2 Related Work - Adaptive Approaches to IE

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. Message Understanding Conference stimulated the development of numerous other IE systems by establishing common standards for evaluation and comparison of their performance. The rule-based approach was the driving force behind first IE systems and has been still the most widely employed, enhanced and improved method in the area of information extraction [App93, Cun01]. Its principle is in providing human syntactic and semantic knowledge in form of extraction rules which should be sufficient to handle linguistic diversity in a certain domain. However, it suffers from the fact that rules have to be specified manually, which implies large human effort. Another big disadvantage is that the rules are tailored for a concrete application domain and cannot be reused. For each new domain the whole effort must be repeated. Classical rule-based approach may be appropriate for smaller domains but it does not scale for many practical applications.

To overcome these serious limitations of classical rule-based approach different machine learning techniques are developed and utilized aiming at reduction of manual effort and human knowledge that is necessary for creation of extraction rules. Some approaches abandon the rule-based concept completely and fall back on statistical or knowledge-based methods, which proved themselves in other areas of NLP. Another group of researches pursues the rule-based approach trying to compensate its deficiencies by automating the critical parts. As opposed to statistical techniques (such as neural networks, hidden Markov models) where the knowledge is hidden in the internal model, rule-based approaches specify the knowledge declaratively and explicitly, which makes the analysis and error correction much easier.

Prior to the introduction of state of the art approaches to information extraction this chapter gives an overview over machine learning and learning methods particularly relevant for IE. It is followed by the presentation of different classes of IE systems [Sin05]. Most interesting and distinct representative approaches are described while a closer look is taken on rule-based approaches that are more related to our approach and on statistical systems *ELIE* [Fin04a] and *TIE* [Sie05]

that are used for evaluation of our approach.

2.1 Relevant Machine Learning Techniques

In many scientific areas human expertise cannot be formulated or is still not sufficient to propose an algorithm for solving problems. Machine learning is applied when one does not know exactly how to solve a problem, but sample data and experience are available (e.g. speech recognition, computer vision). Besides, machine learning is applied in domains where the dynamic changes of the environment and the problem context cannot be predicted and a system has to adjust its behavior depending on the current state (e.g. in robotics). IE deals with very complex input data, since the diversity of human language does not make it possible to predict all different kinds of possible expressions for certain information, and is therefore a suitable application domain for machine learning.

Supervised learning can be performed when there is some sample input and output data. Abstracting from the application domain the task of machine learning is to find the mapping from input to the output. Typically some model of the data is defined and its parameters are "learned" (optimized) by a learning procedure. One can see the model as a function, and the goal of machine learning is to minimize its error, that is, function values should be as close as possible to the correct values given in a training set.

The main difference between *unsupervised* and supervised learning is that the correct output values are not known (no training set is provided by the supervisor). The main task of unsupervised learning is to find regularities and patterns in the input data and to use this knowledge for solving problems in the application domain. A typical problem solved by unsupervised learning is clustering. Finding clusters or groups of similar instances is successfully employed to perform text classification.

In many domains the expected output is not represented by data but by a sequence of actions. The actions are performed to reach a global goal and therefore the correct order of the actions is much more important than the goodness of a single action. Learning to estimate the quality and efficiency of a sequence of actions and learning from past good sequences to derive new sequences is called *reinforcement learning*. The examples of application areas are game playing and robot navigating where the sequence of correct moves helps to win or reach the destination.

Machine learning in information extraction

Considering the problems and complexity of IE supervised learning appears to be the most appropriate and is the most widely used learning technique in this field. The task of IE involves identification of relevant content in text. Supervised learning methods rely on the training examples to obtain features characterizing the relevant information. Unless the IE system is bound to a fixed domain, the only known reliable way of identifying training instances is to use human understanding of texts. Thus when we try to employ unsupervised learning for the IE we are confronted with even harder problem of text understanding. Consequently, one of the most significant advantages of supervised learning is that it allows to circumvent the problem of text understanding utilizing the training examples that were obtained by human.

In the following paragraphs the supervised learning techniques especially significant for IE are discussed.

2.1.1 Classification

Classification is the general problem of deciding whether an input instance belongs to some of the predefined classes. Typically, when building a classifier one describes the input data by a model - a view on the data that focuses on the aspects especially relevant for the distinction of different classes. An important and difficult task is to identify attributes of the data that best discriminate the data, that is the presence or absence of an attribute, its value or attribute combination is characteristic for a certain class. If the input has two features (attributes), all input instances can be represented as points in a two-dimensional space. In case of a binary classifier the task is to find a function which separates the positive instances from the negatives – *the discriminator*. In a simple case the classes can be separated by a linear function. If classes are not linearly separable a polynomial function can be used. For multiple features a discriminator has to be found in a multidimensional space. If more than two classes have to be learned, the binary classifier for every class can be used cascadingly.

Reducing the problem of information extraction to a classification problem every attribute of the target structure is declared as a class. Every token in text is classified as belonging to a certain class or irrelevant. Extractions comprising several tokens can be handled by defining two classes for each attribute: one for beginning or continuation of extraction and one for its end. Such an approach presumes however that there is only one attribute value per attribute in the text and is appropriate for slot filling. To extract relations additional classifiers have to be incorporated in order to decide what partial expressions belong together or the relations have to be determined heuristically.

2.1.2 Determining the success of learning

Assume that for some classification problem a linear discriminator is proposed and we can derive a rule to decide for an input instance whether it belongs to a class comparing the function value of the discriminator with the feature values of the input. Such a rule is called a *hypothesis*. The complexity of the hypothesis (e.g. using a linear or quadratic discriminator) determines the number of the points that can be separated by this hypothesis. Choosing the appropriate hypothesis class is called model selection [Alp04]. If a model is too simple, that is if the hypothesis is less complex than the function underlying the data, then *underfitting* takes place. If the model is too complex, we have *overfitting*. In this case the data cannot sufficiently constrain the hypothesis or the hypothesis may learn not only the underlying function, but also the noise in the data. More training data helps only up to a certain point.

The aim of learning a hypothesis is to minimize the error of classification of the training input by adjusting the parameters of the model. The hope is that the best hypothesis for the training data will also perform best for the data outside the training set. To assess the quality of a hypothesis the training set is divided into two parts. One part is used for training (learning the parameters of the model) and another is called validation set and is used to test the ability to predict the right output for new instances. If the training and validation sets are large enough, the hypothesis that produces the most accurate output on a validation set is the best. This process is called *cross-validation* and is the usual

method to estimate the goodness of a learning approach, which is also employed in other fields of machine learning.

2.1.3 Rule Learning

Statistical approaches use mathematical models (feature vectors, hidden Markov models, conditional probabilities etc.) adjusting it during the learning process to reflect the characteristics of the possible domain input represented by the training data. The input features are expressed by the inner parameters of the model. Rule-based approaches capture the input features at a significantly higher level using explicit representation in form of rules as the learning model. A *rule* consists of a condition and an action. As soon as the condition is fulfilled, the action is executed. Conditions can be expressed logically or just as some kind of patterns. Actions can add or remove rules, modify the parent rule, set some parameters in the configuration, output the result values etc. The big advantage of rules is that they are declaratively specified [Neu99] so that the state of the model is not hidden in the numeric parameters of some mathematical structure but can be comprehended at every moment during runtime. The whole learning process can be reconstructed for better comprehension and interpretation of resulting model and test results. However, it is not so easy to extend the model by a parameter as in statistical approaches, because it involves altering the rule heuristics, which in turn requires changes at the implementation level.

2.2 Rule-Based Approaches

As their name suggests, rule-based approaches rely on rules that explicitly encode how the relevant information has to be localized in the text and extracted. Expression patterns or a set of conditions specifying the relevant fragments and/or their contexts serve for identification of relevant content. Besides, a rule prescribes how the identified fragments are handled, i.e. defines a processing action for extracted values.

2.2.1 Automatic pattern and template creation

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. As representatives for this approach Nobata's and Sekine's system for pattern acquisition and a successor of *AutoSlog-TS* that also accomplishes template creation [Ril98] are considered.

Nobata and Sekine introduced a system for automatic pattern acquisition for Japanese [Nob99]. After selecting relevant articles by keywords and performing POS tagging and NE-recognition they regard each sentence as initial pattern. The pattern is represented by an ordered sequence of lexical items and name entities; in a later work [Sud01] a tree representation for patterns is proposed to better account for syntactic dependencies and free word order problem. Only sentence types that occur more than once and contain at least one of predefined named entities are merged to obtain a more general pattern. The authors use

quite primitive merging techniques clustering different items on the same position: (A B C) +(A D C) = A (B | D) C and omitting non-common elements: (A B C) + (A C) -> A ignore C. The pair of most similar patterns is merged and the original patterns are discarded. The result is a set of patterns that should identify sentences containing desired information. The system tries to solve only the problem of localizing information on a sentence level since nor determination of facts in the sentence is performed neither items are extracted. Primitive merging technique does not allow to achieve an acceptable degree of generalization of patterns and thus has a negative effect on recall unless all possible types of sentences were in the initial set. Besides, taking only lexical features and POS-tags into account restricts the generalizing capabilities even more, since nor structural, nor syntactic, neither semantic features are considered. Authors do not provide a clear termination condition so that the optimal pattern set has to be determined empirically.

Riloff's system is guided by the assumption that the items to be extracted are exclusively comprised by noun phrases (NPs). Therefore it uses heuristics to create linguistic patterns that represent relevant context for extracting of a given noun phrase. Typical examples of patterns are *<subject> exploded* , *exploded in <noun-phrase>*. 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. The pattern score depends on the number of extracted NPs also found in the semantic domain lexicon and on their percentage among all extracted NPs. After human review the best patterns are selected.

As opposed to our notion of information extraction Riloff's system does not presuppose a priori defined target structure. It rather constructs it dynamically depending on the text content. After acquisition of patterns described above they are assigned to semantic categories depending on which semantic category prevailed among the extracted items (semantic category of an item is determined in the semantic lexicon). A very important semantic information that contains the functional dependency of template slots on semantic categories is provided by the human. Using this dependency a pattern can be unambiguously assigned to extract a certain template slot. However, a pattern can extract items of different semantic categories and thus different slot values. Since the prevalent semantic category is chosen, many extractions of items with different categories would be assigned a wrong slot, so that the quality of extraction suffers. 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 several slot values 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 available slots. Giving the system the set of slots only sketches the scope of possible target structure while its actual creation is based on described algorithms.

The very simple syntactic structure of patterns cannot recognize many relevant facts that are expressed in a complex linguistic context with complicated syntactic structure. Facts expressed over multiple sentences remain also uncovered. 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 slot wrong slots may be generated, which

leads to an adulterated target structure. Another big problem is unknown words (not occurring in the training texts and therefore absent in the semantic lexicon) because no slot can be assigned to them and hence the item will not be extracted.

2.2.2 Covering algorithms

Learning patterns from the annotated training corpus is the primary goal of covering (separate & conquer [Für99]) algorithms. Some of them use a part of training corpus to obtain initial definitions of relevant content (separate) and improve them on the remaining part (conquer). *Crystal* [Sod95, Sod97] learns simple patterns consisting of sequences of syntactic constituents or semantic classes. All words in the domain have to be classified in a lexicon. Generation of initial patterns is based on finding the syntactic constituent with a certain semantic class. Initial patterns are highly constrained. Crystal tries to unify the patterns by relaxing the constraints of similar patterns. The similarity metric is based on the number of constraint changes necessary for the unification and not by the information recognized by the patterns. The unification heuristics are very simple: unifying semantic classes results in the most specific common ancestor, unifying two word sets – in the common subset. Extraction of composite facts is handled insufficiently because every possible subset of their attributes serves as a concept to be learned, which results in data sparseness. The system is not able to extract exact fragments, but only determines the syntactic constituent to extract from. Crystal does not support negation of constraints.

The successor system Whisk [Sod99] introduces ordering constraints for attribute values extracting several values at once at a sentence level to compensate the lack of negation. It is targeted at handling a broader range of texts, from free texts as found in newspapers and books to semi-structured texts (often ungrammatical or in "telegram style"). The learned rules utilize a subset of regular expressions. Expression pattern can contain verbatim text, character classes (e.g. digit, number) and wildcards "*", which lazily skips any characters until the next part of the pattern can match. Whisk uses the Laplacian expected error: $Laplacian = \frac{e+1}{n+2}$ to assess the quality of rules, where n is the number of made extractions and e is the number of errors among them. The initialization of most general rule and the following top-down derivation face the limitations of hill climbing, because each specialization is evaluated in isolation. As opposed to Riloff's approach semantic classes are not learned by the system, but must be specified by the user in advance. Though integrating elements of regular expressions makes the patterns more expressive than in Crystal, the lack of combination of syntactic and semantic features restrict their effectiveness.

Whisk incorporates *active learning* [Coh96] reducing the amount of initial training data. The system presents three kinds of untagged instances for interactive annotation 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).

$(LP)^2$ [Cir01a] is a covering algorithm that learns annotating rules. A rule inserts an SGML tag in the text highlighting a beginning or an end of relevant information if its set of conditions is fulfilled. The conditions specify words in a fixed context around the placed tag and provide shallow linguistic information about them. Principal form of a word (lemma), part of speech, capitalization and semantic categories obtained from a user-defined dictionary are captured. Initial rules are built from positive examples in the training corpus provided by

a human annotator. They are generalized relaxing the strict initial conditions. Generalization is achieved by reducing the length of context and omitting certain specifications (e.g. requiring only a semantic category instead of specific word). The k best generalizations of each initial rule found by a beam search are stored in a best rules pool. $(LP)^2$ proceeds in four steps:

1. The *tagging rules* from the "best rules pool" are applied.
2. The inserted tags are used as new context constraints and enable better quality of rules that were not included in the best rules pool. Such *contextual rules* perform better when restrained to the close environment of tags placed in the first step (for example, a rule that inserts an end tag will be applied if a corresponding start tag occurs some words before).
3. *Correction rules* change the position of misplaced tags. They use the same contextual model as the contextual rules and are obtained by a similar learning algorithm as the "best rules".
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 [Cir02]. 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. localization of a person name might require more NLP input than identification of dates or times).

$(LP)^2$ is restricted to extraction of single attribute values and not able to recognize relations. In Amilcare a shallow discourse representation module is added for this purpose [Han02]. Slot fillers are unified in templates or subtemplates with the nearest preceding slot fillers of a suitable type. For instance, 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*.

2.2.3 Relational Learners

Another branch of covering algorithms is represented by "relational" algorithms [Fre98]. The notion of pattern is described by a very big (sometimes even unrestricted [Cal03]) relation of features, which include character format, position, length, POS tag etc. Since the features are collected for every token and the context window is not restricted the whole document can serve as a pattern for extraction. However, there is much less semantic information integrated in comparison with the systems described above, these approaches try to detect regularities and patterns in the natural language by acquiring big amounts of simple statistical properties.

The system *Rapier* [Cal98] utilizes POS tags and WordNet semantic classes as features to derive rules for extraction of single attribute values. A rule consists of three parts, a pre-filler pattern, a pattern for the actual slot filler and a post-filler pattern. Each pattern contains an ordered list of constraints that restrict the POS tag, the semantic class and the word itself. Training examples represent most specific rules specifying all possible constraints. The pre- and post-filler

patterns of training examples contain every word from the start to the end of the document, there is no "context window" of limited length.

In contrast to Crystal no similarity metric is defined, random selection of rules is used instead for creating the least general generalization for the filler pattern. Several reasonable generalizations are possible (different values of a constraint can be disjointed 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 disjunctions in a rule. Semantic classes are generalized by finding the nearest common ancestor in the WordNet hypernym hierarchy (omitting the constraint if no common ancestor exists). Training 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 support *active learning* [Tho99]. After initial training with a small number of annotated examples the system attempts to annotate a large number of untagged examples, selecting examples with the least certain annotation (certainty-based selective sampling). After the user corrected the annotations of the selected examples, the system is incrementally retrained and the interactive learning process continues. Rapier does not use probabilistic evaluation, 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 amount of incorrect ones. The active learning version requires approximately half the examples to reach the performance level of Rapier without active learning.

SRV [Fre98] 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 carries out only a binary classification, i.e. different rule sets are learned for classifying each text fragment as an instance or non-instance of a single attribute – there is no component for relation extraction or other postprocessing.

The learning algorithm is similar to the relational rule learner FOIL [Qui95]. SRV learns top-down greedily extending rules by 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.

In the validation step the accuracy of the rules is estimated by three-fold cross validation. The three resulting rule sets are then merged. The accuracy estimations are available for each prediction.

Relational approaches are distinguished by a rich and more flexible context representation than most other rule-learning and statistical approaches. The downside is a limited feature space that negatively affects the expressivity of extraction rules. Besides, the large space of possible rules can lead to high training times and there is no guarantee of finding optimal rules due to local maxima problem.

Wrapper Generation Many commercial web pages are generated automatically from a database (e.g. product and price listings, flight schedules) and presented in a structured way to human readers. In a business context it is often useful to reconstruct the database content extracting the attribute values of database records from the generated pages. While earlier approaches to wrapper induction (e.g. *Stalker* [Mus01]) required a set of annotated training examples to recognize the common schema, more recent algorithms such as *RoadRunner* ([Cre01, Cre03]) and *ExAlg* [Ara03] are able to identify common structural information from a set of unlabeled documents, avoiding the tedious manual annotation phase. Such approaches are extremely useful for "reverse-engineering" of highly structured texts, but manually created documents are outside their scope. The *BWI* system [Fre00a] is conceived to handle both kinds of texts.

Case-based Extraction Case-based learning techniques are used by Cardie [Car93] to determine the part of speech, the meaning and the semantic category of a word. A case comprises the POS tag, the general and special word meanings and the context features. The case base is populated with cases created by the user. When inserting a new word, case base is queried and the most frequent features of the retrieved cases are assigned. Even though the result for this task were promising the algorithm can hardly be adapted to identify relevant content in bigger text fragments.

2.3 Knowledge-based and Statistical Approaches

A number of researchers choose the knowledge-based approach to accomplish tasks closely related to IE. The Delisle's system [Del94] extracts the entire text content and transforms it into Horn clauses. Although there is no target structure, the formal logical representation structures the information and allows logical queries. The big drawback, however, is that human has to solve difficult tasks in almost every stage of processing while assisting the system. Further approaches presuppose an extensive knowledge base in the form of an ontology [Emb98, Ait02] or thesaurus [Bag97]. It serves as the main source for semantic and syntactic knowledge and contains the description of the application domain. Knowledge base is used as well for searching for relevant information as for the verification of correctness.

Statistical approaches

Current statistical approaches model the information extraction task according to their underlying statistical technique. In multiple systems the sequential tagging is performed by trained Hidden Markov Models [Fre99, Sko03]. Scheffer et al. [Sch01, Sch02] model the tags as hidden states of the Markov Model. The state (tag) sequence minimizing the per-token error is found using forward-backward algorithm. While state transition in HMMs is often defined manually, Freitag & McCallum [Fre00b] employ stochastic optimization that performs hill-climbing

starting from a simple model and splitting states until a (locally) optimal state-transition structure has been found. Some approaches [Sch02, McC00] embed active learning based on Baum-Welch algorithm, so that complete annotation of training corpus is not necessary.

The *Maximum Entropy Markov Models (MEMMs)* used by [McC00] are a conditional alternative to HMMs. MEMMs calculate the conditional probability of a state (tag) given an observation (token) and the previous state (tag) merging the transition and observation functions of Markov models into a single function. The most probable tagging sequence is found using a variation of Viterbi search adjusted for MEMMs. 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)* [Laf01a, McC03b] 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. They have been employed for preprocessing tasks such as part-of-speech (POS) tagging [Laf01a], coreference resolution [McC03a] and information extraction [Sut05].

In contrast to sequential tagging of HMMs many approaches rely on classification of single words or tokens. Each token is classified whether it belongs to some type of the target structure. Since an attribute value may consist of several tokens, the boundaries of the extracted fragments have to be distinguished. Using a single class for each attribute does not allow to distinguish between two different attribute values immediately following each other. Therefore separate classes are usually used to tag the beginning (*B-type*)and continuation *I-type* of an attribute value (*IOB* classification) or additionally its end (*BIE* classification). In both cases one separate class (*O*) represents all tokens that are not assigned to any of the attributes. While both IOB and BIE require only one classification of each token, two different classifiers determining the beginning and the end of an extraction can be used alternatively. Attribute values are identified as the most likely begin/end pair of tokens of the same type that is determined, for example, considering the length distribution of attributes.

The both classification strategies are applied in combination with standard classification algorithms: [Chi02] uses *Maximum Entropy (MaxEnt)* classifier with *BIE* tagging; [Zav03] investigates *Memory-based Learning (MBL)* with the *IOB* strategy; Dynamic Bayesian Networks are used by [Pes03], while Roth and Yih propose an approach combining Bayesian networks with Winnow classification algorithm [Rot01].

The *TIE* system [Sie05] pairs the *Winnow* algorithm [Lit88] with *IOB2*. Winnow is a linear separator, but by providing a feature space that itself allows conjunction and disjunction, complex non-linear features may be recognized by the composite feature-extractor and Winnow system. For this purpose the so called *orthogonal sparse bigrams* (OSB) are introduced. Unlike conventional bigrams OSB allows to generate joint features not only between the neighbor tokens but also between all tokens in a window with a fixed length. A hierarchical tree structure obtained during preprocessing is used as the context model. It includes morphological and syntactic as well as some statistical information about the

token itself and up to four preceding and following neighbors on the POS level and also the ancestors of the token and their neighbors in the XML hierarchy. Additionally, the semantic classes the token belongs to are listed that are obtained from a collection of optional gazetteers and dictionaries. Beside context specification information on the last four attribute values found in the current document is provided. The complete description of each token and its context results in a fairly high number of features. They are arranged in an ordered list to allow recombination via OSB; the resulting feature vector serves as input for the classifier.

Since Winnow is an online algorithm that can learn from a single pass over the training data, *TIE* supports *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 [Fin04a, Fin04b] uses two Support Vector Machines (SVM) for *Begin/End* tagging. Besides, a sophisticated extension of standard classification is introduced augmenting this setup with a second level *(L2)* of begin/end classifiers. The purpose of *L2* end classifier is to identify appropriate end tokens for "orphaned" (i.e. not matched) beginning tokens from the B-classifier of the first level 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 take only the near context of orphaned *L1* begin/end tokens into account, which allows a more confident classification. The extension contributes to a considerable improvement of the coverage (recall) without seriously penalizing the precision.

2.4 Deficiencies of state of the art approaches

In spite of advancing research many approaches to IE suffer from substantial deficiencies that restrict their practical applicability. Multiple systems depend to a large degree on the manually created semantic resources. In relatively small application domains that require a reasonable effort for the creation of such resources the success of information extraction is significantly connected with the goodness and the completeness of provided semantic information. Evaluation of results in such isolated domains is quite misleading because in bigger "real world domains" both the contribution of initially provided semantic resources to the goodness of extraction diminishes and the quality of the resources can hardly be maintained without considerable human effort. The kind of investigated texts has also a big impact on the extraction results. Sometimes texts are used that feature a very rigid structure or a certain format (e.g. newspaper classifieds). The desired information can be much easier localized in such texts than in documents with much free text because the semantics is to some extent provided by the text formatting.

Rule based systems can often generate patterns for and process only trivial syntactic structures. In many cases patterns are specified as simple sequences or as a subset of regular expressions so that a large part of information which is expressed in a linguistically non-trivial way cannot be captured. Some systems do not accomplish the actual extraction, but only identify the syntactic parts of the sentences (or even sentences) that contain the relevant information. In this case the objective of IE to structure the information and enable queries is missed. Knowledge based systems presuppose heavy resources that are not automatically created. Particularly for concrete application domains they are hardly available.

The common disadvantage of statistical systems is that due to the token-based view of the text they offer no possibilities to unify partial information, which is part of complex facts. Lack of statistical approaches that are able to tackle the most challenging problems of information extraction (in particular confident identification of complex, distributed information) suggests that statistical systems at the current point of development are not very suitable for tasks where deeper semantic analysis is necessary, since it is difficult to express semantics statistically. It is much simpler to describe a single word or token by statistical features rather than a sentence or even the whole document. This statement is also supported by fact that statistical methods are very successfully employed in other areas of NLP like POS-Tagging, word sense disambiguation, syntactic analysis, NE recognition etc. – tasks where the semantic component can be supported by other features or where it plays a subordinate role (Manning and Schütze come to a similar conclusion, [Man99, p. 8]).