Bootstrapping for Entity Recognition with Type Annotations and Syntax Features

William Bryce*, Timothy Miller†, Lifeng Jin‡, Victoria Krakovna§,
Finale Doshi-Velez§, William Schuler‡, Lane Schwartz*
* University of Illinois
† Boston Children's Hospital and Harvard Medical School
‡ The Ohio State University
§ Harvard University

***Abstract*—This paper describes an entry to the 2016 LoreHLT named entity recognition task that attempted to leverage a grammar induction system to provide features that would be an improvement over simple word-based features. The system used was an unsupervised Bayesian Hierarchical Hidden Markov Model implementation of a left-corner parser [van Schijndel et al., 2013]. Our primary submission used a conditional random field to do bootstrapping, with lexical features extracted from the surface text and syntactic features extracted from parses of our grammar induction system. While our syntactic features did not now benefit the task of named entity extraction, we conclude that this may have been due to the fact that, at time of evaluation, the grammar induction system parsed sentences without recursion, or that features extracted closer to system convergence would have fared better. Otherwise, our syntactic features, which have contributed to performance on parsing and part of speech induction tasks, may not have benefitted named entity extraction because the task itself may not be syntactic in nature.**

I. Introduction

This paper describes an entry to the 2016 LoreHLT named entity recognition task that attempted to leverage a grammar induction system to provide features that would be an improvement over simple word-based features. The 2016 LoreHLT challenge consisted of three tasks: Named entity recognition, machine translation, and situation frame extraction. The evaluation language was Uyghur, and was divided up into sets, with set 0 containing the vast majority of the data. Our basic approach was a bootstrapping system that, starting from some minimal number of labeled annotations, could self-train, re-label the training set, re-train, and so on, until some stopping criteria was met. We used multilingual gazetteers and a native informant to provide type annotations rather than instance annotations and used these type annotations to label a large number of instances.

This approach was somewhat inspired by previous work in fast annotations carried out on part of speech (POS) tags [Garrette and Baldridge, 2013]. Our vision in the LORELEI project [Onyshkevych, 2014] in general is that unsupervised systems can take advantage of statistical regularities in linguistic sequences to tag tokens with syntactic information. Even though Garrette and Baldridge only spend a few hours of annotation for building a POS tagging system, even that time might be better spent performing higher level annotations (e.g, named entity annotations), which often require even less expertise. For this particular task, we hypothesized that named entity types were unambiguous enough that we could multiply our annotator efforts by having them label types instead of instances. The string *Barack Obama*, for example, is highly likely to be a person (PER) tag in every single instance.

Our primary submission uses a conditional random field to do bootstrapping, with lexical features extracted from the surface text and syntactic features extracted from parses of our grammar induction system. We also submitted many variants of the primary submission, which we will detail below in the evaluation section.

II. Background

A. Bootstrapping for Supervised Classification

Bootstrapping offers an approach to NLP applications on languages with little available human-annotated, gold standard data. It requires only a small amount of training data which can be gathered in a short amount of time. A bootstrapping system trains on a handful of examples, or seeds. It selects more examples from unlabeled data, and adds the new examples to the seed set for further training on the next iteration. This process is repeated iteratively

until some stopping condition is reached [Abney, 2002], [Blum and Mitchell, 1998].

Yarowsky [Yarowsky, 1995] used bootstrapping to attain very good results on disambiguation of keywords in context. The technique has also been applied to named entity recognition, using rules indicating named entities as seeds for other rules [Strzalkowski and Wang, 1996], or similarly to our approach, using a handful of entities as seeds to discover more entities [Putthividhya and Hu, 2011]. There is also precedent for discovering seeds from raw text by use of a gazetteer, generated either by hand or automatically via unsupervised methods [Kozareva, 2006].

B. Unsupervised Hierarchical Hidden Markov Models

The system we used for generating our syntax features is an unsupervised Bayesian Hierarchical Hidden Markov Model implementation of a left-corner parser [van Schijndel et al., 2013]. A left-corner parser hypothesizes stacks of derivation fragments consisting of an active category lacking an awaited category yet to come, and proceeds by (nondeterministically) forking off words as new derivation fragments and joining up derivation fragments with syntactic branches. To simplify training, this implementation is constrained to hypothesize stacks of no more than one derivation fragment. Like Bayesian unsupervised part of speech tagging [Johnson, 2007], [van Gael et al., 2009], the system treats a sentence as a sequence to tag, and alternates between sampling tag sequences for each sentence in the corpus and re-estimating models based on the most recently acquired tag samples. The unsupervised HHMM system (UHHMM), however, contains more variables than the standard hidden Markov models used for unsupervised POS tagging. Specifically, at each time step, the system contains three grammar variables, an *Active* variable, an *Awaited* variable, and a *Pos* variable. A supervised version of this system has obtained results on par with state of the art on the Wall Street Journal section of the Penn Treebank, and detailed descriptions of the parsing model can be found in other work [van Schijndel et al., 2013].

To convert the supervised HHMM system into an unsupervised grammar induction system, we define linguistically motivated priors over all the probability distributions in the model. Specifically, the priors encourage sparse distributions which correspond to the commonly seen Zipfian distributions in linguistic data. For example, some parts of speech occur frequently while others less so; within parts of speech some words are common while there is a long tail of infrequent words. This is modeled in our system by having the Categorical distributions over words, for example, governed by a Dirichlet distribution with hyperparameters < 1 that encourage sparsity.

III. Named Entity Task System Description

A. Core Bootstrapping Algorithm

Our system uses a bootstrapping approach to discover and categorize named entities (NEs). This system is a modified version of the NER system included with the LDC BOLT language data used for training and testing on this task. Our novel contributions are the use of syntactic features modeled by our UHHMM and our implementation of a bootstrapping approach. For each bootstrapping iteration, we trained a conditional random field (CRF) with CRFSuite [Okazaki, 2007] with LBFGS optimization; we set no maximum number of iterations for CRF training, and we set L1 and L2 values each to 1.0.

As an initial step, we compiled a multilingual gazetteer, G, containing exemplars of NE types that are used as the seeds in our bootstrapping process. For details on constructing the gazetteers used for this study, see section III.D below. The raw Uyghur data was scanned for tokens and token spans matching NEs within G. For each NE $g \in G$ whose language is identified as Uyghur, if a span matching g was found in the raw Uyghur data, that span was identified as an NE with type corresponding to g. From the resulting annotated dataset, T_0, we derived lexical and syntactic features (described below) from the set of annotated NEs, E_0, within T_0. These features were used to train the CRF to identify more NEs during the first iteration of bootstrapping.

Generally, for each iteration of bootstrapping, i_n: $n = 1...\infty$, the CRF is trained on the features of the set of NEs, E_{n-1} found in the annotated dataset, T_{n-1}. The CRF assigns to each span a probability of being an NE, either a single token NE or part of an NE which is a contiguous span of tokens. In order for our system to identify a single token as an NE, its probability must be greater than or equal to some threshold parameter, t_n. Our system identifies a span of multiple tokens as an NE if the mean of the probabilities assigned to all tokens in that span is greater than or equal to t_n. The threshold t_1 is initialized for i_1, and for each subsequent iteration, i_n, the threshold $t_n = t_{n-1} - d$, where d is a constant decrement parameter value. Any new NEs identified by the CRF at i_n are added to a set E_n. Once all new NEs have been identified, E_n is set to $E_{n-1} \cup E_n$. The raw Uyghur data is then annotated with E_n, thereby creating

the dataset T_n. If T_n is equivalent to T_{n-1}, i.e. if no new NEs were identified by the CRF, then bootstrapping ceases and T_n is the output of the system with E_n being the final set of identified NEs. Bootstrapping may also cease if the value of t_n falls below a minimum value parameter, in which case T_{n-1} and E_{n-1} are the outputs of the system.

B. Features

The features used by our system are derived from individual tokens within the text. For every token tok_i, we consider a window including tok_{i-2} through tok_{i+2}. From every token in this window, we take the following set of lexical features: the token itself; the lowercased token; whether the first character of the token is capitalized; whether all characters are capitalized; whether the token is all digits; whether the token is all non-alphanumerics; whether the token contains a period; and the first and last x characters of a token, where $x = 1...4$.

We also take syntactic features for every token in the window from the parsed sentence output by our UHHMM. These syntactic features are the *active*, *awaited*, and *pos* variables described above, as well as variables signifying whether the parsed tree undergoes a *fork* operation (when a word forks off a new derivation fragment) and/or a *join* operation (when a branch joins a derivation fragment to the previous derivation fragment) at the token; two features signify these variables independently and another encodes their values as one feature. Finally, we have a feature for whether the token is the first in the sentence and a feature for whether the token is the second in the sentence. We included these last two features because the values of *fork* and *join* for the first and second tokens of a sentence are consistently different from those of other tokens.

In some of our experiments, our bootstrapping system selects features at each iteration. In these experiments, the CRF is trained at each iteration on the set of features selected by the previous iteration, F_{n-1}. For the first iteration, the CRF is trained on the full set of features, F_0. For each iteration, i_n, for each feature $f \in F_{n-1}$, that feature f is assigned a score, s_{fn}, which is the sum of the absolute values of the coefficients of f in the CRF model trained during i_n. Features with greater s_{fn} are selected and placed in F_n. The cardinality of F_n is a percentage p of the cardinality of F_{n-1}, where $p_n = 1 - t_n$. As t decreases with each iteration, p increases, thus eliminating a greater percentage of the original feature set during earlier iterations than later ones.

C. Corpus Pre-processing

The data resources provided by the Linguistic Data Consortium for the task were larger than previous packs and provided a challenge for our induction system, and even some of the tools we used to preprocess previous corpora. In order to make the task more manageable, we extracted a smaller subset of sentences from the corpora that match the strengths of our fastest running induction system. In particular, we have evaluated extensively on sentences of length ten or less – this was at one point a common evaluation setup for the grammar induction task. For the baseline non-recursive system we developed during year one of LORELEI, length ten sentences are a good fit because they are unlikely to contain much recursion. We have also noted previously that by training on shorter sentences, our system converges faster and yet can still parse longer sentences, extracting what are essentially chunks when it encounters complex syntax that exceeds its memory constraints.

For the first checkpoint we filtered the data to remove all sentences of more than three and less than ten. We noticed that one- or two-word sentences were surprisingly common, but do not contribute any syntactic structure, and thus take up too much model space. We next replaced any words that occurred with frequency less than two with a special *UNK* token. This change allows our system to learn approximately how rare words should be distributed so that it can adapt to new tokens at test time. For checkpoint one we limited ourselves to these changes, wanting to see how much we could learn in one week with minimal deviations.

In reviewing the performance of the induction system after checkpoint one, we noticed that the system still did not appear to have converged. We took this as an opportunity to explore another mechanism for speeding grammar induction. The approach we took was based on the hypothesis that a smaller vocabulary would be easier to acquire. To examine this, we first sorted all lexical types in the corpus by occurrence frequency, and took the first 50,000 as a vocabulary. We then filtered the corpus by only allowing sentences which used terms in the 50,000 word vocabulary.

For checkpoint two our system still had not completely converged, so we took advantage of the longer time between checkpoints two and three to try to run our system to convergence. We therefore did not change any of the preprocessing for this time frame.

D. Primary and Alternative Submissions

At each checkpoint we submitted multiple runs to explore different configurations as well as our bootstrapping starting points. The systems are described briefly in the following lists, with parenthesized names referring to rows in Table I, which contains system scores.

Checkpoint One

- Raw wiki gazetteer (Raw) – Text spans matching entities found in the multilingual Wikipedia language links dataset are labeled with the type in the gazetteer. No learning or other processing is performed.
- Native informant gazetteer baseline (NI) – Text spans matching entities found in the gazetteer built during a one hour session with the native informant are labeled with the type found in the gazetteer.
- Bootstrapping CRF with baseline features (Words) – Used the bootstrapping approach described above, with initial labeling done by native informant gazetteer and supervised machine learning done with the default set of features included in the LDC-distributed tagger
- Primary (Words+Syn) – Used the bootstrapping approach described above, with initial labeling done by native informant gazetteer and supervised machine learning done with additional syntactic features extracted from UHHMM system output.

The first two systems are designed to set a floor for system performance and evaluate the quality of our gazetteers. The first gazetteer was extracted by harvesting multi-lingual information from the English language Wikipedia. Each Wikipedia page has a list of links to the same page in other languages for every language in which the page exists. We manually created a list of prototype named entities for each category, found the English Wikipedia page for each entity, and harvested the list of other language links, thus creating a multilingual resource. Both the dry run language (Mandarin) and evaluation language (Uyghur) have their own Wikipedias, so this gazetteer found a number of hits in each case. In the case that a language is so low-resource that it has poorer coverage, one may consider using less restrictive matching criteria so that named entities may match pages from more widely used languages in the same language family as the target language.

The second gazetteer was extracted by simply asking the native informant to create it. Specifically, we started with the set of prototypes from the Wikipedia gazetteer, separated them by type, and had them translate each into their language's writing system. We also asked the native informant to produce 5-10 additional names for each entity type. For both dry run and evaluation the informants were very capable of this task and easily translated the vast majority of entities. The task of generating new entities of the same type was more difficult, and the informants were more productive if they were given subtypes (e.g. for the PER category we gave them mostly politicians but then suggested they could also be actors, business people, athletes, etc.).

TABLE I
CHECKPOINT SYSTEM PERFORMANCE USING THE *strong_typed_mention_match* METRIC REPORTED BY THE SUBMISSION INTERFACE.

System	CP1 Score	CP2 Score	CP3 Score
Raw	0.022		
NI	0.113		
Nolex		0.184	0.137
Words	0.184	0.178	0.168
Words+Syn	0.147	0.173	0.156

Given that the native informant-generated gazetteer was a superset of the Wikipedia gazetteer, for all other experiments we ignored the Wikipedia gazetteer and used the informant-generated gazetteer as the starting point for bootstrapping. We compared two bootstrapping systems for checkpoint one, a baseline which used only the default features, and a test system which used features extracted from the UHHMM system parses of set E in addition to the default features.

For checkpoints two and three, we made minor changes to the system, mainly in the way we preprocessed the data as described above. In addition to our preprocessing changes, which attempt to measure whether convergence is an issue, we introduced a new feature configuration to test whether overfitting might be an issue. One explanation is that syntactic features might be redundant with word identity features. To test this, we submitted one additional run where we used syntax features but no word identity features.

Checkpoints Two and Three

- No lex id feats (Nolex) - Used the bootstrapping approach described above, with initial labeling done by native informant gazetteer and supervised machine learning done with syntactic features, and all features from the LDC-distributed tagger *except* word identity features.
- Bootstrapping CRF with baseline features (same as above)
- Primary (same as above)

IV. Discussion and Conclusion

While the challenge was useful and exciting, and while we believe our system was learning useful syntactic information, we were nonetheless disappointed that our syntactic features did not benefit the task of named entity extraction. We have three main points of discussion around these results that point to future research directions. The first point of interest is that the grammar induction system used here parses sentences with no recursion. While such a system can correctly parse many short sentences, with more complex sentences it reverts to being a chunker. There is some chance that the output of what is essentially "supertags" does not provide meaningfully more information than lexical items. While we had no true supervised data, set E was actually quite large and perhaps lexical sparsity was not a major issue, making POS-tag-like features redundant. This suggests that allowing even some minimal level of recursion (bounded) may provide better higher-level features that capture different levels of information than words and POS tags.

Second, we note that even with the severe restrictions of sentence length and lexical complexity, the corpus used to train the grammar induction system was still quite large and the system did not converge even after 150,000 iterations. Perhaps a more fully converged system would extract better features for named entity tagging. This points us in the direction of exploring curriculum learning strategies to improve system convergence. In curriculum learning, examples are presented in an order that most benefits the learning system, typically easiest first. In our problem setting, what exact form that learning takes is an empirical question. Perhaps simpler lexical items should be presented first, but maybe shorter sentences are easier as well. Future work should explore different dimensions of complexity and how they affect learning in the UHHMM system.

Finally, it is possible that named entity extraction as a task is not amenable to syntactic features. We have evaluated our system internally using part of speech evaluations and parsing evaluations with positive results. Named entity extraction is a straightforward task so it would be wonderful if it was amenable to features we output, but if not we will continue to search for downstream tasks that can make use of syntactic features, perhaps in collaboration with teams more focused on the downstream tasks.

References

[Abney, 2002] Abney, S. (2002). Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 360–367, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

[Garrette and Baldridge, 2013] Garrette, D. and Baldridge, J. (2013). Learning a part-of-speech tagger from two hours of annotation. In *HLT-NAACL*, pages 138–147. Citeseer.

[Johnson, 2007] Johnson, M. (2007). Why doesn't EM find good HMM POS-taggers. *Proceedings of the 2007 Joint Conference on* ..., (June):296–305.

[Kozareva, 2006] Kozareva, Z. (2006). Bootstrapping named entity recognition with automatically generated gazetteer lists. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, EACL '06, pages 15–21, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Okazaki, 2007] Okazaki, N. (2007). Crfsuite: a fast implementation of conditional random fields (crfs).

[Onyshkevych, 2014] Onyshkevych, B. (2014). Low resource languages for emergent incidents (LORELEI). Broad Agency Announcement DARPA-BAA-15-04, Defense Advanced Research Projects Agency (DARPA).

[Putthividhya and Hu, 2011] Putthividhya, D. P. and Hu, J. (2011). Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1557–1567, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Strzalkowski and Wang, 1996] Strzalkowski, T. and Wang, J. (1996). A self-learning universal concept spotter. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 931–936, Stroudsburg, PA, USA. Association for Computational Linguistics.

[van Gael et al., 2009] van Gael, J., Vlachos, A., and Ghahramani, Z. (2009). The infinite HMM for unsupervised PoS tagging. (August):678–687.

[van Schijndel et al., 2013] van Schijndel, M., Exley, A., and Schuler, W. (2013). A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196.