Prototype-Driven Learning for Sequence Models

Aria Haghighi
Computer Science Division
University of California Berkeley
aria42@cs.berkeley.edu

Dan Klein
Computer Science Division
University of California Berkeley
klein@cs.berkeley.edu

Abstract

We investigate *prototype-driven* learning for primarily unsupervised sequence modeling. Prior knowledge is specified declaratively, by providing a few canonical examples of each target annotation label. This sparse prototype information is then propagated across a corpus using distributional similarity features in a log-linear generative model. On part-of-speech induction in English and Chinese, as well as an information extraction task, prototype features provide substantial error rate reductions over competitive baselines and outperform previous work. For example, we can achieve an English part-of-speech tagging accuracy of 80.5% using only three examples of each tag and no dictionary constraints. We also compare to semi-supervised learning and discuss the system's error trends.

1 Introduction

Learning, broadly taken, involves choosing a good model from a large space of possible models. In supervised learning, model behavior is primarily determined by labeled examples, whose production requires a certain kind of expertise and, typically, a substantial commitment of resources. In unsupervised learning, model behavior is largely determined by the structure of the model. Designing models to exhibit a certain target behavior requires another, rare kind of expertise and effort. Unsupervised learning, while minimizing the usage of labeled data, does not necessarily minimize total effort. We therefore consider here how to learn models with the least effort. In particular, we argue for a certain kind of semi-supervised learning, which we call *prototype-driven* learning.

In prototype-driven learning, we specify prototypical examples for each target label or label configuration, but do not necessarily label any documents or sentences. For example, when learning a model for Penn treebank-style part-of-speech tagging in English, we may list the 45 target tags and a few examples of each tag (see figure 4 for a concrete prototype list for this task). This manner of specifying prior knowledge about the task has several advantages. First, is it certainly compact (though it remains to be proven that it is effective). Second, it is more or less the minimum one would have to provide to a human annotator in order to specify a new annotation task and policy (compare, for example, with the list in figure 2, which suggests an entirely different task). Indeed, prototype lists have been used pedagogically to summarize tagsets to students (Manning and Schütze, 1999). Finally, natural language does exhibit proform and prototype effects (Radford, 1988), which suggests that learning by analogy to prototypes may be effective for language tasks.

In this paper, we consider three sequence modeling tasks: part-of-speech tagging in English and Chinese and a classified ads information extraction task. Our general approach is to use distributional similarity to link any given word to similar prototypes. For example, the word *reported* may be linked to *said*, which is in turn a prototype for the part-of-speech VBD. We then encode these prototype links as features in a log-linear generative model, which is trained to fit unlabeled data (see section 4.1). Distributional prototype features provide substantial error rate reductions on all three tasks. For example, on English part-of-speech tagging with three prototypes per tag, adding prototype features to the baseline raises per-position accuracy from 41.3% to 80.5%.

2 Tasks and Related Work: Tagging

For our part-of-speech tagging experiments, we used data from the English and Chinese Penn treebanks (Marcus et al., 1994; Ircs, 2002). Example sentences

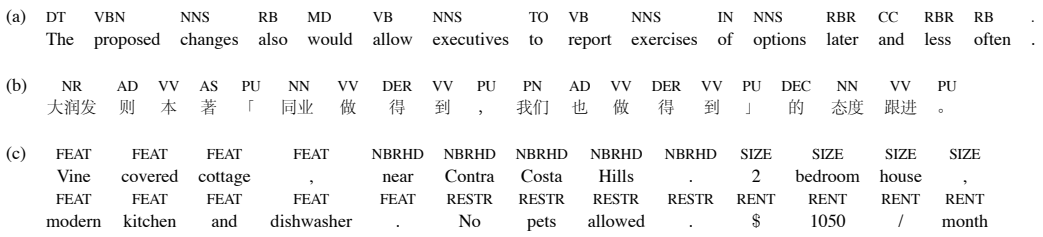

Figure 1: Sequence tasks: (a) English POS, (b) Chinese POS, and (c) Classified ad segmentation

are shown in figure 1(a) and (b). A great deal of research has investigated the unsupervised and semi-supervised induction of part-of-speech models, especially in English, and there is unfortunately only space to mention some highly related work here.

One approach to unsupervised learning of part-of-speech models is to induce HMMs from unlabeled data in a maximum-likelihood framework. For example, Merialdo (1991) presents experiments learning HMMs using EM. Merialdo's results most famously show that re-estimation degrades accuracy unless almost no examples are labeled. Less famously, his results also demonstrate that re-estimation can improve tagging accuracies to some degree in the fully unsupervised case.

One recent and much more successful approach to part-of-speech learning is *contrastive estimation*, presented in Smith and Eisner (2005). They utilize task-specific comparison neighborhoods for part-of-speech tagging to alter their objective function.

Both of these works require specification of the legal tags for each word. Such dictionaries are large and embody a great deal of lexical knowledge. A prototype list, in contrast, is extremely compact.

3 Tasks and Related Work: Extraction

Grenager et al. (2005) presents an unsupervised approach to an information extraction task, called CLASSIFIEDS here, which involves segmenting classified advertisements into topical sections (see figure 1(c)). Labels in this domain tend to be "sticky" in that the correct annotation tends to consist of multi-element fields of the same label. The overall approach of Grenager et al. (2005) typifies the process involved in fully unsupervised learning on new domain: they first alter the structure of their HMM so that diagonal transitions are preferred, then modify the transition structure to explicitly model boundary tokens, and so on. Given enough refinements the model learns to segment with a reasonable match to the target structure.

Label	Prototypes
ROOMATES	`roommate respectful drama`
RESTRICTIONS	`pets smoking dog`
UTILITIES	`utilities pays electricity`
AVAILABLE	`immediately begin cheaper`
SIZE	`2 br sq`
PHOTOS	`pictures image link`
RENT	`$ month *number*15*1`
CONTACT	`*phone* call *time*`
FEATURES	`kitchen laundry parking`
NEIGHBORHOOD	`close near shopping`
ADDRESS	`address carlmont *ordinal*5`
BOUNDARY	`; . !`

Figure 2: Prototype list derived from the development set of the CLASSIFIEDS data. The BOUNDARY field is not present in the original annotation, but added to model boundaries (see Section 5.3). The starred tokens are the results of collapsing of basic entities during pre-processing as is done in (Grenager et al., 2005)

In section 5.3, we discuss an approach to this task which does not require customization of model structure, but rather centers on feature engineering.

4 Approach

In the present work, we consider the problem of learning sequence models over text. For each document $x = [x_i]$, we would like to predict a sequence of labels $y = [y_i]$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We construct a generative model, $p(x, y|\theta)$, where θ are the model's parameters, and choose parameters to maximize the log-likelihood of our observed data D:

$$\begin{aligned} \mathcal{L}(\theta; D) &= \sum_{x \in D} \log p(x|\theta) \\ &= \sum_{x \in D} \log \sum_{y} p(x, y|\theta) \end{aligned}$$

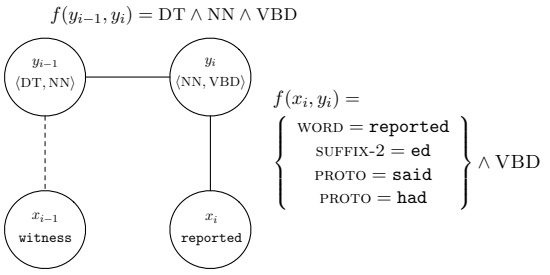

Figure 3: Graphical model representation of trigram tagger for English POS domain.

4.1 Markov Random Fields

We take our model family to be chain-structured Markov random fields (MRFs), the undirected equivalent of HMMs. Our joint probability model over (x, y) is given by

$$p(x, y|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \phi(x_i, y_i)\phi(y_{i-1}, y_i)$$

where $\phi(c)$ is a potential over a clique c, taking the form $\exp\left\{\theta^T f(c)\right\}$, and $f(c)$ is the vector of features active over c. In our sequence models, the cliques are over the edges/transitions (y_{i-1}, y_i) and nodes/emissions (x_i, y_i). See figure 3 for an example from the English POS tagging domain.

Note that the only way an MRF differs from a conditional random field (CRF) (Lafferty et al., 2001) is that the partition function is no longer observation dependent; we are modeling the joint probability of x and y instead of y given x. As a result, learning an MRF is slightly harder than learning a CRF; we discuss this issue in section 4.4.

4.2 Prototype-Driven Learning

We assume prior knowledge about the target structure via a *prototype list*, which specifies the set of target labels $\mathcal{Y}$ and, for each label $y \in \mathcal{Y}$, a set of prototypes words, $p_y \in \mathcal{P}_y$. See figures 2 and 4 for examples of prototype lists.[1]

[1]Note that this setting differs from the standard semi-supervised learning setup, where a small number of fully labeled examples are given and used in conjunction with a larger amount of unlabeled data. In our prototype-driven approach, we never provide a single *fully* labeled example sequence. See section 5.3 for further comparison of this setting to semi-supervised learning.

Broadly, we would like to learn sequence models which both explain the observed data and meet our prior expectations about target structure. A straightforward way to implement this is to constrain each prototype word to take only its given label(s) at training time. As we show in section 5, this does not work well in practice because this constraint on the model is very sparse.

In providing a prototype, however, we generally mean something stronger than a constraint on that word. In particular, we may intend that words which are in some sense similar to a prototype generally be given the same label(s) as that prototype.

4.3 Distributional Similarity

In syntactic distributional clustering, words are grouped on the basis of the vectors of their preceeding and following words (Schütze, 1995; Clark, 2001). The underlying linguistic idea is that replacing a word with another word of the same syntactic category should preserve syntactic well-formedness (Radford, 1988). We present more details in section 5, but for now assume that a similarity function over word types is given.

Suppose further that for each non-prototype word type w, we have a subset of prototypes, S_w, which are known to be distributionally similar to w (above some threshold). We would like our model to relate the tags of w to those of S_w.

One approach to enforcing the distributional assumption in a sequence model is by supplementing the training objective (here, data likelihood) with a penalty term that encourages parameters for which each w's posterior distribution over tags is compatible with it's prototypes S_w. For example, we might maximize,

$$\sum_{x \in D} \log p(x|\theta) - \sum_{w} \sum_{z \in S_w} KL(\, t|z \,||\, t|w)$$

where $t|w$ is the model's distribution of tags for word w. The disadvantage of a penalty-based approach is that it is difficult to construct the penalty term in a way which produces exactly the desired behavior.

Instead, we introduce distributional prototypes into the learning process as features in our log-linear model. Concretely, for each prototype z, we introduce a predicate PROTO $= z$ which becomes active

at each w for which $z \in S_w$ (see figure 3). One advantage of this approach is that it allows the strength of the distributional constraint to be calibrated along with any other features; it was also more successful in our experiments.

4.4 Parameter Estimation

So far we have ignored the issue of how we learn model parameters θ which maximize $\mathcal{L}(\theta; D)$. If our model family were HMMs, we could use the EM algorithm to perform a local search. Since we have a log-linear formulation, we instead use a gradient-based search. In particular, we use L-BFGS (Liu and Nocedal, 1989), a standard numerical optimization technique, which requires the ability to evaluate $\mathcal{L}(\theta; D)$ and its gradient at a given θ.

The density $p(x|\theta)$ is easily calculated up to the global constant $Z(\theta)$ using the forward-backward algorithm (Rabiner, 1989). The partition function is given by

$$\begin{aligned} Z(\theta) &= \sum_x \sum_y \prod_{i=1}^{n} \phi(x_i, y_i)\phi(y_{i-1}, y_i) \\ &= \sum_x \sum_y \mathit{score}(x, y) \end{aligned}$$

$Z(\theta)$ can be computed exactly under certain assumptions about the clique potentials, but can in all cases be bounded by

$$\hat{Z}(\theta) = \sum_{\ell=1}^{K} \hat{Z}_\ell(\theta) = \sum_{\ell=1}^{K} \sum_{x:|x|=\ell} \mathit{score}(x, y)$$

Where K is a suitably chosen large constant. We can efficiently compute $\hat{Z}_\ell(\theta)$ for fixed ℓ using a generalization of the forward-backward algorithm to the lattice of all observations x of length ℓ (see Smith and Eisner (2005) for an exposition).

Similar to supervised maximum entropy problems, the partial derivative of $\mathcal{L}(\theta; D)$ with respect to each parameter θ_j (associated with feature f_j) is given by a difference in feature expectations:

$$\frac{\partial \mathcal{L}(\theta; D)}{\partial \theta_j} = \sum_{x \in D} \left(\mathbb{E}_{y|x,\theta} f_j - \mathbb{E}_{x,y|\theta} f_j \right)$$

The first expectation is the expected count of the feature under the model's $p(y|x, \theta)$ and is again easily computed with the forward-backward algorithm, just as for CRFs or HMMs. The second expectation is the expectation of the feature under the model's joint distribution over all x, y pairs, and is harder to calculate. Again assuming that sentences beyond a certain length have negligible mass, we calculate the expectation of the feature for each fixed length ℓ and take a (truncated) weighted sum:

$$\mathbb{E}_{x,y|\theta} f_j = \sum_{\ell=1}^{K} p(|x| = \ell) \mathbb{E}_{x,y|\ell,\theta} f_j$$

For fixed ℓ, we can calculate $\mathbb{E}_{x,y|\ell,\theta} f_j$ using the lattice of all inputs of length ℓ. The quantity $p(|x| = \ell)$ is simply $\hat{Z}_\ell(\theta)/\hat{Z}(\theta)$.

As regularization, we use a diagonal Gaussian prior with variance $\sigma^2 = 0.5$, which gave relatively good performance on all tasks.

	Num Tokens	
Setting	48K	193K
BASE	42.2	41.3
PROTO	61.9	68.8
PROTO+SIM	79.1	80.5

Table 1: English POS results measured by per-position accuracy

5 Experiments

We experimented with prototype-driven learning in three domains: English and Chinese part-of-speech tagging and classified advertisement field segmentation. At inference time, we used maximum posterior decoding,[2] which we found to be uniformly but slightly superior to Viterbi decoding.

5.1 English POS Tagging

For our English part-of-speech tagging experiments, we used the WSJ portion of the English Penn treebank (Marcus et al., 1994). We took our data to be either the first 48K tokens (2000 sentences) or 193K tokens (8000 sentences) starting from section 2. We used a trigram tagger of the model form outlined in section 4.1 with the same set of spelling features reported in Smith and Eisner (2005): exact word type,

[2] At each position choosing the label which has the highest posterior probability, obtained from the forward-backward algorithm.

Label	Prototype	Label	Prototype
NN	% company year	NNS	years shares companies
JJ	new other last	VBG	including being according
MD	will would could	-LRB-	-LRB- -LCB-
VBP	are 're 've	DT	the a The
RB	n't also not	WP$	whose
-RRB-	-RRB- -RCB-	FW	bono del kanji
WRB	when how where	RP	Up ON
IN	of in for	VBD	said was had
SYM	c b f	$	$ US$ C$
CD	million billion two	#	#
TO	to To na	:	– : ;
VBN	been based compared	NNPS	Philippines Angels Rights
RBR	Earlier duller	“	“ ‘ non-“
VBZ	is has says	VB	be take provide
JJS	least largest biggest	RBS	Worst
NNP	Mr. U.S. Corp.	,	,
POS	'S	CC	and or But
PRP$	its their his	JJR	smaller greater larger
PDT	Quite	WP	who what What
WDT	which Whatever whatever	.	. ? !
EX	There	PRP	it he they
”	”	UH	Oh Well Yeah

Figure 4: English POS prototype list

Correct Tag	Predicted Tag	% of Errors
CD	DT	6.2
NN	JJ	5.3
JJ	NN	5.2
VBD	VBN	3.3
NNS	NN	3.2

Figure 5: Most common English POS confusions for PROTO+SIM on 193K tokens

character suffixes of length up to 3, *initial-capital*, *contains-hyphen*, and *contains-digit*. Our only edge features were tag trigrams.

With just these features (our baseline BASE) the problem is symmetric in the 45 model labels. In order to break initial symmetry we initialized our potentials to be near one, with some random noise. To evaluate in this setting, model labels must be mapped to target labels. We followed the common approach in the literature, greedily mapping each model label to a target label in order to maximize per-position accuracy on the dataset. The results of BASE, reported in table 1, depend upon random initialization; averaging over 10 runs gave an average per-position accuracy of 41.3% on the larger training set.

We automatically extracted the prototype list by taking our data and selecting for each annotated label the top three occurring word types which were not given another label more often. This resulted in 116 prototypes for the 193K token setting.[3] For comparison, there are 18,423 word types occurring in this data.

Incorporating the prototype list in the simplest possible way, we fixed prototype occurrences in the data to their respective annotation labels. In this case, the model is no longer symmetric, and we no longer require random initialization or post-hoc mapping of labels. Adding prototypes in this way gave an accuracy of 68.8% on all tokens, but only 47.7% on non-prototype occurrences, which is only a marginal improvement over BASE. It appears as though the prototype information is not spreading to non-prototype words.

In order to remedy this, we incorporated distributional similarity features. Similar to (Schütze, 1995), we collect for each word type a context vector of the counts of the most frequent 500 words, conjoined with a direction and distance (e.g +1,-2). We then performed an SVD on the matrix to obtain a reduced rank approximation. We used the dot product between left singular vectors as a measure of distributional similarity. For each word w, we find the set of prototype words with similarity exceeding a fixed threshold of 0.35. For each of these prototypes z, we add a predicate PROTO $= z$ to each occurrence of w. For example, we might add PROTO $=$ *said* to each token of *reported* (as in figure 3).[4]

Each prototype word is also its own prototype (since a word has maximum similarity to itself), so when we lock the prototype to a label, we are also pushing all the words distributionally similar to that prototype towards that label.[5]

[3]To be clear: this method of constructing a prototype list required statistics from the labeled data. However, we believe it to be a fair and necessary approach for several reasons. First, we wanted our results to be repeatable. Second, we did not want to overly tune this list, though experiments below suggest that tuning could greatly reduce the error rate. Finally, it allowed us to run on Chinese, where the authors have no expertise.

[4]Details of distributional similarity features: To extract context vectors, we used a window of size 2 in either direction and use the first 250 singular vectors. We collected counts from all the WSJ portion of the Penn Treebank as well as the entire BLIPP corpus. We limited each word to have similarity features for its top 5 most similar prototypes.

[5]Note that the presence of a prototype feature does not ensure every instance of that word type will be given its prototype's label; pressure from "edge" features or other prototype features can cause occurrences of a word type to be given different labels. However, rare words with a single prototype feature are almost always given that prototype's label.

This setting, PROTO+SIM, brings the all-tokens accuracy up to 80.5%, which is a 37.5% error reduction over PROTO. For non-prototypes, the accuracy increases to 67.8%, an error reduction of 38.4% over PROTO. The overall error reduction from BASE to PROTO+SIM on all-token accuracy is 66.7%.

Table 5 lists the most common confusions for PROTO+SIM. The second, third, and fourth most common confusions are characteristic of fully supervised taggers (though greater in number here) and are difficult. For instance, both JJs and NNs tend to occur after determiners and before nouns. The CD and DT confusion is a result of our prototype list not containing a *contains-digit* prototype for CD, so the predicate fails to be linked to CDs. Of course in a realistic, iterative design setting, we could have altered the prototype list to include a *contains-digit* prototype for CD and corrected this confusion.

Figure 6 shows the marginal posterior distribution over label pairs (roughly, the bigram transition matrix) according to the treebank labels and the PROTO+SIM model run over the training set (using a collapsed tag set for space). Note that the broad structure is recovered to a reasonable degree.

It is difficult to compare our results to other systems which utilize a full or partial tagging dictionary, since the amount of provided knowledge is substantially different. The best comparison is to Smith and Eisner (2005) who use a partial tagging dictionary. In order to compare with their results, we projected the tagset to the coarser set of 17 that they used in their experiments. On 24K tokens, our PROTO+SIM model scored 82.2%. When Smith and Eisner (2005) limit their tagging dictionary to words which occur at least twice, their best performing neighborhood model achieves 79.5%. While these numbers seem close, for comparison, their tagging dictionary contained information about the allowable tags for 2,125 word types (out of 5,406 types) and the their system must only choose, on average, between 4.4 tags for a word. Our prototype list, however, contains information about only 116 word types and our tagger must on average choose between 16.9 tags, a much harder task. When Smith and Eisner (2005) include tagging dictionary entries for all words in the first half of their 24K tokens, giving tagging knowledge for 3,362 word types, they do achieve a higher accuracy of 88.1%.

Setting	Accuracy
BASE	46.4
PROTO	53.7
PROTO+SIM	71.5
PROTO+SIM+BOUND	74.1

Figure 7: Results on test set for ads data in (Grenager et al., 2005).

5.2 Chinese POS Tagging

We also tested our POS induction system on the Chinese POS data in the Chinese Treebank (Ircs, 2002). The model is wholly unmodified from the English version except that the suffix features are removed since, in Chinese, suffixes are not a reliable indicator of part-of-speech as in English (Tseng et al., 2005). Since we did not have access to a large auxiliary unlabeled corpus that was segmented, our distributional model was built only from the treebank text, and the distributional similarities are presumably degraded relative to the English. On 60K word tokens, BASE gave an accuracy of 34.4, PROTO gave 39.0, and PROTO+SIM gave 57.4, similar in order if not magnitude to the English case.

We believe the performance for Chinese POS tagging is not as high as English for two reasons: the general difficulty of Chinese POS tagging (Tseng et al., 2005) and the lack of a larger segmented corpus from which to build distributional models. Nonetheless, the addition of distributional similarity features does reduce the error rate by 35% from BASE.

5.3 Information Field Segmentation

We tested our framework on the CLASSIFIEDS data described in Grenager et al. (2005) under conditions similar to POS tagging. An important characteristic of this domain (see figure 1(a)) is that the hidden labels tend to be "sticky," in that fields tend to consist of runs of the same label, as in figure 1(c), in contrast with part-of-speech tagging, where we rarely see adjacent tokens given the same label. Grenager et al. (2005) report that in order to learn this "sticky" structure, they had to alter the structure of their HMM so that a fixed mass is placed on each diagonal transition. In this work, we learned this structure automatically though prototype similarity features without manually constraining the model (see

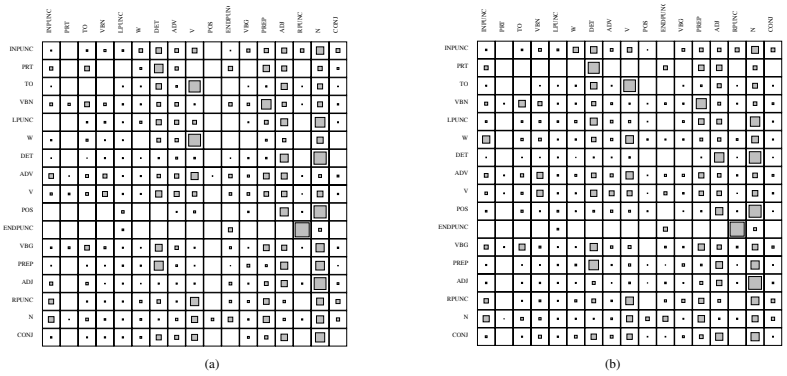

Figure 6: English coarse POS tag structure: a) corresponds to "correct" transition structure from labeled data, b) corresponds to PROTO+SIM on 24K tokens

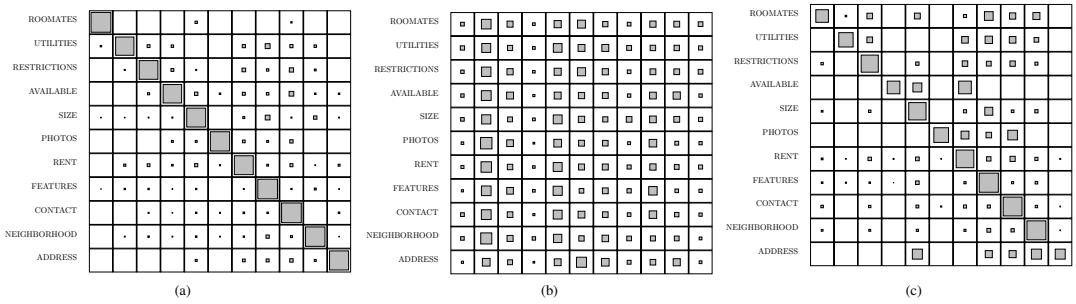

Figure 8: Field segmentation observed transition structure: (a) labeled data, (b) BASE(c) BASE+PROTO+SIM+BOUND (after post-processing)

figure 8), though we did change the similarity function (see below).

On the test set of (Grenager et al., 2005), BASE scored an accuracy of 46.4%, comparable to Grenager et al. (2005)'s unsupervised HMM baseline. Adding the prototype list (see figure 2) without distributional features yielded a slightly improved accuracy of 53.7%. For this domain, we utilized a slightly different notion of distributional similarity: we are not interested in the syntactic behavior of a word type, but its topical content. Therefore, when we collect context vectors for word types in this domain, we make no distinction by direction or distance and collect counts from a wider window. This notion of distributional similarity is more similar to latent semantic indexing (Deerwester et al., 1990). A natural consequence of this definition of distributional similarity is that many neighboring words will share the same prototypes. Therefore distributional prototype features will encourage labels to persist, naturally giving the "sticky" effect of the domain. Adding distributional similarity features to our model (PROTO+SIM) improves accuracy substantially, yielding 71.5%, a 38.4% error reduction over BASE.[6]

Another feature of this domain that Grenager et al. (2005) take advantage of is that end of sentence punctuation tends to indicate the end of a field and the beginning of a new one. Grenager et al. (2005) experiment with manually adding *boundary* states and biasing transitions from these states to not self-loop. We capture this "boundary" effect by simply adding a line to our protoype-list, adding a new BOUNDARY state (see figure 2) with a few (hand-chosen) prototypes. Since we utilize a trigram tagger, we are able to naturally capture the effect that the BOUNDARY tokens typically indicate transitions between the fields before and after the boundary token. As a post-processing step, when a token is tagged as a BOUNDARY

[6]Distributional similarity details: We collect for each word a context vector consisting of the counts for words occurring within three token occurrences of a word. We perform a SVD onto the first 50 singular vectors.

Correct Tag	Predicted Tag	% of Errors
FEATURES	SIZE	11.2
FEATURES	NBRHD	9.0
SIZE	FEATURES	7.7
NBRHD	FEATURES	6.4
ADDRESS	NBRHD	5.3
UTILITIES	FEATURES	5.3

Figure 9: Most common classified ads confusions

token it is given the same label as the previous non-BOUNDARY token, which reflects the annotational convention that boundary tokens are given the same label as the field they terminate. Adding the BOUNDARY label yields significant improvements, as indicated by the PROTO+SIM+BOUND setting in Table 5.3, surpassing the best unsupervised result of Grenager et al. (2005) which is 72.4%. Furthermore, our PROTO+SIM+BOUND model comes close to the *supervised* HMM accuracy of 74.4% reported in Grenager et al. (2005).

We also compared our method to the most basic semi-supervised setting, where fully labeled documents are provided along with unlabeled ones. Roughly 25% of the data had to be labeled in order to achieve an accuracy equal to our PROTO+SIM+BOUND model, suggesting that the use of prior knowledge in the prototype system is particularly efficient.

In table 5.3, we provide the top confusions made by our PROTO+SIM+BOUND model. As can be seen, many of our confusions involve the FEATURE field, which serves as a general purpose background state, which often differs subtly from other fields such as SIZE. For instance, the parenthical comment: *(master has walk - in closet with vanity)* is labeled as a SIZE field in the data, but our model proposed it as a FEATURE field. NEIGHBORHOOD and ADDRESS is another natural confusion resulting from the fact that the two fields share much of the same vocabulary (e.g [ADDRESS *2525 Telegraph Ave.*] vs. [NBRHD *near Telegraph*]).

Acknowledgments We would like to thank the anonymous reviewers for their comments. This work is supported by a Microsoft / CITRIS grant and by an equipment donation from Intel.

6 Conclusions

We have shown that distributional prototype features can allow one to specify a target labeling scheme in a compact and declarative way. These features give substantial error reduction on several induction tasks by allowing one to link words to prototypes according to distributional similarity. Another positive property of this approach is that it tries to reconcile the success of sequence-free distributional methods in unsupervised word clustering with the success of sequence models in supervised settings: the similarity guides the learning of the sequence model.

References

Alexander Clark. 2001. The unsupervised induction of stochastic context-free grammars using distributional clustering. In *CoNLL*.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Trond Grenager, Dan Klein, and Christopher Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the 43rd Meeting of the ACL*.

Nianwen Xue Ircs. 2002. Building a large-scale annotated chinese corpus.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Bernard Merialdo. 1991. Tagging english text with a probabilistic model. In *ICASSP*, pages 809–812.

L.R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE*.

Andrew Radford. 1988. *Transformational Grammar*. Cambridge University Press, Cambridge.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL*.

Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Meeting of the ACL*.

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help pos tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.