# Hybrid Semantic Tagging for Information Extraction

Ronen Feldman, Benjamin Rosenfeld, Moshe Fresko

Computer Science Department
Bar-Ilan University
Ramat Gan, ISRAEL 52900
feldman@cs.biu.ac.il

Brian D. Davison

Computer Science and Engineering
Lehigh University
Bethlehem, PA USA 18015
davison (at) cse.lehigh.edu

## ABSTRACT

The semantic web is expected to have an impact at least as big as that of the existing HTML based web, if not greater. However, the challenge lays in creating this semantic web and in converting existing web information into the semantic paradigm. One of the core technologies that can help in migration process is automatic markup, the semantic markup of content, providing the semantic tags to describe the raw content. 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 labor 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 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 amount of training data. We also demonstrate the robustness of our system under conditions of poor training data quality. This makes the system very suitable for converting legacy web pages to semantic web pages.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning

## General Terms

Algorithms, Performance, Experimentation, Languages, Theory.

## Keywords

Semantic Web, Text Mining, Information Extraction, HMM, Rules Based Systems.

## 1. INTRODUCTION

Knowledge engineering systems (mostly rule based) traditionally have been the top performers in most Information Extraction (IE) benchmarks, such as MUC [1], ACE and the KDD CUP. 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 [3].

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. In contrast, 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 upon 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.

## 2. The TEG System

### 2.1 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 $P : R \rightarrow [0..1]$ defines their probabilities. The rules have the form $n \rightarrow s_1 s_2 \ldots s_k$, 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. Similarly to a regular (non-stochastic) 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.

**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 $\rightarrow$ 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


*WWW* 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

string can be generated in many different ways. With non-stochastic grammars 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 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.

## 2.2 TEG - Using SCFG to perform IE

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 which 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 data the system would require.

## 2.3 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 refered by the rules must be declared before usage. 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 source. Examples of termlists are countries, cities, states, genes, proteins, people 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 which 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 upon 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, containing the main acquisition verb phrase. Finally, the single rule for the **Acquisition** concept is defined as a **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 behavior 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 which 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 ;
```

These twenty 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. REFERENCES


[1] Chinchor, N., L. Hirschman, and D. Lewis, *Evaluating Message Understanding Systems: An Analysis of the Third Message Understanding Conference (MUC-3).* Computational Linguistics, 1994. **3**(19): p. 409-449.

[2] Bikel, D.M., R. Schwartz, and R.M. Weischedel, *An Algorithm that Learns What's in a Name.* Machine Learning, 1999(34): p. 211–231.

[3] McCallum, A., D. Freitag, and F. Pereira. *Maximum Entropy Markov Models for Information Extraction and Segmentation*. In *Proceedings of the 17th Int'l Conf. on Machine Learning*, 2000.