Knowledge Information Systems (2005) 00: 1–18
DOI 10.1007/s10115-005-0204-y

RESEARCH ARTICLE

Benjamin Rosendfeld · Ronen Feldman · Moshe Fresko

TEG—a hybrid approach to information extraction

Received: date / Published online: date

Q1

Abstract This paper describes a hybrid statistical and knowledge-based information extraction model, able to extract entities and relations at the sentence level. The model attempts to retain and improve the high accuracy levels of knowledge-based systems while drastically reducing the amount of manual labour by relying on statistics drawn from a training corpus. The implementation of the model, called TEG (trainable extraction grammar), can be adapted to any IE domain by writing a suitable set of rules in a SCFG (stochastic context-free grammar)-based extraction language and training them using an annotated corpus. The system does not contain any purely linguistic components, such as PoS tagger or shallow parser, but allows to using external linguistic components if necessary. We demonstrate the performance of the system on several named entity extraction and relation extraction tasks. The experiments show that our hybrid approach outperforms both purely statistical and purely knowledge-based systems, while requiring orders of magnitude less manual rule writing and smaller amounts of training data. We also demonstrate the robustness of our system under conditions of poor training-data quality.

Keywords HMM · Information extraction · Rules-based systems · Text mining · Information extraction · HMM

1 Introduction

The knowledge engineering (mostly rule-based) systems traditionally were the top performers in most IE benchmarks, such as MUC [5], ACE and the KDD CUP [23]. Recently, though, the machine learning systems became state of the art, especially for simpler tagging problems, such as named entity recognition [2] or field extraction [20].

B. Rosendfeld · R. Feldman (✉) · M. Fresko
Computer Science Department, Bar-Ilan University, Ramat Gan 52900, Israel
E-mail: feldman@cs.biu.ac.il

Still, the knowledge-engineering approach retains some of its advantages. It is focused around manually writing patterns to extract the entities and relations. The patterns are naturally accessible to human understanding and can be improved in a controllable way. Whereas improving the results of a pure machine-learning system would require providing it with additional training data. However, the impact of adding more data soon becomes infinitesimal while the cost of manually annotating the data grows linearly.

We present a hybrid entities- and relations-extraction system, which combines the power of knowledge-based and statistical machine-learning approaches. The system is based on stochastic context-free grammars. It is called TEG, for trainable extraction grammar. The rules for the extraction grammar are written manually, while the probabilities are trained from an annotated corpus. The powerful disambiguation ability of PCFGs allows the knowledge engineer to write very simple and naive rules while retaining their power, thus greatly reducing the required labour. In addition, the size of the needed training data is considerably smaller than the size of the training data needed for pure machine-learning systems (for achieving comparable accuracy results). Furthermore, the tasks of rule writing and corpus annotation can be balanced against each other.

In the sections that follow, we briefly survey the related work, describe our system and then proceed to the experiments and comparison with other information-extraction systems.

2 Related work

The knowledge-based IE systems are well known. We use the DIAL system developed by the authors [9] as a typical example, showing both the pros and cons of the knowledge-based approach. DIAL is based on a general-purpose rule language. The system was top performing at ACE-2 after being manually prepared for the task during 2 months by a team of four people.

There have been several attempts to automate the task of developing information extraction modules using machine-learning methods. The general idea is that a domain expert labels the target concepts in a set of documents. The system then learns a model of the extraction task, which can be applied to new documents automatically. Approaches can be roughly divided by the type of model they use. First, there are systems generating sets of rules, very much in the flavour of hand-written rules for representing the target concepts. These approaches include methods based on finite-automata [16], grammar learning[12] and ILP [1, 13, 14].

Second, there are approaches using diverse forms of probabilistic representations. Most prominently, hidden Markov models (HMM) have been used for the task of IE (e.g. [3, 10, 11, 17]). HMM are probabilistic automata for which the states themselves are hidden. Once trained, such a model can give an estimate on how probable a text fragment contains the target concept. Besides HMM, there are also other approaches based on naive Bayes [8], SVM [22] or approaches combining several of these techniques [8, 13, 15].

Recently, conditional approaches based on maximal entropy were reported to outperform generative HMM models on several tasks. The models are maximum entropy Markov models (MEMM) [20] and conditional random fields (CRF) [18].

Stochastic context-free grammars were also used for information extraction in several systems. Typically, a SCFG is used for syntactic parsing of sentences, as in the BBN SIFT system [19].

Our approach is different in that our system does not employ full syntactic parsing. Instead, semantically oriented SCFG are constructed manually. The closest to our approach is the system described in [6], which extracted management succession events using a semantically oriented SCFG of a special predefined form. Our system can be viewed as a generalisation and extension of this work. Our system allows grammars of arbitrary structure, extends the possibilities for leaf grammar nodes and adds the possibility of conditioning the probabilities of rules upon context. Also, in contrast with [6], our system works on real-world documents, which contain many irrelevant sentences.

3 TEG—bridging the gap between statistical and rule-based IE systems

Although the formalisms based on probabilistic finite-state automata are quite successful for entity extraction, they have shortcomings, which make them harder to use for the more difficult task of extracting relationships.

One problem is that a finite-state automaton model is flat, so its natural task is assignment of a tag (state label) to each token in a sequence. This is suitable for the tasks where the tagged sequences do not nest and where there are no explicit relations between the sequences. Part-of-speech tagging and entity extraction tasks belong to this category, and indeed the HMM-based PoS taggers and entity extractors are state of the art.

Extracting relationships is different in that the tagged sequences can and must nest, and there are relations between them, which must be explicitly recognised. While it is possible to use nested automata to cope with this problem, we felt that using a more general context-free grammar formalism would allow for a greater generality and extendibility without incurring any significant performance loss.

3.1 SCFG formalism

Classical definition: A stochastic context-free grammar (SCFG) is a quintuple $G = (T, N, S, R, P)$, where T is the alphabet of terminal symbols (tokens), N is the set of nonterminals, S is the starting nonterminal, R is the set of rules, and P: R [0..1] defines their probabilities. The rules have the form $n \to s_1 s_2 \ldots s_k$, where n is a nonterminal and each s_i either token or another nonterminal. As can be seen, SCFG is a usual context-free grammar with the addition of the P function.

Similar to a canonical (nonstochastic) grammar, SCFG is said to *generate* (or *accept* a given string (sequence of tokens) if the string can be produced starting from a sequence containing just the starting symbol, S, and one by one expanding nonterminals in the sequence using the rules from the grammar. The particular way a string was generated can be naturally represented by a *parse tree*, with the starting symbol as a root, nonterminals as internal nodes and the tokens as leaves.

The semantics of the probability function P is straightforward. If r is the rule $n \to s_1 s_2 \ldots s_k$, then $P(r)$ is the frequency of expanding n using this rule. Or, in Bayesian terms, if it is known that a given sequence of tokens was generated by

expanding n, then $P(r)$ is the a priori likelihood that n was expanded using the rule r. Thus, it follows that, for every nonterminal n, the sum $P(r)$ of probabilities of all rules r headed by n must equal one.

How SCFG is used: Usually, some of the nonterminal symbols of a grammar correspond to meaningful language concepts, and the rules define the allowed syntactic relations between these concepts. For instance, in a parsing problem, the nonterminals may include S, NP, VP, etc. and the rules would define the syntax of the language. For example, S NP VP. Then, when the grammar is built, it is used for parsing new sentences. In general, grammars are ambiguous in the sense that a given string can be generated in many different ways. With nonstochastic , there is no way to compare different parse trees, so the only information we can gather for a given sentence is whether or not it is *grammatical*, that is, whether it can be produced by any parse. With SCFG, different parses have different probabilities; thus, it is possible to find the best one, resolving the ambiguity.

In designing our system, it was decided that it is neither necessary nor desirable (for performance reasons) to perform a full syntactic parsing of all sentences in the document. Instead, a very basic parsing is employed for the bulk of a text, but within the relevant parts, the grammar is much more detailed. Thus, the extraction grammars can be said to define *sublanguages* for very specific domains. Examples of such grammars will be presented in the next section.

In the classical definition of SCFG, it is assumed that the rules are all independent. In this case, it is possible to find the (unconditional) probability of a given parse tree by simply multiplying the probabilities of all rules participating in it. Then the usual parsing problem is formulated as follows: given a sequence of tokens (a*string*, find the most probable parse tree that could generate the string. A simple generalisation of the Viterbi algorithm is able to efficiently solve this problem.

In practical applications of SCFGs, it is rarely the case that the rules are truly independent. Then, the easiest way to cope with this problem while leaving most of the formalism intact is to let the probabilities $P(r$ be conditioned upon the context where the rule is applied. If the conditioning context is chosen reasonably, the Viterbi algorithm still works correctly even for this more general problem.

3.2 TEG—using SCFG to perform IE

In a first attempt, we created a relationship extractor based on *markovian* SCFG. Markovian means that every possible rule that can be formed from the available symbols has nonzero probability. Usually, all probabilities are initially set to be equal and then adjusted according to the distributions found in the training data. This strategy is a straightforward generalisation of an HMM-based entity extractor. It has the benefit that all rules are created automatically, and the system only needs the tagged training corpus in order to learn the target domain. But there is also an obvious downside. If the amount of training data is insufficient, such a system performs poorly.

For some problems, the available training corpora appear to be adequate. In particular, Markovian SCFG parsers trained on the Penn Treebank perform quite well. HMM entity extractors, which are a particularly simple case of Markovian SCFGs, also perform well [4, 7, 21]. But for the task of relationship extraction, it

turns out to be impractical to manually tag the amount of documents that would be sufficient to adequately train such grammar. At a certain point, it becomes more productive to go back to the original hand-crafted system and write rules for it, even though it is a much more skilled labour!

Therefore, we adopted a hybrid strategy, which we coined TEG (trainable extraction grammars), which attempts to strike a balance between the two knowledge engineer chores—writing the extraction rules and manually tagging the documents. In TEG, the knowledge engineer writes SCFG rules, which are then trained on the data that is available. The powerful disambiguating ability of the SCFG makes writing rules a much simpler and cleaner task. Furthermore, the knowledge engineer has the control of the generality of the rules (s)he writes and consequently on the amount and the quality of the manually tagged training corpus the system would require.

3.3 Syntax of a TEG rulebook

A TEG rulebook consists of declarations and rules. Rules basically follow the classical grammar rule syntax, with a special construction for assigning concept attributes. Notation shortcuts like [] and | can be used for easier writing. The nonterminals referred by the rules must be declared before use. Some of them can be declared as *output concepts*, which are the entities, events and facts that the system is designed to extract. Additionally, two classes of terminal symbols also require declaration: *termlists* and *ngrams*.

A termlist is a collection of terms from a single semantic category, either written explicitly or loaded from external sources. Examples of termlists are countries, cities, states, genes, proteins, people's first names and job titles. Some linguistic concepts, such as lists of prepositions, can also be defined as termlists. Theoretically, a termlist is equivalent to a nonterminal symbol that has a rule for every term.

An ngram is a more complex construction. When used in a rule, it can expand to any single token. But the probability of generating a given token is not fixed in the rules, but learned from the training dataset, and may be conditioned on one or more previous tokens. Thus, ngrams is one of the ways the probabilities of TEG rules can be context dependent. The exact semantics of ngrams is explained in the next section.

Let us see a simple meaningful example of a TEG grammar:

output concept Acquisition(Acquirer, Acquired);
ngram AdjunctWord;
nonterminal Adjunct;
Adjunct :- AdjunctWord Adjunct | AdjunctWord;
termlist AcquireTerm = acquired bought (has acquired) (has bought);
Acquisition :- Company → Acquirer [","Adjunct ","] AcquireTerm
Company→ Acquired;

The first line defines a target relation **Acquisition**, which has two attributes, **Acquirer** and **Acquired**. Then an ngram **AdjunctWord** is defined, followed by

a nonterminal **Adjunct**, which has two rules, separated by |, together defining **Adjunct** as a sequence of one or more **AdjunctWord**s. Then a termlist **AcquireTerm** is defined, expanding as the main acquisition verb phrase. Finally, the single rule for the **Acquisition** concept is defined—as **Company,** followed by optional **Adjunct**, delimited by commas, followed by **AcquireTerm** and a second **Company**. The first **Company** is the **Acquirer** attribute of the output frame and the second is the **Acquired** attribute.

The final rule requires the existence of a defined **Company** concept. The following set of definitions defines the concept in a manner emulating the behaviour of a HMM entity extractor:

```
output concept Company;
ngram CompanyFirstWord;
ngram CompanyWord;
ngram CompanyLastWord;
nonterminal CompanyNext;
Company :- CompanyFirstWord CompanyNext | CompanyFirstWord;
CompanyNext :- CompanyWord CompanyNext | CompanyLastWord;
```

Finally, in order to produce a complete grammar, we need a starting symbol and the special nonterminal that would match the strings that do not belong to any of the output concepts:

```
start Text;
nonterminal None;
ngram NoneWord;
None :- NoneWord None | ;
Text :- None Text | Company Text | Acquisition Text | ;
```

Those 20 lines of code are able to accurately find a fair number of acquisitions after a very modest training. Note that the grammar is extremely ambiguous. An ngram can match any token, so **Company**, **None** and **Adjunct** are able to match any string. Yet, using the learned probabilities, TEG is usually able to find the correct interpretation.

3.4 TEG training

Currently, there are three different classes of trainable parameters in a TEG rulebook: the probabilities of rules of nonterminals, the probabilities of different expansions of ngrams and the probabilities of terms in wordclasses. All those probabilities are smoothed maximum likelihood estimates, calculated directly from the frequencies of the corresponding elements in the training dataset.

For example, suppose we have the following simple TEG grammar, which finds simple person names:

```
nonterm start Text;
concept Person;
ngram NGFirstName;
ngram NGLastName;
ngram NGNone;
```

termlist TLHonorific = Mr Mrs Miss Ms Dr;
(1) Person :- TLHonorific NGLastName;
(2) Person :- NGFirstName NGLastName;
(3) Text :- NGNone Text;
(4) Text :- Person Text;
(5) Text :- ;

By default, the initial untrained frequencies of all elements are assumed to be 1. They can be changed using $\langle count \rangle$ syntax, an example of which is shown below. The numbers in parentheses at the left side are not part of the rules and are used only for reference. Let us train this rulebook on the training set containing one sentence:

Yesterday, $\langle Person \rangle$*Dr Simmons*$\langle /Person \rangle$, *the distinguished scientist presented the discovery*.

This is done in two steps. First, the sentence is parsed using the untrained rulebook, but with the constraints specified by the annotations. In our case, the constraints are satisfied by two different parses, expanding **Person** by rules (1) and (2), respectively. The ambiguity arises because both **TLHonorific** and **NGFirstName** can generate the token Dr. In this case, the ambiguity is resolved in favour of the **TLHonorific** interpretation because, in the untrained rulebook, we have

- $P(Dr \mid \textbf{TLHonorific}) = 1/5$ (choice of one term among five equiprobable ones),
- $P(Dr \mid \textbf{NGFirstName}) \approx 1/N$, where N is the number of all known words (untrained ngram behaviour).

After the training, the frequencies of the different elements are updated, which produces the following trained rulebook (only lines that were changed are shown). Note the $\langle count \rangle$ syntax:

termlist TLHonorific = Mr Mrs Miss Ms ⟨2⟩**Dr;**
Person :- ⟨2⟩**TLHonorific NGLastName;**
Text :- ⟨11⟩**NGNone Text;**
Text :- ⟨2⟩**Person Text;**
Text :- ⟨2⟩**;**

Additionally, the training will generate a separate file containing the statistics for the ngrams. It is similar in nature, but more complex, because the bigram frequencies, token feature frequencies and unknown word frequencies are taken into consideration. In order to understand the details of ngrams training, it is necessary to go over the details of their internal working.

An ngram always generates a single token. Any ngram can generate any token, but naturally the probability of generating depends on the ngram, on the token and on the immediate preceding context of the token. This probability is calculated at the runtime using the following statistics:

- $Freq(*)$ = total number of times the ngram was encountered in the training set.

- $Freq(W)$, $Freq(F)$, $Freq(T)$ = number of times the ngram was matched to the word W, the feature F and the token T, respectively. Note that a token T is a pair consisting of a word $W(T)$ and its feature $F(T)$.
- $Freq(T \mid T_2)$ = number of times token T was matched to the ngram in the training set and the preceding token was T_2.
- $Freq(* \mid T_2)$ = number of times the ngram was encountered after the token T_2.

So, assuming all those statistics are gathered, the probability of the ngram generating a token T given that the preceding token is T_2 is estimated as

$$\mathrm{P(T \mid T2)} = \frac{1}{2} \cdot \frac{\mathrm{Freq(T \mid T2)}}{\mathrm{Freq}(* \mid \mathrm{T2})} + \frac{1}{4} \cdot \frac{\mathrm{Freq(T)}}{\mathrm{Freq}(*)} + \frac{1}{4} \cdot \frac{\mathrm{Freq(W)} \cdot \mathrm{Freq(F)}}{\mathrm{Freq}(*)^2}$$

This formula linearly interpolates between the three models—the bigram model, the back-off unigram model and the further backoff word+feature unigram model. The interpolation factor was chosen to be $1/2$, which is a natural choice. However, the experiments have shown that varying the lambdas in reasonable ranges does not significantly influence the performance.

Finally, the matters are made a bit more complicated by the *unknown* tokens. The fact that a token was never encountered during the training gives by itself an important clue to the token's nature. In order to be able to use this clue, the separate unknown model is trained. The training set for it is created by dividing the available training data into two halves, and treating all tokens in one half, that are not present in the other half as special unknown tokens. The model trained in this way is used whenever an unknown token is encountered during runtime.

3.5 Additional features

There are several additional features that improve the system and help to customise it for other domains. First, the probabilities of different rules of a nonterminal need not be fixed, but may depend on their context. Currently, the rules for a specific nonterminal can be conditioned on the previous token in a way similar to the way ngram probabilities depend on previous token. Other conditioning is, of course, possible, even to the extent of using maximal entropy for combining several conditioning events.

Second, an external tokeniser and/or token feature generator can be substituted for the regular one. It is even possible to use several feature generators simultaneously—different ngrams may use different token feature sets. This is useful for languages other than English, as well as for special domains. For instance, in order to extract the names of chemical compounds or complex gene names, it may be necessary to provide a feature set based on morphological features. In addition, an external part-of-speech tagger or shallow parser may be used as a feature generator.

For real-life IE tasks, it is often necessary to extract very rare target concepts. This is especially true for relations. While there could be thousands of persons or organisations in a dataset, the number of acquisitions could well be less than

50. The ngrams participating in the rules for such concepts are surely to be undertrained. In order to alleviate this problem, the *shrinkage* technique can be used. An infrequent specific ngram can be set to *shrink* to another, more common and more general ngram. Then the probability of generating a token by the ngram is interpolated with the corresponding probability for the more common parent ngram. A similar technique was used with great success for HMM [3], and we found it very useful for TEG as well.

3.6 Example of real rules

Here we shall demonstrate a fragment of the TEG rules, written for the extraction of the PersonAffiliation relation from a real industry corpus. The fragment will show a usage of the advanced features of the system, as well as give another glimpse of the flavour of rule writing in TEG.

The PersonAffiliation relation contains three attributes—name of the person, name of the organisation and position of the person in the organisation. It is declared as follows:

concept output PersonAffiliation(Name, Position, Org);

Most often, this relation is encountered in the text in the form Mr.Name,PositionofOrg or OrgPositionMs.Name. Almost any order of the components is possible, with commas and prepositions inserted as necessary. Also, it is common for Name, Position, or both to be conjunctions of pairs of corresponding entities: Mr.Name1 and Ms.Name2, the Position1 and Position2 of Org, or Org's Position1 and Position2, Ms.Name. In order to catch those complexities, and for general simplification of the rules, we use several auxiliary nonterminals: **Names**, which catches one or two Names; **Positions**, which catches one or two Positions and **Orgs**, which catches Organisations and Locations, which can also be involved in PersonAffiliation, as in Bush, president of US:

nonterms Names, Positions, Orgs;
Names :- PERSON → Name | PERSON → Name1 "and"
PERSON → Name2;
Positions :- POSITION → Position | POSITION → Position1 "and"
POSITION → Position2;
Orgs :- ORGANIZATION→Org | LOCATION→Org;

We also use auxiliary nonterminals for catching pairs of attributes: **PosName** and **PosOrg**:

nonterms PosName, PosOrg;
PosName :- Positions Names | PosName and PosName;
wordclass wcPreposition = at in of for with;
wordclass wcPossessive = ('s) ' ;
PosOrg :- Positions wcPreposition Orgs;
PosOrg :- Orgs [wcPossessive] Positions;

Finally, the PersonAffiliation rules:

PersonAffiliation :- Orgs [wcPossessive] PosName;
PersonAffiliation :- PosName wcPreposition Orgs;

PersonAffiliation :- PosOrg [,] Names;
PersonAffiliation :- Names , PosOrg;
PersonAffiliation :- Names is a PosOrg;

The rules above catch about 50% of all PersonAffiliation instances in the texts. Other instances do not conform to the patterns above in several respects. So, in order to improve the accuracy, additional rules need to be written. First, the organisation name is often entered into a sentence as a part of a descriptive noun phrase, as in: Ms.Name is a Position of the industry leader Org. In order to catch this in a general way, we define an **OrgNP** nonterm, which uses an external PoS tagger:

ngram ngOrgNoun featureset ExtPoS restriction Noun;
ngram ngOrgAdj featureset ExtPoS restriction Adj;
ngram ngNum featureset ExtPoS restriction Number;
ngram ngProper featureset ExtPoS restriction ProperName;
ngram ngDet featureset ExtPoS restriction Det;
ngram ngPrep featureset ExtPoS restriction Prep;
nonterm OrgNounList;

OrgNounList :- ngOrgNoun [OrgNounList];
nonterms OrgAdjWord, OrgAdjList;
OrgAdjWord :- ngOrgAdj | ngNum | ngProper;
OrgAdjList :- OrgAdjWord [OrgAdjList];
nonterm OrgNP;
OrgNP :- [ngDet] [OrgAdjList] OrgNounList;
OrgNP :- OrgNP ngPrep OrgNP;
OrgNP :- OrgNP and OrgNP;

The external PoS tagger provides an alternative token feature set, which can be used by ngrams via the ngram *featureset* declaration. The *restriction* clause in the ngram declaration specifies that the tokens matched by the ngram must belong to the specified feature. Altogether, the set of rules above defines an **OrgNP** nonterm in a way similar to the way a syntax-parsing grammar would define a noun phrase. In order to use the nonterm in the rules, we simply modify the **Orgs** nonterminal:

Orgs :- [OrgNP] ORGANIZATION → Org | LOCATION → Org;

Note that, although OrgNP is defined very generally (it is able to match any noun phrase whatsoever), the way it is used is very restricted. During training, the ngrams of OrgNP learn the distributions of words for this particular use and, during the run, the probability of OrgNP generating a true organisation-related noun phrase is much greater than for any other noun phrase in the text.

Finally, we will demonstrate the usage of ngram shrinkage. There are PersonAffiliation instances, in which some irrelevant sentence fragments separate the attributes. For example, ORG . . . . . said the company's Position Mr.Name. In order to catch the ". . . . . " part, we can use the **None** nonterm, which generates all irrelevant fragments in the text. Alternatively, we can create a separate ngram and a nonterminal for the specific use of catching irrelevant fragments inside PersonAffiliation. Both those solutions have their disadvantages. The None nonterminal is too general and doesn't catch the specifics of the particular case. A

specific nonterminal, on the other hand, is very much undertrained. The solution is to use a specific nonterminal but shrink its ngram to None:

nonterm BlaBla;
ngram ngBlaBla, ngNone;
BlaBla :- ngBlaBla [BlaBla];
PersonAffiliation :- Orgs BlaBla PosName;

The rules described above catch 70% of all PersonAffiliation instances, which is already a good result for relationship extraction from a real corpus. And the process of writing rules can be continued to further improve the accuracy.

4 Experimental evaluation

For evaluation of our system, we used three different corpora, available to us: MUC-7, ACE-2 and an industry corpus, which we will call INC.

4.1 MUC-7 evaluation—comparison with HMM-based NER

The MUC-7-named entity recognition corpus consists of a set of news articles related to aircraft accidents, altogether containing about 200 K words, with the named entities manually categorised into three basic categories: PERSON, ORGANIZATION and LOCATION. Some other entities are also tagged: dates and times, monetary units, etc., but they did not take part in our evaluation.

The corpus does not contain tagged relationships, so it was used to evaluate the difference in the performance between the four entity extractors: the regular HMM, its emulation using TEG, a set of hand-craft rules written in DIAL and the Full TEG system, which consists of the HMM emulation augmented by a small set of hand-crafted rules (about 50 lines of code added).

The results of our experiments are summarised in Table 1.

The small accuracy difference between the regular HMM and its emulation is due to slight differences in probability conditioning methods. It is evident that the hand-crafted rules performed better than the HMM-based extractors but were inferior to the TEG extractor. Significantly, the hand-crafted rules achieved the best precision; however, their recall was far worse.

The HMM named-entity recognition results published in [3] are somewhat higher than we were able to produce using our version of a HMM entity extractor.

Table 1 Accuracy results for MUC-7.

	HMM entity extractor			Emulation using TEG			DIAL Rules			Full TEG system		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
Person	86.91	85.13	86.01	86.31	86.83	86.57	81.32	93.75	87.53	93.75	90.78	92.24
Org	87.94	89.75	88.84	85.94	89.53	87.7	82.74	93.36	88.05	89.49	90.9	90.19
Location	86.12	87.2	86.66	83.93	90.12	86.91	91.46	89.53	90.49	87.05	94.42	90.58

We hypothesise that the reason for the difference is the usage of additional training data in the Nymble experiments. The paper mentions using approximately 750 K words of training data, while we had only 200 K. Regardless of the reasons for the difference, the experiment clearly shows that the addition of a small number of handcrafted rules can further improve the results of a purely automatic HMM-based named entity extraction.

4.2 ACE-2 evaluation—extracting relationships

The ACE-2 was a follow-up to ACE-1 and included, in addition to tagged entities, also tagged relationships. The ACE-2 annotations are more complex than those supported by the current version of our system. Most significantly, the annotations resolve all anaphoric references, which is outside the scope of the current implementation. Therefore, it was necessary to remove annotations that contained anaphoric references. This was done automatically, using a simple Perl script.

For the purpose of evaluating relationship extraction, we choose the ROLE relation. The original ACE-2 annotations make finer distinctions between the different kinds of ROLE, but for the purposes of current evaluation, we felt it sufficient to just recognise the basic relationships and find their attributes.

The results of this evaluation are shown in Table 2. For Comparison, we also show the performance of the HMM entity extractor on the entities in the same dataset.

As expected, the accuracy of a purely Markovian SCFG without additional rules is rather mediocre. However, by adding a small number of hand-crafted rules (altogether about 100 lines of code), it was raised considerably (by 15% in F1). The performances of the three systems on the named entities differ very little because it is essentially the same system. The slight improvement of the Full TEG system is due to better handling of the entities that take part in ROLEs.

In Fig. 1, we can see how the accuracy of the TEG system changes as a function of the amount of available training data. There are three graphs in the figure, a graph that represents the accuracy of the grammar with no specific ROLE rules, a graph that represents the accuracy of the grammar with four ROLE rules and, finally, a graph that represents the accuracy of the grammar with seven ROLE rules.

Analysis of the graphs reveals that, in order to achieve 70% accuracy the system needs 125 K of training data when using all of the specific ROLE rules,

Table 2 Accuracy results for ACE-2.

	HMM entity extractor			Markovian SCFG			Full TEG system (with 7 ROLE rules)		
	Recall	Prec	F	Recall	Prec	F	Recall	Prec	F
Role				67.55	69.86	68.69	83.44	77.3	80.25
Person	85.54	83.22	84.37	89.19	80.19	84.45	89.82	81.68	85.56
Organization	52.62	64.735	58.05	53.57	67.46	59.71	59.49	71.06	64.76
GPE	85.54	83.22	84.37	86.74	84.96	85.84	88.83	84.94	86.84

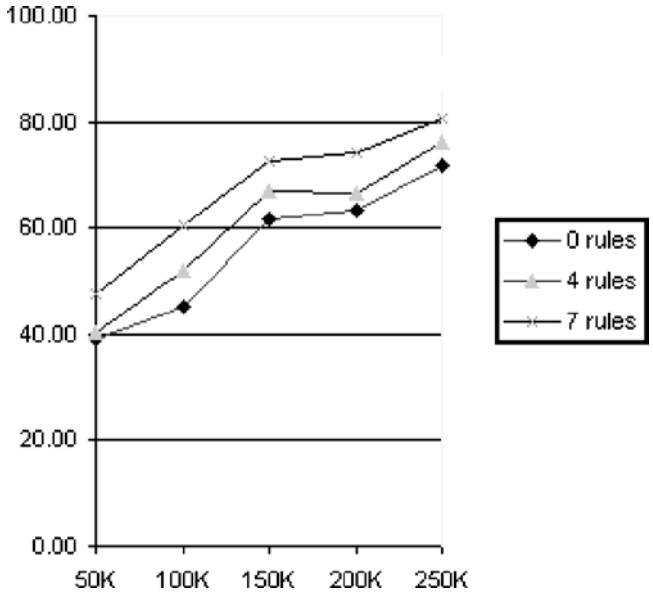

Fig. 1 Accuracy (F1) of the TEG system (with different grammars) as a function of the size of the training corpus (ACE-2).

while 250 K of training data are needed when no specific rules are present. Thus, adding a small set of simple rules may save 50% of the training-data requirements.

4.3 Internal news corpus evaluation—extracting real-world relationships

In addition to the classic data sets, we used one large new corpus that was annotated internally by students (the INC Corpus). Clearly, the quality of the annotation is not as good as the professional annotators that annotated the MUC and ACE corpuses. Based on our experiments, we estimate that about 5–10% of the tags in the training corpus are erroneous. The INC corpus contains about 1 million tokens of news articles, annotated with the common Person, Organisation, Location, Currency entities and three common relations—PersonAffiliation, OrgLocation and Acquisition. OrgLocation has two attributes—an organisation name and its location. Acquisition also has two attributes—the buyer company and the acquired company. PersonAffiliation was described in one of the previous sections.

In Table 3, we can see the accuracy results of the relationship extraction from the INC corpus. In the exact-match results, an instance of a relation is considered

Table 3 Accuracy results for relation extraction from the INC Corpus.

	Partial match results			Exact match results		
	Recall	Prec	F	Recall	Prec	F
PersonAffiliation	89.61	94.52	**92.00**	75.33	79.46	**77.33**
OrgLocation	85.32	77.78	**80.00**	76.47	72.22	**74.29**
Acquisition	76.00	86.36	**80.85**	68.00	77.27	**72.34**

Table 4 Accuracy results for entity extraction from the INC Corpus.

	INC development test			INC final test		
	Recall	Prec	F	Recall	Prec	F
Person	93.816	83.397	**88.30**	93.437	89.127	**91.23**
Organization	80.856	78.248	**79.53**	83.481	78.859	**81.10**
Location	91.775	81.748	**86.47**	94.892	82.477	**88.25**
Currency	97.667	96.7	**97.18**	98.058	94.393	**96.19**

to be correctly extracted if every attribute of the relation is correctly assigned and all boundaries are correctly set. In the partial-match results, an instance is considered to be correctly extracted if at least one attribute is correct. The accuracy results of the entity extraction from the INC corpus are summarised in Table 4.

4.4 INC corpus evaluation—effects of poor annotation quality

The INC corpus is annotated by nonexperts, and the quality of annotations is significantly worse than the quality of MUC and ACE corpora. We estimate the annotations to be about 90–95% accurate. The following experiment shows that TEG can work robustly under poor training conditions, a very important trait it shares with most other probabilistic-based systems.

It should be noted that, even in the case of poor training corpus quality, the test corpus should be clean because, otherwise, the results of the evaluation are misleading. Therefore, we randomly selected a small set of documents (about 30 K words) from the corpus, manually double checked and cleaned the annotations and used this clean corpus for the final test. The training corpus and the development test corpus were left noisy. This is why the final test results are actually better than the development test results, as we show in Table 4.

As can be seen, the final results are 3–6% better than the development test results. This shows the ability of TEG to effectively ignore bad or inconsistent examples. And it shows the possibility of using TEG to improve the annotation quality.

We also tested a HMM entity extractor on the INC dataset. As expected, it also showed a robust behaviour and consistently achieved about 6–7% F-measure less than TEG in all categories. By disabling different parts of the TEG ruleset, we analysed exactly which of TEG abilities produces the improvement over the baseline HMM. It turned out that there exist three distinctly different parts, contributing about equally to the overall improvement. First, there are *gazetteers*—the lists of known entities—which TEG is able to use and HMM is not. Second, there are *structural rules*, which define the internal structure of concepts with greater precision. And finally, there are *context rules*, which place entities into contexts made of words and other entities. While the gazetteers and structural rules are possible to implement using FSA-based models, such as MEMM or CRF, the context rules by their nature require stronger models.

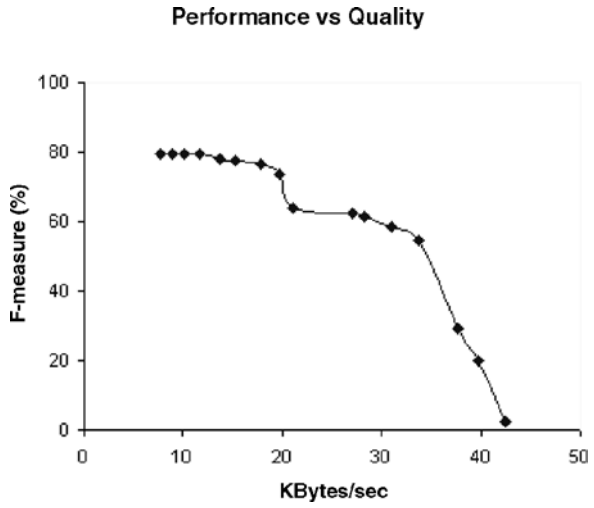

Fig. 2 Trading accuracy for performance.

5 Implementation issues

The current version of TEG is written in C++ and implements the agenda-based probabilistic chart parser. The worst-case performance of the parser is$O(CSN^3)$, where C is the number of nonterminals, S is the number of grammar states and N is the sequence length. For reasonable grammars, the average case performance is actually $O(N^2)$, which is much better than the worst case but still not as good as the linear $O(S^2\ N$ performance of HMM models. However, we found that, by implementing a simple approximation, it is possible to greatly increase the performance without reducing the extraction accuracy. The idea of the method is to exclude a grammar edge from further consideration if its inner probability is less than a small fraction of the best probability currently achieved for the sequence spanned by the edge. By lowering the fraction constant it is possible to trade accuracy for performance. The graph of their interdependence for the ACE-2 experiment is shown in Fig. 2. Note that the left-most values where the best accuracy is achieved are still about 15 times faster than the original nonoptimised version. The optimised version achieves performance of 600 Kb/min on a 2-GHz processor.

6 Future work

The next stages of the TEG development will continue in three principal directions. First, we will improve the abilities of the system by adding desirable features such as the ability to handle overlapping concepts and more refined probability conditioning. Second, we plan to develop an integrated rule writing and tagging GUI environment, which will allow the two tasks to be done in parallel, supporting and bootstrapping each other. There is also a significant practical problem of checking the consistency of tagging and of helping the annotators find and fix the tagging errors. Finally, in order to be more useful, the system itself should be

integrated into a larger information extraction architecture, which might include a knowledge base of known facts, a postprocessor resolving coreferences and an ability to combine the extracted relations into complete scenario templates.

7 Summary

We have presented the TEG system, which is a novel information-extraction system based on the SCFG formalism. The TEG system takes a middle ground between the knowledge-engineering approach and the machine learning-based approach. Because the rules used in TEG are far simpler than the typical information extraction rules, it overcomes the knowledge acquisition bottleneck, which is the main hurdle of the knowledge-engineering approach. In addition, compared with the pure machine-learning approach, it enables getting better accuracies using a smaller number of annotated documents. This reduction in the needed number of annotated documents is crucial in real-world scenarios because it enables a much more rapid deployment of the extraction technology.

References

1. Aitken JS (2002) Learning information extraction rules: an inductive logic programming approach. In: Proceedings of 15th European Conference on Artificial Intelligence. IOS Press, Amsterdam
2. Bikel DM, Miller S, Schwartz R, Weischedel R (1997) Nymble: a high-performance learning name-finder. In: Proceedings of ANLP-97, pp 194–201
3. Bikel DM, Schwartz RL, Weischedel RM (1999) An algorithm that learns what's in a name. Mach Learn **34**(1–3), 211–231
4. Charniak E (2000) A maximum-entropy-inspired parser. In: Proceedings of the Meeting of the North American Association for Computational Linguistics
5. Chinchor N, Hirschman L, Lewis D (1994) Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). Comput Linguistics **3**(19), 409–449
6. Collins M, Miller S (1998) Semantic tagging using a probabilistic context free grammar. In: Proceedings of the 6th Workshop on Very Large Corpora, pp 38–48
7. Collins M (1997) Three generative, lexicalized models for statistical parsing. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, pp 16–23
8. De Sitter A, Daelemans W (2003) Information extraction via double classification. In: Proceedings of International Workshop on Adaptive Text Extraction and Mining, pp 66–73
9. Feldman R (2002) Text mining. In: Kloesgen W, Zytkow J (eds) Handbook of Data Mining and Knowledge Discovery. MIT Press, Cambridge, MA
10. Freitag D, McCallum AK (1999) Information extraction with hmms and shrinkage. In: Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction
11. Freitag D, McCallum A (2000) Information extraction with hmm structures learned by stochastic optimization. In: Proceedings of AAAI/IAAI, pp 584–589
12. Freitag D (1997) Using grammatical inference to improve precision in information extraction. In: Proceedings of Workshop on Grammatical Inference, Automata Induction, and Language Acquisition (ICML'97). Nashville, TN
13. Freitag D (1998) Information extraction from html: application of a general machine learning approach. In: Proceedings of AAAI/IAAI, pp 517–523
14. Grieser G, Jantke KP, Lange S, Thomas B (2000) A unifying approach to html wrapper representation and learning. In: Discovery Science, Third International Conference, DS 2000. Kyoto, Japan. Proceedings, vol 1967, pp 50–64. Springer, Berlin Heidelberg New York

Q2

15. Kushmerick N, Johnston E, McGuinness S (2001) Information extraction by text classification. In: Proceedings of IJCAI-01 Workshop on Adaptive text Extraction and Mining. Seattle, WA
16. Kushmerick N (2002) Finite-state approaches to web information extraction. In: Proceedings of 3rd Summer Convention on Information Extraction. Rome
17. Leek TR (1997) Information Extraction Using Hidden Markov Models.
18. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of 18th International Conference on Machine Learning, pp 282–289. Morgan Kaufmann, San Francisco, CA
19. Miller S, Crystal M, Fox H, Ramshaw L, Schwartz R, Stone R, Weischedel R (1998) The annotation group. Algorithms that learn to extract information–BBN: description of the SIFT system as used for MUC. In: Proceedings of the 7th Message Understanding Conference (MUC-7)
20. McCallum A, Freitag D, Pereira F (2000) Maximum entropy Markov models for information extraction and segmentation. In: Proceedings of 17th International Conference on Machine Learning, pp 591–598. Morgan Kaufmann, San Francisco, CA
21. Roark B, Johnson M (1999) Efficient probabilistic top-down and left-corner parsing. In: Proceedings of the 37th Annual Meeting of the ACL.
22. Sun A, Naing M, Lim E, Lam W (2003) Using support vector machine for terrorism information extraction. In: Proceedings of 1st NSF/NIJ Symposium on Intelligence and Security Informatics.
23. Yeh A, Hirschman L (2002) Background and overview for kdd cup 2002 task 1: information extraction from biomedical articles. KDD Explorations **4**(2), 87–89

Q3

Q4

Author Queries:

Q1. Au: Pls. provide history dates.
Q2. Au: Please provide the publisher and place in Ref. No. 2, 4, 6, 7, 8, 10, 11, 13, 16, 19, 21.
Q3. Au: Please provide the complete bibliographical details.
Q4. Au: Pls. provide biographies.