4 Statistical Approaches

4.1 Probabilistic Semantic Parsing

The *SIFT* system [Mil98, Mil00] 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 named entity (NE) recognition and the finding of relationships (semantic annotations), so the results of each task can influence the others. Relationships link two attribute values 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, coreferences between attribute values 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 (attribute values, 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 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 [Wit91] 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 attribute values, whether one of the attribute values was referred to in the first sentence of an article) and *content features* (e.g. whether attribute values 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.

4.2 Hidden Markov Models

Hidden Markov Models (HMM) are one of the earliest and most frequently used techniques in the area of statistical language processing. They have been used in various ways in the field of information extraction and have lead to several spin-off techniques such as the MEMMs and CRFs that will be presented in the next section.

The algorithm developed by Scheffer et al. [Sch01, Sch02] employs *active learning* (cf. Sec. 3.3) to learn 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 sequence minimizing the per-token error is found using the forward-backward algorithm.

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.

The state-transition structure of HMMs is usually chosen manually. Freitag and McCallum [Fre00b] 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 attribute (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*. As usual, the Baum-Welch algorithm is used to estimate transition and emission probabilities of each tested model; the estimates of different models are combined using a weighted average learned through Estimation-Maximization. For learning a suitable HMM structure, non-target states are differentiated as either *prefix* or *suffix* (preceding or 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.

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 cross-validation.

Another variation that has been used for IE combines several levels of HMMs in a *Hierarchical HMM* (HHMM) [Fin98, Sko03]. Text sequences are modeled at two different granularity levels in a a two-level HHMM: 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 each phrase and to ensure the typing of the phrase model.

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 [Sko03].

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.

4.3 Maximum Entropy Markov Models and Conditional Random Fields

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). 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 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, MEMMs perform significantly better than both classical HMMs and a stateless maximum entropy model [McC00] .

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)* [Laf01, 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.

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

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 and y that participate in clique c.

CRFs have been employed successfully for information extraction [Sut05] as well as for related tasks such as coreference resolution [McC03a].

4.4 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 attribute value (e.g. *speaker* or *location* of a seminar).

4.4.1 Tagging Strategies

To re-assemble the classified tokens into multi-token attribute values, various *tagging strategies* can be used. A popular strategy is to classify each token as the begin of a value of a certain attribute (B-*type*, where *type* is the name of the attribute), as a continuation of the previously started attribute value, if any (I-*type*), or as not belonging to any attribute value (O). This strategy is usually called *IOB2* tagging; there is a variation called *IOB1* tagging that uses B-*type* only when necessary to avoid ambiguity (i.e. if two values of the same attribute immediately follow each other); otherwise I-*type* is used even at the beginning of attribute values.

BIE tagging differs from *IOB* in using an additional class for the last token of each attribute value. One class is used for the first token of an attribute value (B-*type*), one for inner tokens (I-*type*) and another one for the last token (E-*type*). A fourth class BE-*type* is used to mark attribute values consisting of a single token (which is thus both begin and end).

The strategies discussed so far require only a single classification decision for each token. Another option is to use two separate classifiers, one for finding the begin and another one for finding the end of attribute values. *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 attribute values are found by combining the most suitable begin/end pairs of the same attribute, e.g. by taking the length distribution of attribute values into account.

4.4.2 Classification Algorithms

There are various approaches that employ a classification algorithm with one of the tagging strategies described above: [Chi02] uses *Maximum Entropy (MaxEnt)* modeling with *BIE2* tagging; [Zav03] uses *Memory-based Learning (MBL)* with the *IOB1* strategy.

The *ELIE* system [Fin04a, Fin04b, Fin06] uses *Support Vector Machines (SVMs)* 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 first-level (*L1*) begin classifier, and the *L2* begin classifier matches left-over end tags.

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. Hence the *L1* classifiers must be tuned to favor precision over recall to avoid producing lots of false positives (spurious extractions) from all over text, but the *L2* classifiers can be tuned to favor recall over precision since they only classify a very small subset of all the tokens. In this way, by adding the second level the recall of the overall system can be increased without overly hurting the precision.

While token-classifying approaches lack the genuinely sequential nature of HMMs and conditional models, they have proved very successful, due to their ability to combine rich feature representations of the tokens to classify with powerful classification algorithms. We will return to this approach in Chapter 10 with a more detailed discussion of token classification and tagging strategies.

4.5 Fragment Classification and Bayesian Networks

The *SNoW-IE* system introduced in [Rot01] employs the *Winnow* (cf. Sec. 11.1) 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 [Rot01], a different classifier is trained for each attribute in each phrase—dependencies between different attributes are not considered. When several classifiers choose identical fragments for extraction, the more confident classifier (higher activation value) wins.

But relations between attribute values can yield important hints for determining the exact attribute. Thus the approach has been modified to recognize attribute values and relations between them at the same time [Rot02]. Borders of attribute values 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 *Bayesian network* (belief network), based on original estimates given by SNoW classifiers.

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

BIEN (Bayesian Information Extraction Network) [Pes03] is an approach based on a different variant of Bayesian networks: *Dynamic Bayesian networks* (DBNs) represent previous decisions to model the order of events ("flow of time") in a generalization of Hidden Markov Models. The BIEN system is based on a DBNS that classifies each token as belonging to one of the target attributes or to the background (hidden variable Tag). Another hidden variable (Last Target) stores the last recognized target attribute, reflecting the order in which target information is expressed.