A Hybrid Approach to NER by MEMM and Manual Rules

Moshe Fresko Binyamin Rosenfeld Ronen Feldman

Computer Science Department, Data Mining Lab.
Bar-Ilan University
Ramat-Gan, 52900, Israel
+972-3-5317874

freskom1@cs.biu.ac.il, grurgrur@gmail.com, feldman@cs.biu.ac.il,

ABSTRACT

This paper describes a framework for defining domain specific Feature Functions in a user friendly form to be used in a Maximum Entropy Markov Model (MEMM) for the Named Entity Recognition (NER) task. Our system called MERGE allows defining general Feature Function Templates, as well as Linguistic Rules incorporated into the classifier. The simple way of translating these rules into specific feature functions are shown. We show that MERGE can perform better from both purely machine learning based systems and purely-knowledge based approaches by some small expert interaction of rule-tuning.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis – Named Entity Recognition, Information Extraction.

General Terms

Named Entity, Information, Document Collection, Statistical Model, Empirical Model, Machine Learning, Maximum Entropy.

Keywords

Named Entity Recognition, Text Mining, Machine Learning, Information Extraction, Maximum Entropy Markov Model.

1. INTRODUCTION

There are two traditional approaches to Named-Entity recognition (NER): knowledge-based approach and machine learning approach. Knowledge-based systems usually achieve better accuracy, but require huge amounts of skilled labor by linguists and domain experts. Because of this, the recent research in NER is concentrated on machine learning techniques, which only require a manually labeled training set of documents. The best published ML-based systems perform on the level of knowledge-based systems for many categories.

In this paper we present MERGE (Maximum Entropy Rule Guided Extraction) – a hybrid NER system which combines machine learning techniques, namely ME and manually written simple rules. MERGE benefits from both approaches and can outperform both manually written rules and standard machine learning systems. The rule language of MERGE is quite simple and the amount of necessary rule-writing is relatively small, as most of the work is done by the ML part of the system.

CIKM'05, ctober 31–November 5, 2005, Bremen, Germany.
ACM 1-59593-140-6/05/0010.

2. MAXIMUM ENTROPY MODELING

A Maximum Entropy approach models a random process by making the distribution satisfy a given set of constraints, and making as few other assumptions as possible. The constraints are specified as real-valued *feature* functions over the data points. The expected value of each feature function under the ME distribution must equal the empirical expected value of function as found in the training dataset. In all other respects, the target distribution should be as uniform as possible, which means it must have the highest entropy. For our purposes, we use ME to model the *conditional* probability distributions, which slightly differ in the way expected values are calculated ([4]).

Let X be the set of conditions, usually very big, and Y the set of possible outcomes. We assume that there is a true joint distribution $P(x,y)$, but we are interested only in modeling the conditional $P(y|x)$. For this purpose we can use a training set $\{(x_k,y_k)\}_{k=1..N}$ generated by the true distribution, and a set of features $f_i: X \times Y \rightarrow \boldsymbol{R}$. Typically, the features are binary and test for specific conditions. It can be shown that the unique most uniform distribution that satisfies all feature constraints has the form:

$$(*) \quad p(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_i \lambda_i f_i(x,y) \right)$$

where λ_i–s are the parameters chosen to maximize the likelihood of the training data, and $Z(x)$ is a normalization constant, which ensures that for every x the sum of probabilities of all possible outcomes is 1. The most common procedure for parameter estimation is the Generalized Iterative Scaling algorithm ([5]).

2.1 Maximum Entropy Markov Models

A MEMM ([1]) consists of $|Y|$ conditional ME models $p_{y'}(y|x) = p(y|x,y')$, one for each y'. The model $p_{y'}(y|x)$ estimates the probability of appearance of the label y immediately after the label y' in the context x. The probability of a whole label sequence $\mathbf{y} = y_1 y_2 \ldots y_m$, given the sentence $\mathbf{x} = x_1 x_2 \ldots x_m$, is the product

$$p(\mathbf{y} \mid \mathbf{x}) = p_0(y_1 | x_1) \cdot \prod_{i=1}^{m-1} p_{y_i}(y_{i+1} | x_{i+1})$$

The best tagging can be found using Dynamic Programming similar to Vitterbi algorithm. The model $p_0(y|x)$ used at the beginning of a sentence is separate.

3. SYSTEM DETAILS

3.1 Preprocessing

The preprocessing of any text is done according to external definitions. First, the text is divided into tokens. A token is defined by a regular expression in which for an English text domain it might be in the form "[A-Za-z]+|[0-9]+|\S".

Then, each token is assigned its features according to the feature templates. A template consists of a context rule – a binary test upon the positions in the text. It can test the exact character value of the current token and its neighbors, capitalization information, membership in an external word list, arbitrary regular expressions, externally supplied features like pos tags, etc. A template can also define a generic context rule, such as "token value".

Each template defines a set of features – one feature for every combination of an instantiated context rule and a category. The features are always built in such a way that exactly one feature from a template tests true at any text position. This implies that the sum of all features is constant, satisfying the requirements of the GIS algorithm. For example, one Feature Template may have the following form:

FeatureTemplate : { (-1, Word) , (0, Capitalization) }

This template would generate a feature function for every combination of the value "Word" feature at the previous token, the value of "Capitalization" feature at the current token, and the current tag. One such feature function might be:

f = { 1 if previous token is "in" and current token is Capitalized
and current tag is Location
0 otherwise }

For Tagging, the labels are assigned at the token level – each word in a sentence is labeled by an entity type label or by label *None*. For multi-token entities we label each token by the same category label, making no distinction between beginning, middle, or ending tokens of the entity.

3.2 MERGE Rule Definition

Defining specific rules is done via a simple pattern matching language, with patterns working at the token level. A pattern syntax is similar to the regular expressions syntax, but with tokens instead of characters. Quantifiers *, +, ? are allowed, as well as the grouping parentheses "(" ")". The angular brackets "<>" delimit the target entity. Features are generated for each token in the delimited entity. The tokens are either specified directly, or represented by token-classes of the form "[Boolean expression]".The expressions are simple token-attribute=value checks, combined with logical operators &(and) |(or), and !(not).

Here is an example set of rules for some tricky organizations:

```
Rule: Organization {
      // to catch: Person of (the) Organization
: [cl=Capital] "of" "the"? < [cl=Capital]{1,3} >
      // to catch: company/firm/group called Xyz
: [wc=CompanyAlias] "called" < [cl=Capital]+ > }
WordClass: Country { france turkey israel … }
WordClass: CompanyAlias { company firm group … }
```

In our MUC-7 evaluation the list of published Word Classes are used as feature functions themselves. Besides that, some intuitive Word Classes are defined within the Rule-Development framework, in order to use them as "wc" feature for Rule Writing. As an example we had a Word Class wc=WeekDay, including the words Sunday, Monday, etc. and it is used in a Rule:

```
Rule: Date { : < ["next"|"last"] > [ wc=WeekDay] }
WordClass: WeekDay { sunday monday tuesday … }
```

The translation of Rules into a Feature Function is done by checking the rule pattern at each token position. The above rule for Date will be translated into feature function:

f = { 1 if current token is "next" or "last", and the next token is
found in wc WeekDay, and current Tag is Date,
0 otherwise }

For keeping the value M constant for each pair (x,y), we defined also this same feature function for all the other possible tags.

4. EXPERIMENTS

We made an evaluation of our methodology on MUC-7 data set, by checking the system performance on the whole possible training data (350 documents). By adding some approximately 50 more rules the overall performance reached to 93.5% as shown in Table 1, while overlapping match results come up to 95.4%. (In overlapping-match, a entity is considered correct if at least one word of it is tagged correctly) The lower overall performance is due to other entity types which had slightly lower results.

	Exact Match			OverlappingMatch		
	Rec.	Pre.	F.	Rec.	Pre.	F.
Org.	92.7	96.4	94.5	94.3	98.3	96.3
Per.	95.1	95.4	95.2	95.1	95.4	95.2
Loc.	92.3	98.4	95.2	92.9	99.0	95.9
All	91.2	95.9	**93.5**	93.0	97.8	**95.4**

Table 1. MUC7: After training with 350 documents.

We also compared our system to other successful models. We run an implementation of Nymble ([2]) and TEG ([3]) on that same corpus, which produced the results shown on Table.2.

	HMM	TEG	Merge+Rules
Org.	87.8	90.9	93.6
Per.	80.5	91.8	93.7
Loc.	90.9	91.9	95.4
Avg.	86.4	91.5	**94.2**

Table 2. Comparetive results for MUC7(without doc. Headers-Footers).

5. REFERENCES

[1] McCallum, A., Freitag, D., Pereira, F.: Maximum Entropy Markov Models for IE and Segmentation. Proc. of the 17th International Conference on Machine Learning. (2000)

[2] Bikel, D. M., Miller, S., Schwartz, R., Weischedel, R.: Nymble: a high-performance learning name-finder. Proceedings of ANLP-97. (1997) 194-201.

[3] Rosenfeld, B., Feldman, R., Fresko, M., Schler, J., Aumann, Y.: TEG - A Hybrid Approach to Information Extraction. Proc. of the 13th ACM. (2004)

[4] Berger, A., della Pietra, S., della Pietra, V.: A maximum entropy approach to NLP. Comp.Ling. 22(1), (2004) 39-71.

[5] Darroch, J. N., Ratcliff., D.: Generalized iterative scaling for log-linear models. Annals of Math.Stat.43(5):(1972)