Named Entity Recognition with Character-Level Models

Dan Klein
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
klein@cs.stanford.edu

Joseph Smarr
Symbolic Systems Program
Stanford University
Stanford, CA 94305-2181
jsmarr@stanford.edu

Huy Nguyen
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
htnguyen@stanford.edu

Christopher D. Manning
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
manning@cs.stanford.edu

Abstract

We discuss two named-entity recognition models which use characters and character n-grams either exclusively or as an important part of their data representation. The first model is a character-level HMM with minimal context information, and the second model is a maximum-entropy conditional markov model with substantially richer context features. Our best model achieves an overall F_1 of 86.07% on the English test data (92.31% on the development data). This number represents a 25% error reduction over the same model without word-internal (substring) features.

1 Introduction

For most sequence-modeling tasks with word-level evaluation, including named-entity recognition and part-of-speech tagging, it has seemed natural to use entire words as the basic input features. For example, the classic HMM view of these two tasks is one in which the observations are words and the hidden states encode class labels. However, because of data sparsity, sophisticated unknown word models are generally required for good performance. A common approach is to extract word-internal features from unknown words, for example suffix, capitalization, or punctuation features (Mikheev, 1997, Wacholder et al., 1997, Bikel et al., 1997). One then treats the unknown word as a collection of such features. Having such unknown-word models as an add-on is perhaps a misplaced focus: in these tasks, providing correct behavior on unknown words is typically the key challenge.

Here, we examine the utility of taking character sequences as a primary representation. We present two models in which the basic units are characters and character n-grams, instead of words and word phrases. Earlier papers have taken a character-level approach to named entity recognition (NER), notably Cucerzan and Yarowsky (1999), which used prefix and suffix tries, though to our knowledge incorporating all character n-grams is new. In section 2, we discuss a character-level HMM, while in section 3 we discuss a sequence-free maximum-entropy (maxent) classifier which uses n-gram substring features. Finally, in section 4 we add additional features to the maxent model, and chain these models into a conditional markov model (CMM), as used for tagging (Ratnaparkhi, 1996) or earlier NER work (Borthwick, 1999).

2 A Character-Level HMM

Figure 1 shows a graphical model representation of our character-level HMM. Characters are emitted one at a time, and there is one state per character. Each state's identity depends only on the previous state. Each character's identity depends on both the current state and on the previous $n-1$ characters. In addition to this HMM view, it may also be convenient to think of the local emission models as type-conditional n-gram models. Indeed, the character emission model in this section is directly based on the n-gram proper-name classification engine described in (Smarr and Manning, 2002). The primary addition is the state-transition chaining, which allows the model to do segmentation as well as classification.

When using character-level models for word-evaluated tasks, one would not want multiple characters inside a single word to receive different labels. This can be avoided in two ways: by explicitly locking state transitions inside words, or by careful choice of transition topology. In our current implementation, we do the latter. Each state is a pair (t, k) where t is an entity type (such as *PERSON*, and including an *other* type) and k indicates the length of time the system has been in state t. Therefore, a state like (*PERSON*, 2) indicates the second letter inside a person phrase. The final letter of a phrase is a following space (we insert one if there is none) and the state is a special final state like (*PERSON*, F). Additionally, once k reaches our n-gram history order, it stays there. We then use empirical, unsmoothed estimates for state-

Description	ALL	LOC	MISC	ORG	PER
Official Baseline	71.2	80.5	83.5	66.4	55.2
Word-level HMM	74.5	79.5	69.7	67.5	77.6
Char-level, no conx	82.2	86.1	82.2	73.4	84.6
Char-level, context	**83.2**	**86.9**	**83.0**	**75.1**	**85.6**

Table 1: HMM F_1 performance, English development set.

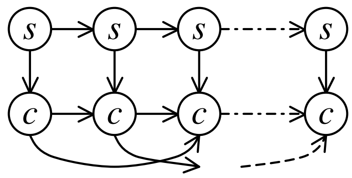

Figure 1: A character-level HMM. The c nodes are character observations and s nodes are entity types.

state transitions. This annotation and estimation enforces consistent labellings in practice. For example, (*PERSON*, 2) can only transition to the next state (*PERSON*, 3) or the final state (*PERSON*, F). Final states can only transition to beginning states, like (*other*, 1).

For emissions, we must estimate a quantity of the form $P(c_0|c_{-(n-1)}, \ldots, c_{-1}, s)$, for example, P(s|Thoma, *PERSON*, 6).[1] We use an n-gram model of order $n = 6$.[2] The n-gram estimates are smoothed via deleted interpolation.

Given this model, we can do Viterbi decoding in the standard way. To be clear on what this model does and does not capture, we consider a few examples (_ indicates a space). First, we might be asked for P(e|to_Denv, *LOC*, 5). In this case, we know both that we are in the middle of a location that begins with Denv and also that the preceding context was to. In essence, encoding k into the state lets us distinguish the beginnings of phrases, which lets us model trends like named entities (all the classes besides *other*) generally starting with capital letters in English. Second, we may be asked for quantities like P(_|Italy, *LOC*, *F*), which allows us to model the ends of phrases. Here we have a slight complexity: by the notation, one would expect such emissions to have probability 1, since nothing else can be emitted from a final state. In practice, we have a special stop symbol in our n-gram counts, and the probability of emitting a space from a final state is the probability of the n-gram having chosen the stop character.[3]

[1] We index characters, and other vector elements by relative location subscripts: c_0 is the current character, c_1 is the following character, and c_{-1} is the previous character.

[2] The smaller space of characters allows us to obtain dense estimates for longer n-grams than is possible with word-level models. The value $n = 6$ was the empirically optimal order.

[3] This can be cleaned up conceptually by considering the entire process to have been a hierarchical HMM (Fine et al., 1998), where the n-gram model generates the entire phrase, followed by a tier pop up to the phrase transition tier.

Using this model, we tested two variants, one in which preceding context was discarded (for example, P(e|to_Denv, *LOC*, 5) was turned into P(e|xx_Denv, *LOC*, 5)), and another where context was used as outlined above. For comparison, we also built a first-order word-level HMM; the results are shown in table 1. We give F_1 both per-category and overall. The word-level model and the (context disabled) character-level model are intended as a rough minimal pair, in that the only information crossing phrase boundaries was the entity type, isolating the effects of character- vs word-level modeling (a more precise minimal pair is examined in section 3). Switching to the character model raised the overall score greatly, from 74.5% to 82.2%. On top of this, context helped, but substantially less, bringing the total to 83.2%.

We did also try to incorporate gazetteer information by adding n-gram counts from gazetteer entries to the training counts that back the above character emission model. However, this reduced performance (by 2.0% with context on). The supplied gazetteers appear to have been built from the training data and so do not increase coverage, and provide only a flat distribution of name phrases whose empirical distributions are very spiked.

3 A Character-Feature Based Classifier

Given the amount of improvement from using a model backed by character n-grams instead of word n-grams, the immediate question is whether this benefit is complementary to the benefit from features which have traditionally been of use in word level systems, such as syntactic context features, topic features, and so on.

To test this, we constructed a maxent classifier which locally classifies single words, without modeling the entity type sequences s.[4] These local classifiers map a feature representation of each word position to entity types, such as *PERSON*.[5] We present a hill-climb over feature sets for the English development set data in table 2. First, we tried only the local word as a feature; the result was that each word was assigned its most common class in the training data. The overall F-score was 52.29%, well below the official CoNLL baseline of 71.18%.[6] We next added n-gram features; specifically, we framed each word with special start and end symbols, and then added every contiguous substring to the feature list. Note that this subsumes the entire-word features. Using the substring features alone scored 73.10%, already breaking the

[4] The classifier was trained using conjugate gradient descent, used equal-scale gaussian priors for smoothing, and learned models of over 800K features in approximately 2 hours.

[5] The B-/I- distinction in the data was collapsed, though see section 4.

[6] The latter assigns phrases at once, which is generally superior, but is noticeably worse at multi-word person names, since it cannot synthesize new first-name/last-name pairs.

Description	Added Features	ALL	LOC	MISC	ORG	PER
Words	w_0	52.29	41.03	70.18	60.43	60.14
Official Baseline	–	71.18	80.52	83.52	66.43	55.20
NGrams	$n(w_0)$	73.10	80.95	71.67	59.06	77.23
Tags	t_0	74.17	81.27	74.46	59.61	78.73
Simple Context	w_{-1}, w_0, t_{-1}, t_1	82.39	87.77	82.91	70.62	85.77
More Context	$\langle w_{-1}, w_0\rangle, \langle w_0, w_1\rangle, \langle t_{-1}, t_0\rangle, \langle t_0, t_1\rangle$	83.09	89.13	83.51	71.31	85.89
Simple Sequence	$s_{-1}, \langle s_{-1}, t_{-1}, t_0\rangle$	85.44	90.09	80.95	76.40	89.66
More Sequence	$\langle s_{-2}, s_{-1}\rangle, \langle s_{-2}, s_{-1}, t_{-2}, t_{-1}, t_0\rangle$	87.21	90.76	81.01	81.71	90.80
Final	(see text)	**92.27**	**94.39**	**87.10**	**88.44**	**95.41**

Table 2: CMM performance with incrementally added features on the English development set.

the phrase-based CoNLL baseline, though lower than the no-context HMM, which better models the context inside phrases. Adding a current tag feature gave a score of 74.17%. At this point, the bulk of outstanding errors were plausibly attributable to insufficient context information. Adding even just the previous and next words and tags as (atomic) features raised performance to 82.39%. More complex, joint context features which paired the current word and tag with the previous and next words and tags raised the score further to 83.09%, nearly to the level of the HMM, still without actually having any model of previous classification decisions.

4 A Character-Based CMM

In order to include state sequence features, which allow the classifications at various positions to interact, we have to abandon classifying each position independently. Sequence-sensitive features can be included by chaining our local classifiers together and performing joint inference, i.e., by building a conditional markov model (CMM), also known as a maximum entropy markov model (McCallum et al., 2000).

Previous classification decisions are clearly relevant: for example the sequence `Grace Road` is a single location, not a person's name adjacent to a location (which is the erroneous output of the model in section 3). Adding features representing the previous classification decision (s_{-1}) raised the score 2.35% to 85.44%. We found knowing that the previous word was an *other* wasn't particularly useful without also knowing its part-of-speech (e.g., a preceding preposition might indicate a location). Joint tag-sequence features, along with longer distance sequence and tag-sequence features, gave 87.21%.

The remaining improvements involved a number of other features which directly targetted observed error types. These features included letter type pattern features (for example `20-month` would become `d-x` for digit-lowercase and `Italy` would become `Xx` for mixed case). This improved performance substantially, for example allowing the system to detect ALL CAPS regions. Table 4 shows an example of a local decision for `Grace` in the context `at Grace Road`, using all of the features defined to date. Note that the evidence against `Grace` as a name completely overwhelms the n-gram and word preference for *PERSON*. Other features included second-previous and second-next words (when the previous or next words were very short) and a marker for capitalized words whose lowercase forms had also been seen. The final system also contained some simple error-driven post-processing. In particular, repeated sub-elements (usually last names) of multi-word person names were given type *PERSON*, and a crude heuristic restoration of B- prefixes was performed. In total, this final system had an F-score of 92.31% on the English development set. Table 3 gives a more detailed breakdown of this score, and also gives the results of this system on the English test set, and both German data sets.

5 Conclusion

The primary argument of this paper is that character substrings are a valuable, and, we believe, underexploited source of model features. In an HMM with an admittedly very local sequence model, switching from a word model to a character model gave an error reduction of about 30%. In the final, much richer chained maxent setting, the reduction from the best model minus n-gram features to the reported best model was about 25% – smaller, but still substantial. This paper also again demonstrates how the ease of incorporating features into a discriminative maxent model allows for productive feature engineering.

English dev.	Precision	Recall	$F_{\beta=1}$
LOC	94.44	94.34	94.39
MISC	90.62	83.84	87.10
ORG	87.63	89.26	88.44
PER	93.86	97.01	95.41
Overall	92.15	92.39	92.27

English test	Precision	Recall	$F_{\beta=1}$
LOC	90.04	89.93	89.98
MISC	83.49	77.07	78.85
ORG	82.49	78.57	80.48
PER	86.66	95.18	90.72
Overall	86.12	86.49	86.31

German dev.	Precision	Recall	$F_{\beta=1}$
LOC	75.53	66.13	70.52
MISC	78.71	47.23	59.03
ORG	77.57	53.51	63.33
PER	72.36	71.02	71.69
Overall	75.36	60.36	67.03

German test	Precision	Recall	$F_{\beta=1}$
LOC	78.01	69.57	73.54
MISC	75.90	47.01	58.06
ORG	73.26	51.75	60.65
PER	87.68	79.83	83.57
Overall	80.38	65.04	71.90

Table 3: Final results obtained for the development and test sets for each language on the shared task.

	PPREV	PREV	**CUR**	NEXT
States	O	O	**LOC**	LOC
Words	morning	at	**Grace**	Road
Tags	NN	IN	**NNP**	NNP
Types	x	x:2	**Xx**	Xx

	O	LOC	MISC	ORG	PER
WORDS					
PWORD:at	-0.18	0.94	-0.31	0.28	-0.73
CWORD:Grace	-0.01	0	0	-0.02	0.03
NWORD:Road	0.02	0.27	-0.01	-0.25	-0.03
PWORD-CWORD:at-Grace	0	0	0	0	1 0
CWORD-NWORD:Grace-Road	0	0	0	0	0
NGRAMS (prefix/suffix only here)					
⟨G	-0.57	-0.04	0.26	-0.04	0.45
⟨Gr	0.27	-0.06	0.12	-0.17	-0.16
⟨Gra	-0.01	-0.37	0.19	-0.09	0.28
⟨Grac	-0.01	0	0	-0.02	0.03
⟨Grace	-0.01	0	0	-0.02	0.03
⟨Grace⟩	-0.01	0	0	-0.02	0.03
Grace⟩	-0.01	0	0	-0.02	0.03
race⟩	0	0	0	-0.02	0.03
ace⟩	0.08	0.24	0.07	-0.30	-0.10
ce⟩	0.44	0.31	-0.34	-0.02	-0.38
e⟩	0.38	-0.14	-0.18	-0.06	0
TAGS					
PTAG:IN	-0.40	0.24	0.16	0.08	-0.08
CTAG:NNP	-1.09	0.45	-0.26	0.43	0.47
NTAG:NNP	0.05	-0.19	0.18	-0.12	0.08
PTAG-CTAG:IN-NNP	0	0.14	-0.03	-0.01	-0.10
CTAG-NTAG:NNP-NNP	-0.11	-0.05	0	-0.38	-0.54
TYPES					
PTYPE:x:2	-0.07	-0.15	0.35	0.18	-0.31
CTYPE:Xx	-2.02	0.46	0.19	0.57	0.80
NTYPE:Xx	-0.22	-0.42	-0.19	0.29	0.54
PTYPE-CTYPE:x:2-Xx	-0.20	0.08	0.10	0.10	-0.09
CTYPE-NTYPE:Xx-Xx	0.55	-0.13	-0.55	-0.13	0.26
PTYPE-CTYPE-NTYPE:x:2-Xx-Xx	0.10	0.37	0.10	0.12	-0.69
WORDS/TYPES					
PWORD-CTYPE:at-Xx	-0.21	0.57	-0.21	0.41	-0.56
CTYPE-NWORD:Xx-Road	-0.01	0.27	-0.01	-0.23	-0.03
STATES					
PSTATE:O	2.91	-0.92	-0.72	-0.58	-0.70
PPSTATE-PSTATE:O-O	1.14	-0.60	-0.08	-0.43	-0.04
WORDS/STATES					
PSTATE-CWORD:O-Grace	-0.01	0	0	-0.02	0.03
TAGS/STATES					
PSTATE-PTAG-CTAG:O-IN-NNP	0.12	0.59	-0.29	-0.28	-0.14
PPSTATE-PPTAG-PSTATE-PTAG-CTAG:O-NN-O-IN-NNP	0.01	-0.03	-0.31	0.31	0.01
TYPES/STATES					
PSTATE-CTYPE:O-Xx	-1.13	0.37	-0.12	0.20	0.68
PSTATE-NTYPE:O-Xx	-0.69	-0.3	0.29	0.39	0.30
PSTATE-PTYPE-CTYPE:O-x:2-Xx	-0.28	0.82	-0.10	-0.26	-0.20
PPSTATE-PPTYPE-PSTATE-PTYPE-CTYPE:O-x-O-x:2-Xx	-0.22	-0.04	-0.04	-0.06	0.22
Total:	-1.40	2.68	-1.74	-0.19	-0.58

Table 4: Example of the features and weights at a local decision point: deciding the classification of `Grace`.

References

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201.

Andrew Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Joint SIGDAT Conference on EMNLP and VLC*.

Shai Fine, Yoram Singer, and Naftali Tishby. 1998. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62.

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

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP 1*, pages 133–142.

Joseph Smarr and Christopher D. Manning. 2002. Classifying unknown proper noun phrases without context. Technical Report dbpubs/2002-46, Stanford University, Stanford, CA.

Nina Wacholder, Yael Ravin, and Misook Choi. 1997. Disambiguation of proper names in text. In *ANLP 5*, pages 202–208.