A Hybrid Machine Learning Approach for Information Extraction from Free Text

Günter Neumann*

LT–Lab, DFKI Saarbrücken, D-66123 Saarbrücken, Germany

Abstract. We present a hybrid machine learning approach for information extraction from unstructured documents by integrating a learned classifier based on the Maximum Entropy Modeling (MEM), and a classifier based on our work on *Data–Oriented Parsing* (DOP). The hybrid behavior is achieved through a voting mechanism applied by an iterative tag–insertion algorithm. We have tested the method on a corpus of *German* newspaper articles about company turnover, and achieved 85.2% F-measure using the hybrid approach, compared to 79.3% for MEM and 51.9% for DOP when running them in isolation.

1 Introduction

In this paper, we investigate how relatively standardized ML techniques can be used for IE from free texts. In particular, we will present a *hybrid ML approach* in which a standard Maximum–Entropy Modeling (MEM) based classifier is combined with a tree-based classifier based on Data–Oriented Parsing (DOP), a widely used paradigm for probabilistic parsing. The major motivations for the work presented in this paper are 1) to explore, for the first time, the benefits of combining these two leading ML paradigms in NLP for information extraction, and 2) to exploit ML–IE approaches for German documents. This issue is of interest, because so far, nearly all proposed ML–IE approaches are considering English documents (in fact, we are not aware of any results reported for German using a ML–IE approach using a comparative IE–task). However, since German is a language with important different linguistic phenomena compared to English (e.g., rich morphology, free–word order, word compounds), one cannot simply *transpose* the performance results of ML–IE approaches obtained for English to German.

The core idea of a supervised ML–IE approach from free text is simple (see also fig. 1): Given a corpus of raw documents annotated only with the relevant slot–tags from the template specification, enrich the corpus with linguistic features automatically extracted by the **Linguistic Text Engine**. Pass this annotated corpus to the **Machine Learning Engine** which computes (through the application of its core learning methods) a set of **template**

* Thanks to Volker Morbach for his great help during the implementation and evaluation phase of the project. This work was supported by a research grant from BMBF to the DFKI project `Quetal` (FKZ: 01 IW C02).

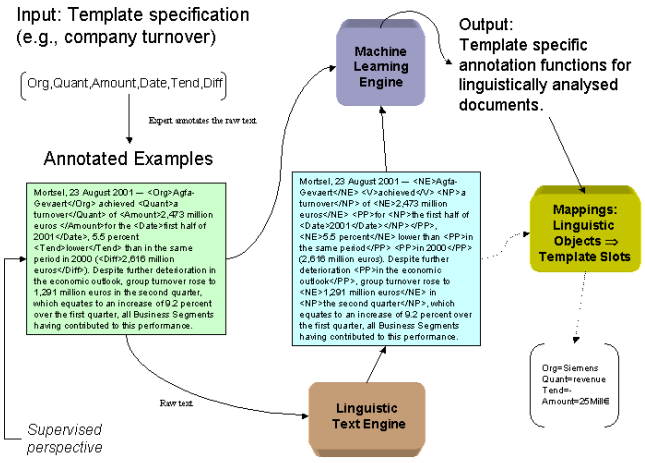

Fig. 1. Blueprint of the Machine Learning perspective of Information Extraction.

specific annotation functions, i.e., mappings from linguistic features to appropriate template slots. These *learned* mappings are then used to **automatically annotate new** documents – pre-processed by the same Linguistic Text Engine, of course – with template specific information.

We are following the standard view of IE "as classification", in that we classify each token to belonging to one of the slot–tags or not. In particular we want to explore the effect of the linguistic feature extraction to the performance of our ML–IE approach. The linguistic features are computed by our system SMES a robust wide-coverage German text parser, cf. Neumann and Piskorski (2002). The features can roughly be classified into lexical (e.g., token class, stem, PoS, compounds) and syntactic (e.g., verb groups (VG), nominal phrases (NP), named entities (NE)). In order to explore the effects of features from different levels, classification is performed as an incremental tagging algorithm, on basis of the following *two–level learning approach*: 1) Token level (cf. sec. 2): each token is individually tagged with one of the slot–tags using only lexical features. 2) Token group level (cf. sec. 3): a sequence of tokens is recognized and tagged with one of the slot–tags by applying a set of tree patterns. Both levels are learned independently from each other, but they are combined in the application phase, and this is why we call our ML–IE approach *hybrid*.

2 MEM for Exploiting the Token Level

The language model for the token level is obtained using Maximum Entropy Modeling (MEM). The major advantages of MEM for IE from unstructured texts are 1) that one can easily combine features from different linguistic

levels, and 2) that the estimation of the probabilities are based on the principle of making as few assumptions as possible, other than the constraints on feature combination and values are imposing, cf. Pietra et al. (1997). The probability distribution that satisfies these properties is the one with the highest entropy, and has the form

$$p(a|b) = \frac{1}{Z(b)} \cdot \prod_{j=1}^{n} \alpha_j^{f_j(a,b)} \text{ with } Z(b) = \sum_{a \in A} \prod_{j=1}^{n} \alpha_j^{f_j(a,b)} \tag{1}$$

where a refers to the outcome (or tag) and A the tag set, b refers to the history (or context), and $Z(b)$ is a normalization function. Features are the means through which an experimenter feeds problem-specific information to MEM (n lexical features in our case), all of them bearing the form

$$f_j(a,b) = \begin{cases} 1 & \text{if } a = a' \text{ and } cp(b) = true \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where cp stands for a contextual predicate, which considers all information available for all tokens surrounding the given token t_0 (our context window is $[t_{-2}, t_{-1}, t_0, t_{+1}, t_{+2}]$) and all information available for t_0. We use the following lexical feature set: token, token class, word stem, and PoS. The task of the MEM training algorithm is to compute the values of the feature weights α_j. We are using Generalized Iterative Scaling, a widely used estimation procedure, cf. Darroch and Ratcliff (1972).

3 DOP for Exploiting the Token Chain Level

Data-Oriented Parsing (DOP) is a probabilistic approach to parsing that maintains a large corpus of analyses of previously occurring sentences, cf. Bod et al. (2003). New input is parsed by combining tree-fragments from the corpus; the frequencies of these fragments are used to estimate which analysis is the most probable one.

So far, DOP has basically been applied on syntactic parse trees. In this paper, we show how DOP can be applied to IE. The starting point is the XML–tree of an annotated template instance. Such a *template tree* t is extracted from an annotated document by labeling the root node with the domain–type (see fig. 2) and the immediate child nodes with the slot–tags (called *slot-nodes*). Each slot-node s is the root of a sub–tree (called *slot-tree* and denoted as t_s) whose yield consists of the text fragment α spanned by s. All other nodes of t_s result from the linguistic analysis of α performed by SMES. Note that in contrast to the token level *all* information computed by SMES is used at this level, i.e., in addition to the lexical features, we also make use of the named entities (NE) and phrasal level.

Each template tree t obtained from the training corpus is *generalized* by cutting off certain sub–trees from t's slot–trees, which is basically performed

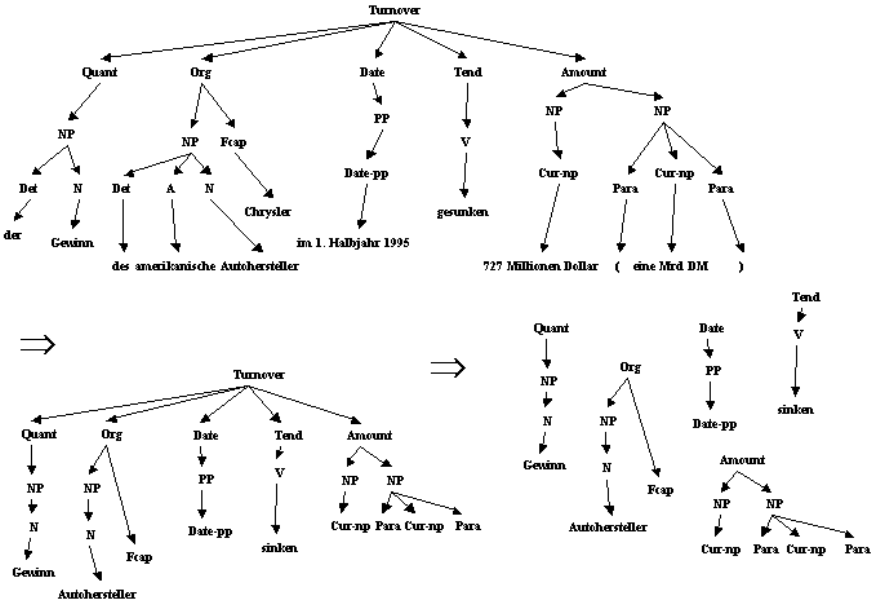

Fig. 2. Example of the tree generalization using DOP.

by deleting the link $n_i \rightarrow n_j$ between a non-terminal node n_i and its child node n_j and by removing the complete subtree rooted at n_j (cf. lower left tree in fig. 2). The resulting tree $t^{'}$ is more general than t, since it has fewer terminal as well as non-terminal nodes than t but otherwise respects the structure of t. All generalized trees are further processed by extracting all slot–trees. Finally, each slot–tree is assigned a probability $p(t_s^{'})$ such that $\sum_{t_i:root(t_i)=s} p(t_i) = 1$).

The tree decomposition operation is linguistically guided by the *head feature principle*, which requires that the head features of a phrasal sign be shared with its head daughter, cf. Neumann (2003). For example, the head daughter of a `NP` is its noun `N`. Using this notation, tree decomposition traverses each slot–tree from the top–downwards by cutting of the non–head daughters with the restriction that if the root label of a non–head daughter d denotes a token class or a named entity, then we retain the root node of d, but cut off d's sub–trees.

4 Hybrid Iterative Tag Insertion

The application phase is realized as a tag–insertion method that is iteratively applied by a central search control on a new document as long as no new slot–

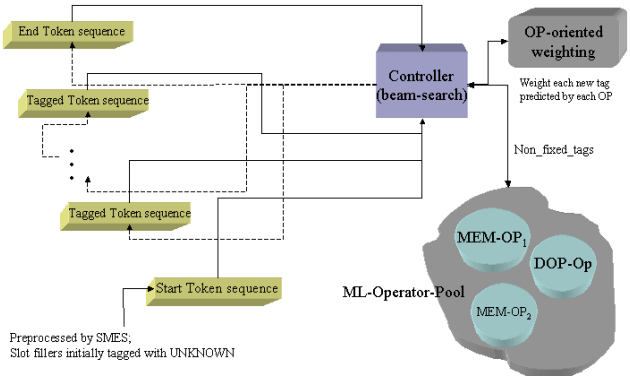

Fig. 3. The Hybrid Iterative Tag Insertion approach.

tag can be inserted (using the slot `unknown` for initializing the tag sequence). The slot–tags are predicted by a set of *operators*. Each operator corresponds to one of the learning algorithms, viz. MEM–OP and DOP–OP, see fig. 3.

The hybrid property of the approach is obtained such that in each iteration all operators are applied independently of each other on the actual tagged sequence. This results in a set of operator–specific new tagged sequences each having an individual weight. The N–best new tagged sequences are passed to the next iteration step, i.e., we perform a *beam–search* with beam size N. The following common weighting scheme is used by each operator op_k

$$w^{(j+1)} = \begin{cases} \dfrac{w^{(j)} \cdot \#p^{(j)} + f_k \cdot \Delta w \cdot \left(\#p^{(j+1)} - \#p^{(j)}\right)}{\#p^{(j+1)}} & , \text{if } \#p^{(j+1)} > \#p^{(j)} \\ w^{(j)} & , \text{if } \#p^{(j+1)} = \#p^{(j)} \end{cases} \quad (3)$$

where $w^{(i)}$ denotes the weight of the tagged sequence determined in iteration step i (setting $w^{(0)} = 0$ enforces $0 \leq w^{(i)} \leq 1$), $\#p^{(i)}$ is the number of fixed tag positions after iteration i (by *fixed* we mean that after the tag `unknown` has been mapped to slot–tag s, s cannot be changed in next iterations). Δw is a feature weight, and f_k a operator–specific performance number (both having values between 0 and 1), which is determined by applying op_k with different parameter settings on a *seen* subset of the training corpus by recording the different values of F–measures obtained.

An operator op_k applies the trained model of a learner on a new linguistically preprocessed token sequence and computes predictions for new slot–tags. Since application can be done in different modes, each operator op_k fixes different parameters. For MEM–OP, we define specific instances of it depending on the search direction (e.g., **leftmost** not yet fixed tag

`unknown`, **rightmost** `unknown` or **best** `unknown`), use of a lexicon, use of previous made predictions, or the maximum number of iterations, cf. also Ratnapharkhi (1998). For DOP–OP different instances could implement different tree matching methods. Currently, we use the following *generate–and–test* tree matching method: from the current token sequence consider all possible sub–sequences (constrained by an automatically computed *breadth–lexicon*, used to restrict the "plausible" length of a potential slot–filler); construct an XML–tree with a root label whose label is the current slot–type in question; apply the same tree generalization method as used in the training phase; finally check for equality of this generalized DOP–tree with corresponding trees from the DOP–model.

5 Experiments

Since there exists no standard IE–corpus for German, we used a corpus of news articles reporting company turnover for the years 1994 and 1995. The corpus has been annotated with the following tags: **Org** (organization name), **Quant** (quantity of the message, which is either `turnover` or `revenue`), **Amount** (amount of the reported event), **Date** (reported time period), **Tend** (increase (+) or decrease (-) of turnover), **Diff** (amount of money announced for that time period). The corpus consists of 75 template instances with 5.878 tokens, from which we used 60 instances for training and 15 for testing.

Evaluation of our hybrid ML–IE approach was done using the standard measures recall (R) and precision (P) and its combined version F–measure.[1] We were mainly been interested in checking whether the combination of MEM and DOP improves the overall performance of our method compared to the performance of our method, when running MEM and DOP in isolation.

Table 4 shows the result of running different instances of the MEM–OP on the test set. Inspecting table 4, we can see that the best result was obtained when MEM was running in **best**–search mode taking into account previous made decisions using no lexicon. Table 5 displays the performance of the DOP–OP applied on different sizes of the training set (using the same test set in all runs). As one can see, precision *decreases* when the training size grows (see next paragraph for a possible explanation). Table 6 shows that the overall performance of the system increases, when MEM and DOP are combined. We can also see that not all instances of the MEM–OP benefit by the combined approach. However, the first table row shows that the F1 value for the MEM–OP increases from 79.3% to **85.2%** when combined with DOP.

The results suggest that MEM performs better than our current DOP tree matcher when running in isolation. The reason is that the tree patterns extracted by means of DOP are more restricted in predicting new tags than MEM. Furthermore, since we currently build tree patterns only for the

[1] $F1=\frac{(\beta^2+1)PR}{\beta^2 P+R}$, where we are using β=1 in our experiments.

L?	P?	leftmost			best			rightmost		
		PRE	**REC**	**FME**	**PRE**	**REC**	**FME**	**PRE**	**REC**	**FME**
•	•	74.9	76.9	75.9	77.4	81.2	79.3	73.2	74.7	73.9
•	○	65.6	80.1	72.2	65.6	80.1	72.2	65.6	80.1	72.2
○	•	79.8	74.2	76.9	82.7	79.6	81.1	80.6	73.7	77.0

Fig. 4. Performance of difference instances of the MEM–OP on the single slot–task. All of them use the model obtained after $i^* = 76$ iterations (which was determined during training as optimal). **L**? indicates whether a lexicon automatically determined from the slot–fillers of the training corpus was used by the MEM–OP. **P**? specifies whether previous made predictions have been taken into account.

op_{DOP}	**PRE**	**REC**	**FME**
C_{15}	071.3	046.8	056.5
C_{30}	064.4	045.7	053.5
C_{45}	059.5	047.3	052.7
C_{60}	055.2	048.9	051.9

Fig. 5. Dependency of the DOP–OP on the size of the training set $C_{|doc|}$.

L?	P?	leftmost			best			rightmost		
		PRE	**REC**	**FME**	**PRE**	**REC**	**FME**	**PRE**	**REC**	**FME**
•	•	75.3	76.9	76.1	85.4	85.0	85.2	77.0	77.4	77.2
•	○	66.4	80.7	72.8	67.4	81.2	73.7	66.7	81.7	73.4
○	•	79.2	73.7	76.3	82.7	79.6	81.1	80.6	73.7	77.0

Fig. 6. The single slot performance values for combined MEM and DOP.

slot–fillers without taking into account context, they are probably too ambiguous. We assume that the degree of ambiguity increases with the number of documents, which might explain, why the performance of DOP decreases. However, when MEM and DOP are combined, it seems that DOP actually can contribute to the overall performance result of F1=**85.2%**. The reason is, that on the one hand side, MEM contributes implicitly contextual information for DOP in that it helps to restrict the search space for tree matching, and on the other hand side, it might be that the more "static" tree patterns might help to filter out some unreliable tag–sequences otherwise predicted by MEM when running in isolation. Our results also suggest, that not all possible combinations of operator instances improve the system performance, and even more, that one cannot expect, that the best operator (when running in isolation) will automatically also be the best choice for a hybrid approach.

6 Related Work

Chieu and Ng (2002) present a MEM approach to IE and compare their system with eight other ML–IE methods for the single slot task. For English seminar announcements data, they report F1=86.9%, which ranks best

(F1=80.9% on average for all systems). Bender et al. (2003) have recently applied MEM for the CoNLL 2003 Named Entity task on English and German data, reporting F1=68.88% for German (83.92% for English). They used a different set of slots (viz. Org, Pers, Loc, Misc), as well as a *cleaned–up corpus* (i.e., linguistically completely disambiguated, which is *not* the case for our method). The best system (88.76% for English, 72,41% for German) also used a hybrid approach by combining MEM, HMM, transformation based learning, and a winnow–based method called RRM, cf. Florian et al. (2003). They also report that MEM belongs to their best standalone performers, and that a combined approach achieved the best overall performance. The major differences wrt. our approach are the use of a cleaned–up corpus, and the use of a non–incremental hybrid approach. A hybrid approach more closely related to our incremental method is described in Freitag (1998), where he combines a dictionary learner, term–space text classification and relational rule reduction.

The experimental results presented here show that a hybrid ML–IE approach combining MEM and DOP can be useful for the problem of IE. So far, we have used our approach for the slot filling task. However, since our approach is in principle open for the integration of more deeper linguistic knowledge, the method should also be applicable for more complex tasks, like learning of n-ary slot relations, or even paragraph–level template filling.

References

BENDER, O., OCH, F., and NEY, H. (2003): Maximum Entropy Models for Named Entity Recognition In: Proceedings of *CoNLL-2003*, pp. 148-151.

BOD, R., SCHA, R. and SIMA'AN, K. (2003): Data-Oriented Parsing. CSLI Publications, University of Chicago Press.

CHIEU, H. L. and NG, H. T. (2002): A Maximum Entropy Approach to Information Extraction from Semi–Structured and Free Text. In Proceedings of *AAAI 2002*.

DARROCH, J. N. and RATCLIFF, D. (1972). Generalized Iterative Scaling for Log-Linear Models. *Annals of Mathematical Statistics, 43, pages 1470–1480.*

FLORIAN, R., ITTYCHERIAH, A., JING, H., and ZHANG, T. (2003): Named Entity Recognition through Classifier Combination. In: Proceedings of *CoNLL-2003*, pp. 168-171.

FREITAG, D. (1998): Multistrategy Learning for Information Extraction. In Proceedings of the *15th ICML*, pages 161–169.

NEUMANN, G. (2003): A Data-Driven Approach to Head-Driven Phrase Structure Grammar. In R. Scha R. Bod and K. Simaan (eds.) *Data-Oriented Parsing*, pages 233-251.

NEUMANN, G. and PISKORSKI, J. (2002): A Shallow Text Processing Core Engine. *Journal of Computational Intelligence, 18, 451–476.*

PIETRA, S. D., PIETRA, V. J. and LAFFERTY, J. D. (1997): Inducing Features of Random Fields. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, 380–393.*

RATNAPARKHI, A. (1998): *Maximum Entropy Models for Natural Language Ambiguity Resolution.* Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA.