Named Entity Extraction with Conditional Markov Models and Classifiers

Martin Jansche
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
jansche.1@osu.edu

1 Introduction

Our approach to multilingual named entity (NE) recognition in the context of the CoNLL Shared Task consists of the following ingredients:

Feature engineering A human expert (though not necessarily a language expert) determines relevant features to be used to determine whether or not a word is part of a named entity.

Extraction In a first phase a conditional Markov model extracts candidate NE phrases, but does not classify them yet into LOC, ORG, etc.

Classification In the second phase a classifier looks at candidate phrases proposed by the extractor in their sentential context and labels them as LOC, ORG, etc.

2 Feature engineering

Languages differ widely in the conventions they use to signal named entities. Spanish, French, and English use the case distinction of the modern Roman alphabet to indicate proper names, and upper case is a fairly good indicator of a proper name. The situation is quite different in German, where upper case is a poor cue. In traditional Chinese scholarly works, certain proper names are indicated by underlining, and without that form of annotation locating a proper name would seem quite challenging. In light of the diversity found across languages and orthographic conventions, it is unclear whether any effective multilingual named entity extraction system will ever be built that does not rely on human expertise for customizing it to a particular language and domain.

Since we started by building a Spanish system without knowing what other language it would have to be applied to, the features we ended up using are all rather simple and generic in nature. No language experts were consulted. In the extraction component, we look at the orthographic string of a word with accents removed (since there are some inconsistencies regarding the presence or absence of accents), and we determine whether it starts with an upper case character. For the classification component we look at entire candidate phrases, and determine the length of the phrase, its position in a sentence, the immediately surrounding words, and what words occur within the phrase. For a word inside the phrase we determine whether it starts with and upper or lower case character (or neither), whether it contains any upper case or lower case characters (or neither), and we also use the entire orthographic string with accents removed.

Fortunately, these features carry over fairly well to Dutch, the second language of the Shared Task, and may also have been sufficient for French or English, but would probably fall short for German. Needless to say, radically different orthographic systems may require entirely different approaches, so the multilingual scope of our proposal is fairly limited.

3 Extraction

The extraction component discards the specific labels LOC, ORG, etc. (from now on we refer to these as *sort labels*) and only predicts whether a token is at the beginning of (B), inside (I), or outside (O) a phrase (we will call these bare labels *phrase tags*). While this move is not unproblematic, we determined empirically that overall performance was higher using only bare phrase tags without sort labels, compared with a single-phase approach that tries to predict phrase tags and sort labels together using a single (conditional or joint) Markov model. The underlying rationale was to enable the extractor to concentrate on any morpho-syntactic regularities across different sorts of phrases without having to determine the sort label yet, which may require more context: for example, Spanish named entities can contain *de*, and this is the case across all sorts; or certain names like *Holanda* are ambiguous between LOC and ORG depending on whether they refer to countries on the one hand, or their governments or national soccer teams on the other. In light of this it makes sense to delay the assignment of sort labels and concentrate on extracting candidate phrases first.

Our extraction approach uses conditional Markov models, and we shall illustrate it using a first order model. Generalizations to higher order models are straightforwardly possible. The problem we are trying to solve is this: we want to find a sequence of phrase tags $\mathbf{t}$ given a sequence of words $\mathbf{w}$. We find the optimal $\mathbf{t}^*$ as

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}) = \arg\max_{\mathbf{t}} \frac{G(\mathbf{w},\mathbf{t})}{W(\mathbf{w})},$$

where the conditional model P is expressed in terms of a joint generative model G of tags and words, and a language model W.

Since $\mathbf{t}$ and $\mathbf{w}$ have the same length n, we regard the training data as a sequence of pairs, rather than a pair of sequences (the two representations are isomorphic via a `zip` operation familiar to Python or Haskell programmers), and decompose the generative model G using a first order Markov assumption:

$$G(\mathbf{w},\mathbf{t}) = S(w_1,t_1)\prod_{i=2}^{n} G_1(w_i,t_i|w_{i-1},t_{i-1}).$$

Doing the same for W and using a designated start event (w_0,t_0) instead of the start distribution S we obtain:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{n} \frac{G_1(w_i,t_i|w_{i-1},t_{i-1})}{W_1(w_i|w_{i-1})}.$$

We further decompose the conditional distribution G_1 as follows:

$$\begin{aligned} &G_1(w_i,t_i|w_{i-1},t_{i-1}) \\ &\quad = T(t_i|w_i,w_{i-1},t_{i-1})\, U(w_i|w_{i-1},t_{i-1}). \end{aligned}$$

In addition to the first order assumption above, the only other assumption we make is that $U(w_i|w_{i-1},t_{i-1}) = U(w_i|w_{i-1})$, which allows us to conclude that $U = W_1$, and so our conditional sequence model simplifies to

$$P(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{n} T(t_i|w_i,w_{i-1},t_{i-1}).$$

This is starting to look familiar: T is a conditional distribution over a finite set of phrase tags, so in principle any probabilistic classifier that uses (features derived from) the variables that T is conditioned on could be substituted in its place. Approaches like this have apparently been used informally in practice for some time, perhaps with a classifier instead of T that does not necessarily return a proper probability distribution over tags. Probability models that predict the next tag conditional on the current tag and an observed word have been criticized for a weakness known as the Label Bias Problem (Lafferty et al., 2001); on the other hand, the practical effectiveness of approaches like the one proposed here for a very similar task was demonstrated by Punyakanok and Roth (2001).

Finding the optimal tag sequence for a given sequence of words can be done in the usual fashion using Viterbi decoding. Training is fully supervised, since we have labeled training data, but could in principle be extended to the (partly) unsupervised case. We only implemented supervised training, which is mostly trivial. When using a simple conditional next-tag model it is especially important to have good estimates of $T(t_i|w_i,w_{i-1},t_{i-1})$. We use a strategy of backing off to less and less informative contexts. In the worst case, $T(t_i|t_{i-1})$ can be estimated very reliably from the training data (in fact, good estimates for much longer tag histories can be found). When conditioning on words, the situation is rather different. For example, we see relatively few events of the form (w_i,w_{i-1},t_{i-1}) in the training data (out of the space of all possible events of that form), and so we may back off to (w_i,u_{i-1},t_{i-1}), where u_{i-1} is binary valued and indicates whether the preceding word started with an upper case letter. We have not determined an optimal back-off strategy, and for now we use an intuitively plausible strategy that tries to use as much conditioning information as possible and backs off to strictly less informative histories. In all cases it is important to always condition on the preceding tag t_{i-1}, or else we would be left with no information about likely tag sequences.

We used first and second order models of this form and manually searched for good parameter settings on a held-out portion of the training data. It turns out that the second order model performs about the same as the first order model, but is at a disadvantage because of data sparseness. Therefore we only consider first order models in the rest of this paper. The performance of the first order model on the development data sets is summarized in Table 1. Note that these figures can be obtained for any system by first piping its output through `sed` using the command `s/-\(LOC\|MISC\|ORG\|PER\)/-FOO/g`. As will become clearer below, within each language it so happens that the extraction component performs better than the classification component, i.e. for now the performance bottleneck is the classification component.

Spanish dev.	precision	recall	$F_{\beta=1}$
overall	87.60%	86.86%	87.23

Dutch devel.	precision	recall	$F_{\beta=1}$
overall	85.84%	84.55%	85.19

Table 1: Extraction results obtained for the development data sets for the two languages used in this shared task.

4 Classification

The candidate phrases proposed by the extraction component are subsequently annotated with sort labels. The main advantage of dividing up the task this way is that we can take a lot more context into account for classifying phrases. For example, features that may be relevant now include: the length of the phrase, the first/last k words in the phrase, the position of the phrase in the sentence, whether the words *fútbol* or *liga* were mentioned in the same sentence, etc. Such features would be awkward to incorporate into a single-phase approach using a Markov model to predict phrase tags at the same time as sort labels.

We chose a fairly standard independent feature (a.k.a. "naive Bayes") model, mostly as a matter of convenience. Obviously any other classifier framework could have been used instead. For both languages we use as features the length of the phrase, its distance from the start of the sentence, the identity of the words inside the phrase viewing it as a set of words (i.e., discarding positional information), the identity and other properties (including whether a word starts with an upper/lower case letter) of the first k and last k' words in the phrase, and the identity and other properties of the word(s) preceding and following the phrase. The optimal parameter settings differ for Spanish and Dutch. For example, in Spanish the identities of the first $k = 6$ words is very important for classification performance, whereas long preceding or trailing contexts do not help much, if at all. For Dutch, the identities of words inside the phrase is less helpful ($k = 3$ is optimal), and more preceding and trailing context has to be used. In addition, knowing whether a sentence (or, ideally, a news article) is about soccer was helpful for Spanish. A feature that tests for the presence of *fútbol* and a few semantically related words is the only aspect of the classification component that is particular to one language. Other language specific information, e.g. names of Spanish provinces, did not turn out to be useful.

Table 2 shows performance figures for the classification component on the raw development data. Equivalently one can think of these results as if we had applied our classifiers to the output of a perfect extraction component that does not make any mistakes. We can already see for Spanish that performance is lowest for the sort MISC, which does not seem very homogeneous, and may perhaps best be chosen by default if no other class applies. Trying to predict MISC directly seems to be a misguided effort. This will become even clearer below when we look at the overall performance of our approach.

Spanish dev.	precision	recall	$F_{\beta=1}$
LOC	71.37%	87.82%	78.74
MISC	70.80%	79.55%	74.92
ORG	87.95%	78.18%	82.78
PER	91.05%	84.12%	87.45
overall	82.17%	82.17%	82.17

Dutch devel.	precision	recall	$F_{\beta=1}$
LOC	75.57%	70.17%	72.77
MISC	79.68%	80.43%	80.05
ORG	82.30%	66.42%	73.51
PER	70.21%	85.88%	77.26
overall	76.39%	76.39%	76.39

Table 2: Classification results obtained for the development data sets for the two languages used in this shared task.

5 Putting it all together

A theoretical problem with our task decomposition is how to train the classifiers used in the second phase. What they will eventually see as input is the output of the extraction component, which may contain mistakes, e.g., cases where the beginning or end of a phrase was mispredicted. Since we want to build and refine the classification component independently of the extraction component, we have to train the classifiers on the phrases in the labeled training data. It is not clear a priori that this kind of independent development comes without a performance penalty, since we may have forgotten to show the second-phase classifiers examples of truncated or badly mangled phrases that were produced because of imperfections of the extraction component which makes up the first phase of our approach. Based on the independence assumption behind the task decomposition we would expect the overall performance on the Spanish development data set to be

$$0.8723 \times 0.8217 \approx 0.7168.$$

As we can see from the actual results in Table 3, this is not very far from the observed performance. We conclude that independent development of the two components did not impact overall performance.

6 Conclusion

We presented a simple, knowledge-poor named entity recognizer using standard components. Our decomposition into extraction and classification phases was motivated by the common syntactic regularities and the ambiguous status of some named entities. We have shown that the conditional next-tag model used for extraction is not unprincipled (a criticism brought forward by McCallum et al. (2000) against next-tag classifiers that do not output probabilities), but arises naturally from a conditional sequence model and plausible independence assumptions. This extraction model achieves fairly high accuracy (and just as observed by Punyakanok and Roth (2001) it outperforms a joint generative Markov model). A separate classification step makes it easy to use sentence-level features and large amounts of contexts. Such features would be difficult to integrated into standard models, the major exception being conditional random fields (Lafferty et al., 2001), compared to which the approach proposed here is much simpler.

Spanish dev.	precision	recall	$F_{\beta=1}$
LOC	65.76%	76.45%	70.70
MISC	45.56%	54.16%	49.49
ORG	77.40%	70.12%	73.58
PER	85.01%	76.60%	80.59
overall	72.35%	71.74%	72.04

Spanish test	precision	recall	$F_{\beta=1}$
LOC	77.18%	73.34%	75.21
MISC	44.47%	50.88%	47.46
ORG	76.75%	77.57%	77.16
PER	80.20%	77.69%	78.92
overall	74.03%	73.76%	73.89

Dutch devel.	precision	recall	$F_{\beta=1}$
LOC	69.87%	67.23%	68.52
MISC	64.69%	62.87%	63.77
ORG	69.65%	56.11%	62.15
PER	63.93%	75.85%	69.38
overall	66.42%	65.43%	65.92

Dutch test	precision	recall	$F_{\beta=1}$
LOC	79.04%	76.78%	77.89
MISC	69.60%	59.98%	64.43
ORG	64.27%	63.17%	63.71
PER	69.21%	78.80%	73.70
overall	70.11%	69.26%	69.68

Table 3: Results obtained for the development and the test data sets for the two languages used in this shared task.

References

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistics models for segmenting and labeling sequence data. In *ICML-01*, pages 282–289.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *ICML 2000*, pages 591–598.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13*, pages 995–1001.