is that the large space of possible rules can lead to high training times and there is no guarantee of finding optimal rules (local maxima problem).

3.4 Case-Based Approaches to IE and Knowledge Acquisition

To identify semantic and syntactic word features, knowledge-based approaches rely on manually prepared world knowledge while statistical approaches are only guided by regularities and patterns found in the training corpus. Case-based approaches combine the advantages of both by accumulating knowledge only from the training corpus and using previous experience to handle new words. The case-based method suggested by Cardie [6] needs therefore no explicit disambiguation heuristics, but domain knowledge provided at the initial stage. It includes knowledge about plausible parts of speech, word senses and contexts in a given domain. The training phase is supervised by a user. The system solves three independent tasks: POS assignment, word sense disambiguation and determination of the word concept (category of the word). Processing of training texts results in a case base of cases comprising these three types of information for each non-functional word in the training corpus. The case base is used to perform the tasks described above in new domain texts with unknown words.

Case base is constructed by acquiring a case for each occurrence of any non-functional word in different context while parsing sentences of the training set. A case summarizes the actual word features and features of its context. Word features describe part of speech, general and specific word senses and concept, context features include word features of two preceding and following words and the parser state before processing of the current word. To define features of the current word the human user is consulted. For the specification of the context features of two preceding words case base is queried, the features of two following words are added after the parser reaches them.

When a new text is processed, sentences are parsed and the context features are identified in the same way. To identify the current word, the features of the most similar cases are retrieved from the case base. Feature values that occur in retrieved cases most frequently are selected to be the features of the new word. Similarity of cases is measured by k-nearest-neighbors metric. Subsets of case features that are relevant for determination of the value of each word feature are identified using decision trees. For each word feature the relevant subset of the context features of the currently processed word is compared with the same subset of context features of cases in the case base. The more equal feature values are found the more similar are the cases. Since similarity is influenced only by context features, the assumption is proposed that the context is the only factor that defines all three types of knowledge for a word.

The approach benefits from the fact that there are no fix word-concept pairings and a much more realistic dependency of concept of the word on context is assumed. Syntactic and semantic knowledge is learned simultaneously and stored in one composite structure—the case. Source of domain information can be human or semantic lexical resources, which makes the training of the system easier.

From the point of view of IE, category of a word and its sense are a very sparse "target structure". Semantics are revealed not on a sentence, but on a word level. Besides, it is arguable whether the context can be captured adequately by

regarding a static context window of 4 words. The assumption of exclusive role of context in determination of word features is simplistic.

An interesting perspective is the extension of this approach to work on the sentence level with cases summarizing sentence features, which would make solving more complex IE tasks such as fact extraction possible. Generally, similar problems in IE have similar solutions, therefore case-based reasoning methods are likely to be very promising for the future of IE.

3.5 Wrapper Induction

Stalker: The approach of *Wrapper Induction* (WI) is mainly targeted at structured and semi-structured documents that were generated automatically, e.g. Web pages offering products of listing information. The *Stalker* algorithm [37] covers documents that can be described in the *embedded catalog (EC)* formalism. This formalism represents a document as a tree whose leaves contain the relevant data (items of interest for the user). The inner nodes contain lists of k-tuples (e.g. of restaurant descriptions). Each item in a tuple is either a leaf or another (embedded) list.

Extraction is based of the EC description of a document and an extraction rule that extracts the contents of each node or tuple from the contents of its parent. List nodes require an additional *list iteration rule* that splits the list into tuples.

Extraction rules are based on groups of successive tokens called *landmarks*. *Start rules* locate the start of an item by find the first matching landmark from the begin of the parent; *end rules* locate the end of the item by finding the last matching landmark before the end of the parent. The text matched by a landmark itself can either be included (SkipUntil condition) or excluded (SkipTo condition) from the item text. Rules can combine several conditions, e.g. SkipTo(Name) SkipTo(<b>)' means that the item starts immediately after the first HTML tag <*b*> that follows the word *Name*. They can refer to specific tokens or to wildcards like Number, Punctuation, or HtmlTag.

Disjunctions (either ... or) are allowed to handle formatting variations. Disjunctions are ordered so the first successful match is used. Each node is extracted independently of the other nodes within its parent, so no fixed ordering is required.

Rules are learned by a covering algorithm that tries to generate rules until all instances of an item are covered (without false extractions, if possible) and returns a disjunction of the found rules. Rules with fewer false extractions (or more correction extractions, in case of a tie) are preferred when ordering the disjunction.

To support *active learning*, Stalker has been embedded in a *Co-Testing* approach [38]. Co-Testing combines a number of *views* that independently learn to recognize slot fillers. In the *Aggressive Co-Testing* approach, views can either be *strong* (they can learn how to reliably recognize slot fillers) or *weak* (they might learn either more general or more specific concepts, i.e. might either miss some instances or extract spurious instances).

Stalker is used as a *strong view*. Stalker's learning how to recognize the *begin* and *end* of a slot filler is complemented by a weak view that learns patterns to recognize the *content* of a slot filler (length range, contained token types etc.). The content recognizer is a *weak view* because it learns concepts that might be more general than the target slot filler, e.g. it cannot discern a phone number from a fax number. A second strong view is provided by running Stalker backwards, starting from the end of the document (BackTo instead of SkipTo). Predictions are combined by *majority voting:* When both strong views agree, the weak view is ignored; otherwise the prediction of the strong view that violates fewer constraints of the weak view is chosen.

Boosted Wrapper Induction: Typical WI algorithms such as Stalker are only suited for documents whose structure and layout are regular and consistent. They are inadequate for free text, where information is mainly expressed in natural language. The *BWI (Boosted Wrapper Induction)* system [22] aims at closing this gap and making WI techniques suitable for free text.

The rules learned by BWI are simple contextual patterns for finding the start and end of the field to extract. A pattern has two parts: a token sequence that immediately precede/follow the field to extract (outside) and a token sequence starting/ending it (inside). E.g. to identify the speaker's name in a seminar announcement, the pattern <[who :] [dr .]> would locate the start of all entries introduced by *Who:* and starting with the honorific *Dr.*. Patterns can also contain wildcards, e.g. <Alph>/<ANum>/<Punc> match tokens that contain only alphabetic/alphanumeric/punctuation characters, while <*> matches any token.

Such specialized simple patterns will often reach high precision but low recall because there are many other ways to express a fact, especially in natural language texts. To address this issue, a large number of simple patterns are learned and their results combined. For this purpose BWI applies the technique of *boosting,* i.e. repeatedly applying the learning algorithm to the training data, each time adjusting the weight of training examples to emphasize those examples where the algorithm failed before.

A wrapper learned by boosting consists in a set F of "fore" and a set A of "aft" detectors (patterns that detect the start and the end of a field) and a length function $H(k)$ that estimates the maximum-likelihood probability that the field has length k. A text fragment $< i, j >$ is extracted if $F(i) \times A(j) \times H(j-i) > \tau$. A trade-off between precision and recall is possible by varying the threshold τ. Generally, BWI is biased toward precision, so setting $\tau = 0$ results in a reasonable recall at still high precision.

While one of the goals of BWI is to make WI algorithms suitable for free (unstructured) text, BWI still performs significantly worse on free text than on highly or partially structured text [27]. Most detectors learned from free text merely memorize specific training examples [27, Sec. 4.2.3]. Also the algorithm is biased towards overfitting to the particularities of the training data—the final rounds of boosting actually lower the reliability of the results. Both precision and recall on free texts can be increased by incorporating the output of a shal-

low parser into the model, splitting the text into a number of noun, verb, and prepositional phrase segments [27, Sec. 8.1].

3.6 Hybrid Approaches

IE²: The *IE²* system [1] submitted by SRA International to the MUC-7 conference is built in a highly modular way. The output of a standard named entity recognizer is complemented by a custom component that recognizes domain-specific entities (e.g. different kinds of vehicles). Another component recognizes domain-specific types of noun phrases and relations between them (e.g. *employee_of*, *location_of*). Both these components are based on hand-written rules, no learning is involved.

However, the IE² system goes further than most approaches described in this paper in also handling template unification beyond the sentence level. For coreference resolution, different strategies are employed: one strategy uses simple hand-written rules, but another one learns decision trees (using the standard implementation *C50*) from a tagged corpus. Optionally these strategies are combined in a hybrid method where the decision tree algorithm works on a subset of possible candidates chosen by the hand-written rules.

IE² also handles *event merging*, i.e. deciding whether or not two descriptions refer to the same event and can be merged. Here hand-written rules are combined with external knowledge sources to check the consistency of locations (*Miami* is in *Florida*) and times (can *Wednesday* and *tomorrow* refer to the same day within the current text?).

While in most aspects IE² is a typical representative of the classical hand-written rules approach that was dominant in the MUC conferences, its hybrid nature has interesting traits. Template unification and event merging beyond the sentence level are complex challenges that so far have been largely out of reach for learning systems. Combining trainable modules with external knowledge sources and specialized hand-written code could be a viable approach to tackle problems where single-paradigm solutions fail.

4 Knowledge-Based Approaches

4.1 Translation of Texts into Horn Clauses

One of the main purposes of IE is to make efficient search and transactions on extracted information possible. Therefore one of the central requirements on the target structure is an efficient querying mechanism. Another important issue is how expressive a target structure is. A relational target structure can for instance express only facts described by predicates but no conjunctions or disjunctions. The approach suggested by Delisle et al. [13] envisions Horn clauses as the representation resource of the target structure. Such a target structure is more expressive and more powerful in comparison with a relational target structure. Horn clauses allow conjunctions and implicit disjunctions and can be used to

infer new facts, draw inferences about extracted information or build a domain theory. Generally, any logical techniques can be applied to this representation.

Since the extracted information is presented in logical form the application of this approach can be regarded as knowledge extraction. Remarkably, the information is extracted exhaustively, that is, everything identified as information is extracted. Such an approach differs significantly from fact extraction paradigm since neither a fix target structure is available nor any focus on items of interest can be given. However, this approach still can be related to IE since relevant (in this case all) information is found in an unstructured text and transferred into formal structure.

The approach does not presuppose any external knowledge sources except for continuous human interaction. To start semantic analysis, shallow parsing of the text is necessary. Parsed sentences are matched against a growing set of (potentially learned) cases. A case is a semantic interpretation of certain syntactic patterns, more precisely, an interpretation of dependencies between the verb and its arguments within a clause produced by a parser. Examples of cases are *Agent, Accompaniment, Location to, Time at.* The best match and thus semantic interpretation is suggested by a case analyzer and either confirmed or corrected by the user. If a significantly different structure of a clause (primitive sentence) is detected (compared to those stored in case pattern list), a new case is stored. Case assignment relies on a hypothesis that "syntax gives a reliable indication of meaning" (at least in examined technical texts).

At the beginning no predefined case patterns are necessary, however, user interaction increases in this case. Bigger elements of a sentence (clauses) are then matched against a restricted number of predefined patterns (causation, prevention, disjunction...) to identify their semantic relationship. The conjunction between the clauses plays thereby a significant role. Every time a match is found it is suggested to the user for confirmation. Additional syntactic information is regarded to label dependencies between the clauses. Source of domain information is the user. Cases are not assigned to "stative clauses" (clauses of form of the verb to be). These are processed directly by creating corresponding predicates. Translation from found semantic dependencies to Horn clauses occurs after clause dependencies, cases in the clauses and semantics of nouns (determined with WordNet) have been completely analyzed. Identified cases are translated into predicates reflecting the pattern structure of the case. Clause dependencies suggest how to piece together the Horn clause. For example, the sentence "Jim is a resident of Canada because he is serving abroad in the armed forces" would be transformed into:

```
is_resident_of(jim, canada) :- serve_agt_lat_benf(jim, abroad, armed_forces).
```

Identified Horn clauses are passed to the EBL (explanation based learning) module for building of domain concepts and developing domain theory.

The biggest strength of this approach is the resulting expressive and powerful target structure, which allows plenty of possibilities for further processing. The idea to directly transform identified cases into predicates and compose the Horn clause according to found clause relationships is very beneficial for the approach.

Since cases and clause relationships already contain semantic dependencies, Horn clauses can be constructed in a very efficient and consistent way. However, the way semantic relationships are derived from syntactic patterns is quite critical because the underlying assumption does not always definitely hold; especially, it is hardly applicable to other languages. Besides, case matching may fail as many clauses would be semantically ambiguous because of interpretation by syntactic structure.

The considerable involvement of human user is advantageous on the one hand since the system gets reliable information, but on the other hand may cause much effort given a text with big variety of case patterns. However, the learning component successfully applies learned cases and case triggers so that the amount of human interaction reduces with time. Another learning unit based on EBL is used in the last stage of processing Horn clauses.

The algorithm is explicitly designed for knowledge acquisition task and can hardly be applied to any other NLP tasks without serious modifications. Giving up the basic assumption and extending case patterns to include lexical data would possibly make it applicable to other languages. Since the main source of domain information is the user and no assumptions are made about the domain properties (except for technical texts) this approach could easily be adapted to various environments.

4.2 Ontology-Based Extraction

The goodness of the initial semantic and syntactic domain description is crucial for any rule and knowledge-based approach to IE. Usually this domain information does not represent any abstract logical dependencies and relations in the domain, since it is trimmed to IE purpose excluding comprehensive elements. Embley et al. [15] chose the ontology as one of the most explicit and complete knowledge representation forms trying to employ its possibilities to express deeper semantic relations.

However, using the ontology imposes some restrictions on the kind of processed texts: they have to belong to a quite narrow domain and contain many constants, which can potentially be extracted (e.g. proper names, numbers). The extraction algorithm relies on the existence of a manually created ontology; elements of the ontology cannot be used directly. Consequently tools for parsing and transforming the ontology in suitable form are required.

The user does not provide an explicit target structure. An ontology parser creates a relational database schema that serves as the target structure for consecutive extraction. Since the same ontology is used for all texts in a certain domain and the parser output is deterministic, one can conclude an existence of predefined, fix target structure, even though it has to be derived anew from the ontology for each domain. The parser produces also a list of constants and keyword rules that describes properties of relations in the ontology and their attributes or their occurrence in text. Possible value range and text occurrence are specified as a regular expression.

In the next step regular expressions are applied to the text to identify relevant items. One expression matches potentially several times, therefore matching text fragments are temporarily kept in a list for "candidate extractions". The decision whether an item will be extracted and what item will be chosen if there are several candidates is guided by heuristics. Heuristics test candidate items for proximity of a relevant keyword, overlapping etc. The item that is the next to a keyword and in case of overlapping items the subsuming one is chosen. If an ontology allows many values for an attribute of a relation all found matches are inserted. Otherwise, if the criteria mentioned above are not applicable, the first one is extracted and the rest ignored. The output is a database with extracted items.

In this approach the ontology represents the only resource and knowledge bundle necessary to process every text of the described domain. The possibility to derive necessary rules or information for extraction from the ontology makes the approach flexible. An ontology contains also predicates describing entity relationships between entities and inference rules. This additional semantic information can be used for more reliable identification of desired facts in a text.

However, the structure of the ontology used in this approach goes far beyond the conventional notion including representational aspect and regular expressions. It is basically a collection of related resources. The extraction suffers from the fact that regular expressions cannot match non-trivial natural language expressions or whole sentences because of their complexity, so some items will not be extracted because they cannot be identified. Moreover, the used regular expressions are not changed or updated according to the results of extraction, learning mechanisms are not employed. Manual creation of ontologies is very tedious and hard to manage for bigger domains.

In the current form the ontology-based approach can handle listings, enumerations, generally preformatted text elements, but not complete sentences. It can be enhanced if creation of regular patterns is not static and manually specified, but can be dynamically influenced by the semantic level of the ontology. Currently the subject of extraction is mainly numbers and proper names. To cope with the variety of natural language the system should be able to extract other parts of speech. Changing elements of an ontology based on the extraction results would ease the adjustment of the ontology to domain texts.

4.3 Thesaurus-Based Extraction

The *TIMES* system developed by Bagga and Chai [2] requires a number of knowledge sources: the WordNet thesaurus, a general English dictionary, a domain-specific dictionary, and a gazetteer of location names. Texts are preprocessed with a tokenizer, a sentence splitter, an entity recognizer that identifies named and numeric entities, and a partial parser that recognizes noun, verb, and prepositional groups with their respective head words. The preprocessing components are based on finite-state rules.

Training is done by a user through a graphical interface. For each of the head words identified by the parser, the user selects the appropriate sense (concept) if WordNet defines several senses for this word. Then the user builds a *semantic*

network to represent the content of each training text. Selected head words from the text are stored as nodes or relations within the network. For example, from the phrase *IBM Corp. seeks jobs candidates in Louisville*, the user might build a relation *seek* between two nodes *IBM Corp.* and *job candidate*.

The text-specific extraction rules created this way are then generalized according to the hypernym/hyponym (super-/subordinate terms) relations defined in WordNet. Generalization replaces a term by its hypernym n steps higher in the WordNet hierarchy. For named entities (NE), the category determined by the NE recognizer is generalized. E.g. *IBM Corp.* is identified as a company—generalizing this concept one step yields business, concern; three steps yields organization. A generalized rule matches any terms that are hyponyms of the generalized term. Increasing the generalization level results in higher recall at the cost of precision, because the generalized rules find instances missed by specialized rules but also produce more false positives.

In later versions of the system, the user only has to mark the target information to extract from a text. The system automatically builds relations between the marked information and generalizes extraction rules to the most suitable level [7].

The hypernyms of a word are sense-dependent. Rules for sense disambiguation of head words are learned from the user-provided word senses [7]. Rules for selecting a word sense contain a number of constraints for the phrases in the context of the word to classify. Each constraint (match function) determines the syntactic type (noun, verb or prepositional phrase) of a phrase and the token value, semantic type (named/numeric entity type) and word sense of its head word. More general rules contain fewer constraints. The system retains only rules whose precision (ratio of correctly identified word senses) on the training data exceeds a predefined threshold. When several rules fire, the rule with the highest precision wins; when none fires, the most frequent word sense is chosen.

5 Statistical Approaches

5.1 Probabilistic Parsing

The *SIFT* system [35, 36] submitted to MUC-7 is one of the earliest statistical approaches to IE. The system simultaneously handles part-of-speech (POS) tagging and parsing (syntactic annotations) as well as NE recognition and the finding of relationships (semantic annotations), so the results of each task can influence the others. Relationships link two entities of different types, e.g. in *GTE Corp. of Stamford* there is a location-of relation between the company and the city. The system was trained from the Penn Treebank corpus (1,000,000 words) for syntactic annotations and a domain-specific annotated corpus (500,000 words) for semantic annotations.

Tasks are primarily performed at the sentence level. In a final step, entity coreferences are resolved beyond the sentence level (trained from coreference annotations of the semantic corpus) and cross-sentence relationships are established.

The domain-specific corpus requires only semantic tagging (entities, coreferences, relationships), no syntactic annotations. After training the syntax model from the Penn Treebank, it is applied to the domain-specific corpus to produce parses that are consistent with the semantic annotations. The result is a single parse tree that contains both syntactic (e.g. S: sentence, VP: verb phrase) and semantic (e.g. per: person entity, emp-of: employee-of relationship) annotations. The sentence-level model is then retrained on the resulting joint annotations to produce an integrated model of syntax and semantics. Named entities are recognized by a Hidden Markov Model.

The statistical model predicts the categories and POS tags of constituents based on the data from the parse-tree context. The category of a head constituent depends on the category of the parent node; of a modifier on the category of the previous modifier and the parent node and its head constituent as well as on the head word itself. The POS tag of a modifier depends on the modifier itself and on the head word and its POS tag.

The probability of a whole augmented parse tree is the product of the probabilities of all components. The most likely augmented parse tree is found by a chart parser that proceeds bottom-up. Dynamic programming techniques and pruning are used to keep the search space feasible. Maximum likelihood estimates for all probabilities are obtained from the frequencies in the training corpus, using Witten-Bell smoothing to compensate data sparseness.

For determining whether a relation exists between two elements in different sentences, the cross-sentence model calculates the probabilities that a relation does or does not exist and chooses the more probable alternative. These probabilities are calculated on the assumption of feature independence. The considered features comprise *structural features* (the distance between the entities, whether one of the entities was referred to in the first sentence of an article) and *content features* (e.g. whether entities with similar names—probable coreferences—or with similar descriptors are related in other contexts).

The results of the SIFT system were close to those of the best (hand-written) systems in MUC-7.

5.2 Hidden Markov Models

Active Hidden Markov Models: The algorithm developed by Scheffer et al. [48, 49] learns Hidden Markov Models (HMMs) from sparsely (partially) labeled texts. Their HMM algorithm tags each token (word) in a document with one of a set of predefined tags, or the special tag none—the tags (to find) are the hidden states of the Markov Model, while the observed tokens are the visible output of the model. The state (tag) sequence minimizing the per-token error is found using the forward-backward algorithm. Thus the observation sequence *John Smith, extension 7343* should correspond to the state sequence *(firstname, name, none, phone)*. Attributes of each token store the output of preprocessing tools—e.g. the POS tag, word stem, or the surrounding HTML element.

For training the model, partially labeled documents where some of the tags are unspecified are sufficient. The remaining unknown tags are estimated using

the Baum-Welch algorithm. Active learning is used to select the most "difficult" untagged tokens for hand-tagging by the user. The tokens with the lowest difference between the probabilities of the two most probable states are considered most difficult.

HMMs Learned by Stochastic Optimization: The state-transition structure of HMMs is often chosen manually. Freitag and McCallum [24] employ stochastic optimization for this purpose. The algorithm performs hill-climbing starting from a simple model and splitting states until a (locally) optimal state-transition structure has been found. The performance of each model is evaluated on a validation set.

The approach employs a separate HMM for each slot type (e.g. *seminar speaker*) in a document. Each model contains two types of states, *target states* that produce the tokens to extract and *non-target states*.

The Baum-Welch algorithm is used to estimate transition and emission probabilities of each tested model. To increase the reliability of estimated emission probabilities, the technique of *shrinkage* [23] is used. Parameter estimates from sparse states in a complex model are "shrinked" towards estimates from related states in a simpler model where more training data is available for each state (because the number of states is lower). All target states are considered as related, as are all non-target states.

A weighted average learned through Estimation-Maximization is used to combine the estimates of different models. The smoothed, shrinkage-based probability of state s emitting word w is $\lambda_1 P(w|s) + \lambda_2 P(w|a(s)) + \lambda_3 P(\frac{1}{K})$, where the last term represents the uniform distribution, $a(s)$ is the parent state of s (a state combining all target states if s is a target state, a state combining all non-target states otherwise), and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

For learning a suitable HMM structure, non-target states are further differentiated as either *prefix* or *suffix* (preceding resp. following a target phrase) or *background* states (anything else). The most simple HMM fitting this structure has four states (one of each kind) and considers exactly one prefix + suffix around each target state.[1]

This model is used as the starting point for hill climbing. Related models are generated by *lengthening* a prefix, suffix, or target string (adding a new state of the same kind that must be traversed before the model can proceed to the next kind of state), by *splitting* a prefix/suffix/target string (creating a duplicate where the first and last states of the duplicated prefix/suffix/target have the same connectivity as in the original), or *adding* a background state. Model variations are evaluated on a hold-out set or via 3-fold cross-validation.

Hierarchical Hidden Markov Models: Skounakis et al. [51] use *hierarchical HMMs* (HHMMs) [17] for IE. HHMMs combine several levels of states to describe a

[1] The background state is connected to itself and to the prefix state which is in turn connected to the target state, the target state is connected to itself and to the suffix state which is connected to the background state.

sequence at different granularity levels. A two-level HHMM is used—the top level models phrase segments (noun, verb, and prepositional phrases) provided by a shallow parser, the lower level models individual words (including their POS tags) within a phrase.

The Viterbi, Forward, and Backward algorithms are adapted to ensure that the embedded word model reaches the end state exactly at the end of a each phrase and to ensure the typing of the phrase model (each state has a type that corresponds to the type of the phrase segment it emits).

Context hierarchical HMMs (CHHMMs) are an extended variant that incorporate additional sentence structure information in each phrase. The word model is extended to consider the left and right neighbor of each word, generating a sequence of overlapping *trigrams*. To reduce the number of possible observations, individual features (words and tags) are combined under the assumption of conditional independence.

Evaluation shows superior results for hierarchical models, especially CHHMMs, compared with flat HMMs.

Generally, HMMs offer a simple yet powerful way to model text that has proved very successful in various areas of language processing. However, the generative nature of HMMs makes it hard to capture multiple interdependent sources of information. The approaches described in the following section address this problem by switching to sequential models that are conditional instead of generative.

5.3 Maximum Entropy Markov Models and Conditional Random Fields

The *Maximum Entropy Markov Models (MEMMs)* used by [33] are a conditional alternative to HMMs. MEMMs calculate the conditional probability of a state (tag) given an observation (token) and the previous state (tag). Thus the two parts of an HMM—calculating the probability of a state depending on the previous one (transition function) and calculating the probability of an observation depending on the current state (observation function)—are collapsed into a single function.

Observations can comprise many features which need not be independent. Features are binary, e.g. *the word "apple"*, *a lower-case word* etc. The actually used features are selected and weighted by maximum entropy (ME) modeling. Generalized Iterative Scaling (GIS) is used to train the parameters of the model. The most probable tagging sequence is found using a variation of Viterbi search adjusted for MEMMs.

A variation of the Baum-Welch algorithm can be used to estimate missing tags (states) during training, so the model can be trained from partially labeled or even unlabeled documents. No experimental results of doing this are reported though.

A disadvantage of associating observations with state transitions instead of states is the high number of parameters: $|S|^2 \times |O|$ instead of the $|S|^2 + |S| \times |O|$ of

classical HMMs ($|S|$ is the number of states, $|O|$ of observations). This increases the risk of data sparseness.

Tested on a text segmentation task, MEMM performs significantly better than both classical HMMs and a stateless maximum entropy model.

A weakness of MEMMs is the *label bias problem:* the probability mass arriving at a state must be distributed among the successor states, thus outgoing transitions from a state compete only against each other, not against other transitions. This results in a bias in favor of states with fewer outgoing transitions. *Conditional Random Fields (CRFs)* [28, 34] address this problem by modeling the joint probability of an entire sequence of labels in a single exponential model instead of modeling the conditional probabilities of next states in per-state exponential models.

CRFs are undirected graphical models (a.k.a. *random fields* or *Markov networks*) that calculate the conditional of values on designated output variables depending on other designated input variables.

$$P(y|x) = \frac{1}{Z_x} \prod_{c \in C} \Phi_c(x_c, y_c)$$

is the conditional probability of output values y given input values x. $Z_x = \sum_{y'} \prod_{c \in C} \Phi_c(x_c, y_c)$ is the normalizer (partition function), C is the set of all cliques, $\Phi_c(\cdot)$ is the potential function for clique c, x_c and y_c are the sub-sets of the variables in x resp. y that participate in clique c.

CRFs have been employed for preprocessing tasks such as part-of-speech (POS) tagging [28] and for IE subtasks such as coreference resolution [32].

5.4 Token Classification

There are multiple approaches that employ standard classification algorithms, modeling information extraction as a token classification task. These systems split a text into a series of tokens and invoke a trainable classifier to decide for each token whether or not it is part of an slot filler of a certain type (e.g. *speaker* or *location* of a seminar).

Combination Strategies: To re-assemble the classified tokens into multi-token slot fillers, various *combination strategies* (or tagging strategies) can be used. The trivial (*Triv*) strategy would be to use a single class for each slot type and an additional "O" class for all other tokens. However, this causes problems if two entities of the same type immediately follow each other, e.g. if the names of two *speakers* are separated by a linebreak only. In such a case, both names would be collapsed into a single entity, since the trivial strategy lacks a way to mark the begin of the second entity.

For this reason (as well as for improved classification accuracy), various more complex strategies are employed that use distinct classes to mark the first and/or last token of an slot filler. The two variations of *IOB* tagging are probably most common: the variant usually called *IOB2* classifies each token as the begin of a slot filler of a certain type (B-*type*), as a continuation of the previously started

slot filler, if any (I-*type*), or as not belonging to any slot filler (O). The *IOB1* strategy differs from *IOB2* in using B-*type* only when necessary to avoid ambiguity (i.e. if two same-type entities immediately follow each other); otherwise I-*type* is used even at the beginning of slot fillers. While the *Triv* strategy uses only $n+1$ classes for n slot types, *IOB* tagging requires $2n+1$ classes.

BIE tagging differs from *IOB* in using an additional class for the last token of each slot filler. One class is used for the first token of a slot filler (B-*type*), one for inner tokens (I-*type*) and another one for the last token (E-*type*). There are two variations that differ in the handling of slot fillers consisting in a single token (which is thus both begin and end): *BIE1* simply assigns the begin class (B-*type*), while *BIE2* uses a fourth class BE-*type* to mark them specially.[2] Thus $3n+1$ classes are used by *BIE1*, $4n+1$ by *BIE2*.

The strategies discussed so far require only a single classification decision for each token (through often multiple binary classifiers are used concurrently instead of a single multi-class classifier). Another option is to use two separate classifiers, one for finding the begin and another one for finding the end of slot fillers. *Begin/End* tagging requires $n+1$ classes for each of the two classifiers (B-*type* + O for the first, E-*type* + O for the second). In this case, there is no distinction between inner and outer (other) tokens. Complete slot fillers are found by combining the most suitable begin/end pairs of the same type, e.g. by taking the length distribution of slots into account.

Classification Algorithms: There are various approaches that employ a classification algorithm with one of the combination strategies described above: [8] uses *Maximum Entropy (MaxEnt)* modeling with *BIE2* tagging; [59] uses *Memory-based Learning (MBL)* with the *IOB1* strategy; the *TIE* system [50] pairs the *Winnow* algorithm [31] with *IOB2*. Since Winnow is an online algorithm that can learn from a single pass over the training data, *TIE* support *incremental learning*, i.e. the extraction model can be updated on-the-fly without requiring a full retraining. However, better results are reported for batch training (multiple passes over the training data).

ELIE [18, 19] uses two Support Vector Machines (SVM) for *Begin/End* tagging. Highly improved results are reached by augmenting this setup with a second level *(L2)* of begin/end classifiers. The *L2* end classifier focuses on finding suitable end tags for matching left-over begin tags from the level-1 begin classifier (and vice versa). While the *L1* classifiers are trained on a very high number of tokens, almost all of which are negative instances (O), the *L2* classifiers only consider the near context of left-over *L1* begin/end tags which allows a more focused classification. In this way, the recall of the system can be increased without seriously affecting the precision.

While token-classifying approaches lack the genuinely sequential nature of HMMs and conditional models, they have proved very successful (cf. Sec. 6.5),

[2] Note that the actual names used to identify classes do not matter and can deviate from those used in the explanation; what matters is the chosen partitioning of tokens into classes.

due to their ability to combine rich feature representations of the tokens to classify with powerful classification algorithms.

5.5 Fragment Classification and Bayesian Networks

SNoW-IE: Roth and Yih [46] employ the *Winnow*-based *SNoW* classifier in a two-stage architecture. Among a small number of possible candidate fragments identified in the *filtering* stage, the (presumably) correct text fragment is determined and extracted in the *classifying* stage. The two-stage architecture allows using a rich feature representation in a second step for the small subset of promising candidates which would be infeasible (or very inefficient) to use for all possible fragments.

Rich context representations are created by encoding certain relational structures in propositional representations. In the first phase, only single word tokens and POS tags and collocations of two adjoint words/tags (bigrams) are used as features. For words and tags in the left and right context window, the relative position is encoded in the feature. In second phase, "sparse collocations" of words/tags from left and right window and target phrase are also considered. A sparse collocation of n elements generates an n-gram feature for each subsequence of elements $v_i \dots v_j$, $1 \leq i < j \leq n$.

In the version presented in [46], a different classifier is trained for each entity type in each phrase—dependencies between different types are not considered. When several classifiers choose identical fragments for extraction, the more confident classifier (higher activation value) wins.

But relations between entities can yield important hints for determining the exact entity type. Thus the approach has been modified to recognize entities and relations between them at the same time [47]. Borders of entities and existence of relations must be given, but their types are established in a joint step, by maximizing the joint likelihood of all type assignments in a *belief network* (Bayesian network), based on original estimates given by SNoW classifiers.

The mathematical model does not allow loops—different relations are assumed to be independent and entity types are assumed to be independent of relationship types. Another limitation of this approach is that entity borders must be known in advance and cannot be changed.

BIEN: BIEN (Bayesian Information Extraction Network) [41] is another approach utilizing Bayesian networks. For preprocessing, words are lemmatized (stemmed) and POS tagged and sentences are split into flat syntactic chunks (noun, verb, prepositional, and other phrases). Additional features are provided by capitalization, word length, and several gazetteers (location identifiers, popular names).

Dynamic Bayesian networks (DBNs) represent previous decisions to model the order of events ("flow of time"), generalizing Hidden Markov Models. The BIEN system is based on a DBNS that classifies each token as belonging to one of the target field types or to the background (hidden variable Tag). Another hidden variable (Last Target) stores the last recognized target field type, reflecting the

order in which target information is expressed. The Viterbi algorithm is used for classification (determining the most likely sequence of Tag variables); the EM algorithm is used for training the model.

6 Comparison of Selected Approaches

In this section we compare the approaches according to the types of tasks and texts they can handle as well as the types of features they consider. We also compare tagging requirements and learning characteristics. The final subsection discusses quantitative evaluation and presents evaluation results on two standard corpora. Table 1 in Sec. 1.3 can be consulted to locate the detailed descriptions of approaches and systems.

6.1 Types of Tasks Handled

The main IE task is to fill a *template* that contains several *slots*, which is typically done in two steps:

- *Slot filling* or *single-slot extraction* to find suitable fillers for the defined slots.
- *Template unification* or *multi-slot extraction* to combine the found slot fillers into templates, resolving coreferences as required.

Most approaches described in this paper handle the first step only. Hence they are limited to corpora where each document contains a single template; otherwise additional pre- or postprocessing is necessary to split the input at template boundaries or to arrange the found slot fillers into adequate templates.

Some systems—*Crystal*[3], *Whisk* and *TIMES*—handle multi-slot extraction at the sentence level. Thus no special processing is necessary if each template is expressed within a single sentence in a input text. This might be sufficient for some domains but it is not a general solution to the template unification task.

Other approaches go further by unifying templates at a logical level, beyond sentence borders: the *Amilcare* extension of *(LP)*^{2}, *IE*^{2}, *SIFT*, and the extended version of *SNoW-IE*.

A significant difference can be observed in the requirements on target structures: the approaches presented in sections 3.1 and 4.1 generate templates dynamically; all other approaches require a predefined target structure. The dynamic construction of target structures reduces the human effort necessary to adapt a new domain; however, in many cases, automatically deduced structures will be less appropriate than hand-modeled ones.

6.2 Types of Texts Handled

Three types of texts are often distinguished (cf. [54, Sec. 1], [14, Sec. 2.5]):

[3] Crystal does not identify exact slot fillers but only sentence constituents containing slot fillers, thus it always requires postprocessing.

- *Free texts* are grammatical natural-language texts, e.g. newspaper articles or scientific papers.
- *Semi-structured texts* are not fully grammatical and sometimes telegraphic in style, e.g. newsgroups or email messages or classified ads.
- *Structured texts* contain textual information strictly following a predefined (but not necessarily known) format where items are arranged in a fixed order and separated by delimiter characters or strings. Examples are comma-separated values or web pages generated from a database.

Even though some systems are designed for certain types of texts, it cannot be assumed that some class of IE approaches is particularly suitable for a particular kind of text. Furthermore, all classes have in common that the performance on structured texts is better than on free texts.

Some approaches—*TANKA/MaLTe*, the original version of *Crystal*[4], *IE*[2], *TIMES* and *SIFT*—rely heavily on linguistic information and are thus suitable for free texts only. The system described by Embley et al. [15] is restricted to structured texts. Most other approaches are suitable for both free and semi-structured texts—they make use of linguistic information as far as it is available, but do not necessarily require it.

Most other systems make little or no use of linguistic knowledge, thus they are suited for semi-structured and structured texts. *Whisk*, *SRV* and *BWI* claim to be targeted at any text type, from free text to structured text. Approaches that allow variable input will play a major role in the future research, since in real world domains an IE system will be confronted with the large diversity of texts.

6.3 Considered Features

There is a wide variety in the types of features that are considered for learning by different approaches. All systems utilize the words (tokens) in a text as the main lexical features. Not only the presence or absence of a word but also the word order play an important role. Morphological information is used not quite as universally, but very frequently. Especially POS (part-of-speech) tags are used by a wide variety of systems. Some systems also utilize a stemmer or lemmatizer to determine the base forms of words.[5]

For linguistic information beyond the word level, several approaches[6] rely on simple chunkers that identify various types of clauses (noun, verb, prepositional clauses etc.) in a sentence. More refined chunk parsers that also assign grammatical roles for chunks (subject, direct or indirect object) are employed by the systems presented in Sec. 3.1 and by *Crystal* and *Whisk* (for free texts). Only two systems, *SRV* and *TANKA/MaLTe*, make use of a deep parser. Rule and knowledge-based systems tend to embed more syntactic information since

[4] [53] describes an extension to semi-structured text.
[5] *(LP)*[2], *TIE* and *BIEN*, optional for *Active HMMs*.
[6] Such as *TIMES*, *(C)HHMMs*, *TIE*, *BIEN*, and the extended version of *BWI*.

syntax is often used for rule construction. Statistical systems consider predominantly linguistic information related to single tokens due to their token-based processing of the text.

Semantic information is used less frequently than syntactic. Typically, it comprises simple gazetteers or word lists assigning semantic classes to words.[7] Some approaches[8] use a complete thesaurus, WordNet [16]. Knowledge-based systems use their own built-in knowledge-bases.

Some approaches[9] consider features derived from the shape of words/tokens, e.g. token type (lower-case, capitalized, all-caps, digits, etc.) or prefixes and suffixes. Most approaches work on plain text input without formatting, but a few can utilize structural information from HTML or XML documents: *Stalker* and *BWI* can handle HTML tags (treating them as normal tokens), *Active HMMs* optionally consider the HTML context of text tokens, *TIE* creates structured context representations based on the DOM tree of XML documents.

While usually the handled types of features are fixed in advance, the *Amilcare* system chooses an adaptive way to consider linguistic information ("LazyNLP"): the amount of linguistic information available for learning rules is gradually increased until the effectiveness of the generated rules stops improving.

The three main classes of IE approaches differ significantly in the amount of used features. Knowledge-based approaches utilize comparably few features restricting them on semantic and syntactic information. Some statistical systems try to exploit all available information about text elements generating relatively big amount of features. Rule-based systems tend to rely heavily on linguistic features for rule generation.

6.4 Tagging Requirements and Learning Characteristics

Most approaches require training texts to be fully tagged, i.e. all items to extract must be marked (either embedded within the texts or in external documents). Full tagging of a large number of documents is a serious burden. Some systems alleviate this requirement by using *active learning* on partially tagged texts (the extended version of *Rapier*, *Whisk*, *Stalker* in Co-Testing setting, *Active HMMs*). *TIE* is the only system that allows *incremental learning*, i.e. the extraction model can be updated on-the-fly without requiring a full retraining. The approaches described in Sections 3.1, 3.4 and 4 utilize human review and interaction instead of postulating pretagged texts.

The general trend should go towards relaxing the input requirements on the training texts by incorporating better learning models. Statistical systems partially succeed in processing not fully consistent text corpora, while rule-based and knowledge-based systems rely on traditional elaborately prepared text resources.

[7] Used by *Riloff's system*, *Crystal*, *(LP)*^{2}, *TIMES*, *TIE*, and *BIEN* for various word classes.
[8] *Rapier*, *SRV*, *TIMES*.
[9] *SRV*, *BWI*, *MEMM*, *TIE*.

6.5 Quantitative Comparison

Evaluation Metrics: The most commonly used metrics for quantitative evaluation of IE systems are *precision* and *recall*; the joint *F-measure* combines them both in a single figure. For each slot type, results are evaluated by counting *true positives tp* (correct slot filters), *false positives fp* (spurious slot filters), *false negatives fn* (missing slot filters) and calculating *precision* $P = \frac{tp}{tp+fp}$ and *recall* $R = \frac{tp}{tp+fn}$. The *F-measure* is the harmonic mean of precision and recall:

$$F = \frac{2 \times P \times R}{P + R}.$$

For a corpus containing multiple slot types, there are several ways to combine results of all types into a single measure. The *microaverage* is calculated by summing the respective *tp*, *fp* and *fn* counts for all types and then calculating P, R, and F over the summed counts. Thus slot types that occur more frequently have a higher impact on the joint measure than rare types. On the other hand, the *macroaverage* is calculated by computing the mean of all type-specific P and R values, so all types are considered of equal importance, no matter how often they occur.

A disadvantage of the *microaverage* is that is depends on knowing the raw counts, which are hardly ever published in research papers. This is addressed by a related metric, the *weighted average* proposed by [8]: here each slot type is weighted by the total number of slot fillers of this type in the corpus. These numbers can be determined by inspecting a corpus, allowing comparisons with other systems evaluated on the same corpus even if no raw counts have been published.

Evaluation Methodology: As discussed in [29, 30], there are several issues that need to be addressed to allow a fair comparison of different systems, some of which have often been neglected in previous IE evaluations. An important issue is the size of the split between training and testing set (e.g. 50/50 or 80/20 split) and the procedure used to determine partitions (n-fold cross-validation or n random splits).

Another issue is how to compare predicted answers (slot fillers) with the expected (true) answers. Typical options are to require that all occurrences of a slot in a document should be found ("one answer per occurrence") or to expect only a single answer per slot which is considered most likely to be correct ("one answer per slot"). The latter option is useful if multiple answers for the same slot are expected to be synonymous (e.g. "2pm" and "2:00 pm"), the former if each occurrence is assumed to contain relevant new information. A less frequently used option would be "one answer per different string" where multiple occurrences of the same string are collapsed into a single occurrence, i.e. different positions in the document are ignored.

Table 2. F-Measure Results on the Seminar Announcements Corpus

Approach	**Slots**	**BWI**	**ELIE/L2**	**HMM**	**(LP)^2**	**MaxEnt**	**MBL**	**SNoW-IE**	**TIE**
Reference		[22]	[18]	[23]	[9]	[8]	[59]	[46]	[50]
etime	*228*	93.9	96.4	59.5	95.5	94.2	96	96.3	**97.5**
location	*464*	76.7	86.5	83.9	75.0	82.6	**87**	75.2	80.6
speaker	*409*	67.7	**88.5**	71.1	77.6	72.6	71	73.8	85.2
stime	*485*	**99.6**	98.5	99.1	99.0	**99.6**	95	**99.6**	99.3
Weighted Avg		83.9	**92.1**	81.7	86.0	86.9	86.6	85.3	89.9
Macroaverage		84.5	**92.5**	78.4	86.8	87.3	87.3	86.2	90.7

Seminar Announcements Corpus: While there is no universal standard corpus that has been used to evaluate all (or most) IE approaches, several reference corpora have been used quite frequently.

The most frequently used IE corpus is probably the *CMU Seminar Announcements*[10] (SemAnn) corpus. The corpus contains 485 seminar announcements (plain text files) collected from university newsgroups; the task is to extract up to four slots from each document (if present): *speaker*, *location*, *start time (stime)* and *end time (etime)* of the talk.

Generally, training and test sets of equal size are used (50/50 split) and results are averaged over five or sometimes ten random splits. There are no predefined random splits for this corpus, so each system uses their own. The page model published with the corpus prescribes that the "one answer per slot" method should be used. Table 2 lists the best known results published on this corpus.[11] The last two rows contain the weighted average (based on the number of existing slot fillers given in column 2) and the macroaverage.

It is noteworthy that the systems reaching best results on this corpus (*ELIE* and *TIE*) are statistical classification-based systems (cf. Sec. 5.4). There are only two rule-learning systems, BWI and (LP)^2, among the eight best systems, and their performance is inferior to that of the best statistical systems.

Job Postings Corpus: Another frequently used corpus is the *Job Postings* collection of Mary E. Califf [3]. The corpus consists of 300 job offers posted to a Usenet newsgroup. The tasks defines 17 slots of information to extract about job offers (job *title*, *company*, *recruiter*, *salary* etc.) and postings (message *id*, *post date*). Sadly, evaluation methodologies used on this corpus vary wildly. Some authors use 10-fold cross-validation while others use a 50/50 training/test split averaged over 10 random splits. Another ([54]) uses only a subset of 100 randomly selected documents for his tests, while others ([12]) use an extended corpus that contains 600 documents. The original description of the corpus seems to suggest that "one answer per occurrence" is expected but is not quite clear about this.

[10] Accessible from [45], a corrected version with some minor annotation errors fixed is available at `http://nlp.shef.ac.uk/dot.kom/resources.html`.

[11] One other statistical approach, *BIEN* [41], is not directly comparable, since it uses an 80/20 split instead of 50/50. BIEN reaches an weighted average F-measure of 88.9%.

Table 3. F-Measure Results on the Job Postings Corpus

Approach	Slots	ELIE/L2	RAPIER	$(LP)^2$	DeSitter	SNoW-IE
id	*301*	99.7	97.5	**100.0**	97	99.7
title	*252*	**55.8**	40.5	43.9	36	52.7
company	*91*	**79.5**	70.0	71.9	57	75.4
salary	*107*	66.3	67.4	62.8	62	**72.9**
recruiter	*167*	82.0	68.4	80.6	53	**85.3**
state	*235*	**92.7**	90.2	84.7	86	91.7
city	*269*	**95.1**	90.4	93.0	89	89.0
country	*138*	**95.8**	93.2	81.0	95	95.5
language	*516*	**91.4**	80.6	91.0	26	82.5
platform	*469*	79.8	72.5	**80.5**	32	74.1
application	*367*	69.7	69.3	**78.4**	30	60.9
area	*658*	48.7	42.4	**66.9**	16	51.6
req-years-exp	*154*	80.0	67.1	68.8	62	**83.9**
des-years-exp	*44*	82.9	**87.5**	60.4	41	79.0
req-degree	*82*	79.0	81.5	**84.7**	35	83.5
des-degree	*21*	55.2	**72.2**	65.1	35	60.9
post date	*298*	97.5	**99.5**	**99.5**	91	99.2
Weighted Avg		78.6	72.9	**79.8**	49.9	76.4
Macroaverage		**79.5**	75.9	77.2	55.5	78.7

An overview of results reached on this corpus is given in Table 3 (based on [30] and [18]). However, due to the inconsistent evaluation methodologies and testing sets, they must be treated with caution. The statistical *ELIE* system and the rule-learning $(LP)^2$ seem to be very close to each other. However, it is likely that $(LP)^2$ was set up to evaluate only "one answer per slot" instead of the "one answer per occurrence" setup used by *ELIE*. This would explain the apparently better performance on fields such as *platform*, *application*, and *area*, which occur multiple times in many documents. Other fields such as message *id* and *post date* are highly regular (part of the message metadata), which explains the superior results of the rule-based system.

Other Corpora: There are various other corpora, many of which can be found in the *RISE Repository* [45]. The most popular of these is probably the *Corporate Acquisitions* corpus, another corpus which comes from the same source as the Seminar Announcements corpus, the PhD thesis of Dayne Freitag [20]. It contains 600 annotated articles about mergers and acquisitions from the *Reuters-21578* corpus. The task is to extract the full and abbreviated names of the parties to an acquisition, the location of the acquired company, the price paid and information about the status of negotiations. Another important corpus is a collection of *Apartment Rentals* created by Stephen Soderland [54]. However, none of these corpora has been used for evaluation of as many trainable systems as those detailed above.

7 Conclusion

Focus of adaptive methods is quite diverse and reaches from accomplishing subtasks of IE to complete IE systems. Adaptive approaches to information extraction comprise methods that apply experience acquired in training or knowledge gained from external resources (such as thesaurus or user) to establish some kind of a domain model that is capable of locating relevant contents in domain texts and matching them to the target structure of the domain. The structure of the model determines to a large degree the features of an approach and is therefore the main criterion for its classification. Rule-based approaches learn the model as a set of pattern-driven extracting rules, statistical approaches build formal mathematical representations, while knowledge-based approaches establish explicit logical models.

The principle of adaptive approaches originates from the endeavor to reduce the amount of hand-coded domain knowledge. However, the approaches still require human contribution in various forms depending on their class. Human knowledge is utilized as explicit knowledge sources (gazetteers, ontologies etc.), examples specifying what to extract by identifying relevant content in training text or in form of human supervision interacting directly with the system during correction of the extraction proposals. While statistical and rule-based approaches rely on the latter, knowledge-based approaches focus mainly (but not exclusively) on the former. Arguably, the annotation of texts is often less costly than the explicit formalization of knowledge, which makes statistical or rule-based approaches more attractive for adaptation to domains where no explicit knowledge sources are available. On the other hand, knowledge-based approaches allow easier re-use of existing formalizations.

Rule learning approaches try to exploit the regularity in expressions of certain information to find common linguistic patterns that match these expressions. The majority of approaches use rule learning techniques to acquire the patterns. During the learning process the existing incomplete, erroneous or insufficiently general patterns are improved using the feedback of a human supervisor or annotations of the training corpus. The interconnection between the learned patterns and the action transferring the relevant content in the target structure explicitly or implicitly constitutes the learned rule. A frequently found limitation is the rudimentary learning mechanisms that do not provide enough generalization capabilities. Often the shortcomings and weak results are camouflaged by restricting the complexity of the target structure. Developing more profound models will be an important research direction for rule learning methods. Another serious drawback of the actual extraction is that it typically happens within sentence boundaries or within the scope of predefined instances.

Statistical approaches basically reduce the IE task to the prediction problem. In the simple case every text token is classified as some attribute of the target structure or not relevant. Thereby they utilize training data very effectively being able to learn the correct prediction even from quite limited numbers of examples. Knowledge-based approaches are much stronger supported by external resources and consider additionally grammatical parse trees and lookup of semantic classes,

either flat (dictionaries, gazetteers, entity recognizers) or arranged in hierarchies (thesauri, ontologies).

Due to their heavy reliance on linguistic information, knowledge-based approaches are suitable for grammatical texts ("free texts") only, while most statistical approaches can also handle "semi-structured texts" that are not fully grammatical and sometimes telegraphic in style, e.g. newsgroups or email messages or classified ads. Therefore they are more robust with respect to textual irregularities such as typographical errors or ungrammatical text. Statistical approaches typically process a single word at a time and combine the results, while knowledge-based approaches focus on whole sentences. A common trend of both types of approaches is that early systems (*TANKA/MaLTe*, *SIFT*) tend to be more ambitious, while later development narrows down on more pragmatic and realizable goals.

Even though remarkable progress has been achieved in the field of IE in recent time, especially making the systems more autonomous and universally applicable, many problems remain hardly or not yet tackled. The majority of approaches is not able to recognize and extract multiple occurrences of complex facts. The systems either cannot handle complex facts at all assuming the target structure to be a flat slot sequence or it is presupposed that every document contains at most one instance of the same complex fact. The major difficulty arises when fragments of information belonging together are scattered in different sometimes distant parts of a text and it has to be determined whether and which fragment belongs to what complex entity. Even the most advanced approaches can hardly handle unification of partial facts beyond the sentence boundaries. The research challenge will be to find a way of reassembling composite information without establishing a complete logical representation of the text content.

A similar problem with very different background occurs when the same instance of a fact repeatedly appears in different forms. Due to the richness of natural language we can refer to an entity in very many ways. However, only one occurrence containing the most complete information should be extracted. Current systems often do not recognize that text fragments refer to the same fact and extract them as a new found instance. The most straightforward solution not yet practiced would be to embed the mechanisms of coreference resolution in the extraction algorithm and develop strategies for selecting the most appropriate occurrence.

Information extraction suffers from uncertainty of the natural language. Often facts are expressed with a certain degree of tentativeness (e.g. indirect speech: "someone reported that…", "s.o. assumed that…"). In such cases even for humans it is difficult to decide whether the information is factual and hence relevant. An important task will be to determine the degree of reliability of information, which is possible deploying fuzzy methods.

Probably the hardest problem of IE is to identify implicitly expressed information. However, implicitness of information can be of different origin too. Saying "The furious battle between pirates and government troops ended in the crack of dawn as the pirates raised the white flag" we have to consult our world

knowledge and retrieve the fact that white flag is a symbol for a defeat to conclude, who won the battle. Consider, on the other hand, "She lived in Berlin. James was born in the small homonymous town in Texas" where it is sufficient to know the semantics of the word "homonymous" to infer James' place of birth. Surprisingly, some rule-based and statistical systems even now in simple cases can capture the implicit semantics in linguistic patterns resp. statistical context models as if it were an explicitly expressed fact. Two prerequisites must be fulfilled: the fact has to occur somewhere in the text explicitly (comp. "Berlin" in the last example) and there must be sufficiently many examples for this kind of implicit reference. More general solutions, however, would require different approaches which so far are not in sight.

In this context a very important and interesting question is whether the more profound embedding of semantic analysis will contribute to the advance in IE. Recent successes in single-slot information extraction reached by statistical systems that almost completely forgo semantic resources and analysis suggest that it may be dispensable. But does the good performance for the simplest of IE tasks open optimistic perspectives for much more difficult tasks or have the statistical approaches already reached their limit? One development perspective consists in the combination of statistical extraction systems with algorithms for normalization, coreference resolution and instance unification that rely more heavily on external knowledge bases or rules.

Future research will have to face these problems and questions to extend the practical usefulness of IE systems to a broader range of applications.

References

[1] C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz. SRA: Description of the IE2 system used for MUC. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[2] A. Bagga and J. Y. Chai. A trainable message understanding system. In *CoNLL*, pages 1–8. 1997.

[3] M. E. Califf. *Relational Learning Techniques for Natural Language Extraction.* PhD thesis, University of Texas at Austin, 1998.

[4] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 6–11, Menlo Park, CA, 1998.

[5] M. E. Califf and R. J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[6] C. Cardie. A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 798–803. AAAI Press, 1993.

[7] J. Y. Chai and A. W. Biermann. The use of word sense disambiguation in an information extraction system. In *AAAI/IAAI*, 1999.

[8] H. L. Chieu and H. T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*, pages 786–791, 2002.

[9] F. Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from Web-related texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, USA, 2001.

[10] F. Ciravegna and A. Lavelli. LearningPinocchio: Adaptive information extraction for real world applications. In *Proceedings of the 2nd Workshop on Robust Methods in Analysis of Natural Language Data (ROMAND 2002)*, Frascati, Italy, 2002.

[11] R. Collier. Automatic template creation for information extraction, an overview. Technical report, University of Sheffield, 1996.

[12] A. De Sitter and W. Daelemans. Information extraction via double classification. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining (ATEM-2003)*, 2003.

[13] S. Delisle, K. Barker, J.-F. Delannoy, S. Matwin, and S. Szpakowicz. From text to Horn clauses: Combining linguistic analysis and machine learning. In *10th Canadian AI Conf.*, 1994.

[14] L. Eikvil. Information extraction from World Wide Web – A survey. Technical Report 945, Norwegian Computing Center, 1999.

[15] D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddl. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Conference on Information and Knowledge Management (CIKM)*, pages 52–59, 1998.

[16] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[17] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[18] A. Finn and N. Kushmerick. Information extraction by convergent boundary classification. In *AAAI-2004 Workshop on Adaptive Text Extraction and Mining*, San Jose, USA, 2004.

[19] A. Finn and N. Kushmerick. Multi-level boundary classification for information extraction. In *ECML 2004*, pages 111–122, 2004.

[20] D. Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

[21] D. Freitag. Toward general-purpose learning for information extraction. In C. Boitet and P. Whitelock, editors, *Proc. 36th Annual Meeting of the Association for Computational Linguistics*, pages 404–408, San Francisco, CA, 1998.

[22] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *AAAI/IAAI*, pages 577–583, 2000.

[23] D. Freitag and A. K. McCallum. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[24] D. Freitag and A. K. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *AAAI/IAAI*, pages 584–589, 2000.

[25] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[26] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM—semi-automatic creation of metadata. In A. Gomez-Perez and V. R. Benjamins, editors, *Proc. 13th International Conference on Knowledge Engineering and Management*, 2002.

[27] D. Kauchak, J. Smarr, and C. Elkan. Sources of success for information extraction methods. Technical Report CS2002-0696, UC San Diego, 2002.

[28] J. Lafferty, A. K. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[29] A. Lavelli, M. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. A critical survey of the methodology for IE evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 2004.

[30] A. Lavelli, M.-E. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. IE evaluation: Criticisms and recommendations. In *AAAI-2004 Workshop on Adaptive Text Extraction and Mining*, San Jose, USA, 2004.

[31] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[32] A. McCallum and B. Wellner. Object consolidation by graph partitioning with a conditionally-trained distance metric. In *KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[33] A. K. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *ICML*, 2000.

[34] A. K. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI'03 Workshop on Learning Statistical Models from Relational Data*, 2003.

[35] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, R. Weischedel, and the Annotation Group. Algorithms that learn to extract information—BBN: Description of the SIFT system as used for MUC. In *MUC-7*, 1998.

[36] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. A novel use of statistical parsing to extract information from text. In *ANLP-NAACL*, pages 226–233, 2000.

[37] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[38] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with strong and weak views: A case study on wrapper induction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.

[39] U. Y. Nahm and R. J. Mooney. Using information extraction to aid the discovery of prediction rules from text. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, Boston, MA, 2000.

[40] C. Nobata and S. Sekine. Towards automatic acquisition of patterns for information extraction. In *International Conference of Computer Processing of Oriental Languages*, 1999.

[41] L. Peshkin and A. Pfeffer. Bayesian information extraction network. In *IJCAI*, 2003.

[42] J. R. Quinlan and R. M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13(3,4):287–312, 1995.

[43] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press, 1999.

[44] E. Riloff and M. Schmelzenbach. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*, 1998.

[45] RISE repository. `http://www.isi.edu/info-agents/RISE/`.

[46] D. Roth and W.-t. Yih. Relational learning via propositional algorithms: An information extraction case study. In *IJCAI*, 2001.

[47] D. Roth and W.-t. Yih. Probabilistic reasoning for entity & relation recognition. In *COLING'02*, 2002.

[48] T. Scheffer, C. Decomain, and S. Wrobel. Active hidden Markov models for information extraction. In *Proceedings of the International Symposium on Intelligent Data Analysis*, 2001.

[49] T. Scheffer, S. Wrobel, B. Popov, D. Ognianov, C. Decomain, and S. Hoche. Learning hidden Markov models for information extraction actively from partially labeled text. *Künstliche Intelligenz*, (2), 2002.

[50] C. Siefkes. Incremental information extraction using tree-based context representations. In A. Gelbukh, editor, *Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2005)*, Lecture Notes in Computer Science, pages 510–521. Springer, 2005.

[51] M. Skounakis, M. Craven, and S. Ray. Hierarchical hidden Markov models for information extraction. In *IJCAI*, 2003.

[52] S. Soderland. *Learning Text Analysis Rules for Domain-specific Natural Language Processing*. PhD thesis, University of Massachusetts, Amherst, 1997.

[53] S. Soderland. Learning to extract text-based information from the World Wide Web. In *Proc. Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 251–254, 1997.

[54] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1–3):233–272, 1999.

[55] S. Soderland. Building a machine learning based text understanding system. In *Proc. IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.

[56] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In C. Mellish, editor, *Proc. 14th International Joint Conference on Artificial Intelligence*, pages 1314–1319, San Francisco, 1995.

[57] K. Sudo, S. Sekine, and R. Grishman. Automatic pattern acquisition for Japanese information extraction. In *HLT2001*, 2001.

[58] C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proc. 16th International Conf. on Machine Learning*, pages 406–414, 1999.

[59] J. Zavrel and W. Daelemans. Feature-rich memory-based classification for shallow NLP and information extraction. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining, Theoretical Aspects and Applications*, pages 33–54. Springer Physica, 2003.