Appears in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001).*

Representing Sentence Structure in Hidden Markov Models for Information Extraction

Soumya Ray[*†]
sray@cs.wisc.edu

Mark Craven[†*]
craven@biostat.wisc.edu

[*]Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706

[†]Department of Biostatistics & Medical Informatics
University of Wisconsin
Madison, Wisconsin 53706

Abstract

We study the application of Hidden Markov Models (HMMs) to learning information extractors for n-ary relations from free text. We propose an approach to representing the grammatical structure of sentences in the states of the model. We also investigate using an objective function during HMM training which maximizes the ability of the learned models to identify the phrases of interest. We evaluate our methods by deriving extractors for two binary relations in biomedical domains. Our experiments indicate that our approach learns more accurate models than several baseline approaches.

1 Introduction

Information extraction (IE) may be defined as the task of automatically extracting instances of specified classes or relations from text. In our research, we are interested in using machine learning approaches, including hidden Markov models (HMMs), to extract certain relationships among objects from biomedical text sources. We present and evaluate two contributions to the state of the art in learning information extractors with HMMs. First, we investigate an approach to incorporating information about the grammatical structure of sentences into HMM architectures. Second, we investigate an objective function for HMM training whose emphasis is on maximizing the ability of the learned models to identify the phrases of interest rather than simply maximizing the likelihood of the training data. Our experiments in two challenging real world domains indicate that both contributions lead to more accurate learned models.

Automated methods for information extraction have several valuable applications including populating knowledge bases and databases, summarizing collections of documents, and identifying significant but unknown relationships among objects. Since constructing information extraction systems manually has proven to be expensive[Riloff, 1996], there has been much recent interest in using machine learning methods to learn information extraction models from labeled training data. Hidden Markov models are among the more successful approaches considered for learning information extractors [Leek, 1997; Freitag and McCallum, 1999; Seymore *et al.*, 1999; Freitag and McCallum, 2000; McCallum *et al.*, 2000].

Previous HMM approaches to information extraction do not adequately address several key aspects of the problem domains on which we are focused. First, the data we are processing is complex natural language text. Whereas previous approaches have represented their data as sequences of tokens, we present an approach in which sentences are first processed by a shallow parser and then represented as sequences of typed phrases. Second, the data we are processing include many sentences that are not relevant to the relations of interest. Even in relevant sentences, only certain phrases contain information to be extracted. Whereas previous approaches to applying HMMs for IE have focused the training process on maximizing the likelihood of the training sentences, we adopt a training method that is designed to maximize the probability of assigning the correct labels to various parts of the sentences being processed [Krogh, 1994]. Our approach involves coupling the algorithm devised by Krogh with the use of *null models* which are intended to represent data not directly relevant to the task at hand.

2 Problem Domain

Our work is focused on extracting instances of specific relations of interest from abstracts in the MEDLINE database [National Library of Medicine, 2001]. MEDLINE contains bibliographic information and abstracts from more than 4,000 biomedical journals.

An example of a binary relation that we consider in our experiments is the subcellular-localization relation, which represents the location of a particular protein within a cell. We refer to the *domains* of this relation as `PROTEIN` and `LOCATION`. We refer to an instance of a relation as a *tuple*. Figure 1 provides an illustration of our extraction task. The top of the figure shows two sentences in a MEDLINE abstract. The bottom of the figure shows the instance of the target relation subcellular-localization that we would like to extract from the second sentence. This tuple asserts that the protein UBC6 is found in the subcellular compartment called the endoplasmic reticulum.

In order to learn models to perform this task, training examples consisting of passages of text, annotated with the tuples that should be extracted from them, are needed. In our

". . .
Here we report the identification of an integral membrane ubiquitin-conjugating enzyme. This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.
. . ."

$\Downarrow$

subcellular-localization(UBC6,endoplasmic reticulum)

Figure 1: An example of the information extraction task. The top shows part of a document from which we wish to extract instances of the subcellular-localization relation. The bottom shows the extracted tuple.

approach, each training and test instance is an individual sentence. There are several aspects of the data that make this a difficult information extraction task: (i) it involves free text, (ii) the genre of text is, in general, not grammatically simple, (iii) the text includes a lot of technical terminology, (iv) there are many sentences from which nothing should be extracted.

In the terminology that has been used in the information extraction literature, our task is inherently a *multiple slot* extraction task. Since we are interested in extracting instances of n-ary relations, we cannot treat each domain of such a relation as a separate unary component to be extracted (also called *single slot* extraction). Consider the subcellular-localization relation discussed above. A document may mention many proteins and many locations but this relation holds only among certain pairs of these proteins and locations.

In the experiments reported here, we use two data sets representing two different binary relations. The subcellular-localization data set includes 545 sentences that represent positive instances (labeled with tuples that should be extracted from them) and 6,700 sentences that represent negative instances (not labeled with any tuples). The 545 positive instances are labeled with 645 tuples in all; there are 335 unique tuples. The second data set is for a binary relation that characterizes associations between genes and genetic disorders. We refer to this relation as disorder-association, and the domains of this relation as GENE and DISORDER. This data set contains 892 positive instances and 11,487 negative instances. The positive instances are labeled with 899 tuples in all (126 unique). For both data sets, the negative instances are "near misses" in that they come from the same population of abstracts as the positive instances, and in many cases they discuss concepts that are associated with the target relation.

The target tuples for the subcellular-localization relation were collected from the Yeast Protein Database (YPD) [Hodges *et al.*, 1998], and the target tuples for the disorder-association relation were collected from the Online Mendelian Inheritance in Man (OMIM) database [Center for Medical Genetics , 2001]. Relevant MEDLINE abstracts were also gathered from entries in these databases. To label the sentences in these abstracts, we matched the target tuples to the words in the sentence. A sentence which contained words that matched a tuple was taken to be a positive instance. Every other sentence was considered to be a negative instance. It is clear that while this process is automatic, it will result in a noisy labeling. A sentence may have the words in a target tuple of the relation while the semantics may not refer to the relation. On the other hand, a tuple in a relation may be described by synonymous words which were not in any target tuple; therefore, sentences where tuples exist as synonyms are labeled incorrectly. We used a random sample of 200 positive and 200 negative sentences to estimate the amount of noise introduced by the labeling process. We estimate with 95% confidence that approximately 10% to 15% of the sentences are labeled incorrectly (either falsely labeled or unlabeled when they should have been) in the subcellular-localization data set. We believe that the disorder-association data set is not as noisy as the subcellular-localization data set.

3 Representing Phrase Structure

Hidden Markov Models (HMMs) are the stochastic analogs of finite state automata. An HMM is defined by a set of states and a set of transitions between them. Each state has an associated emission distribution which defines the likelihood of a state to emit various tokens. The transitions from a given state have an associated transition distribution which defines the likelihood of the next state given the current state.

In previous HMM approaches to information extraction, sentences have been represented as sequences of tokens. We hypothesize that incorporating sentence structure into the models we build results in better extraction accuracy.

Our approach is based on using syntactic parses of all sentences we process. In particular, we use the Sundance system [Riloff, 1998] to obtain a shallow parse of each given sentence. Our representation does not incorporate all of the information provided by a Sundance parse, but instead "flattens" it into a sequence of phrase segments. Each phrase segment consists of a *type* describing the grammatical nature of the phrase, and the words that are part of the phrase.

In positive training examples, if a segment contains a word or words that belong to a domain in a target tuple, it is annotated with the corresponding domain. We refer to these annotations as *labels*. Labels are absent in the test instances. Figure 2a shows a sentence containing an instance of the subcellular-localization relation and its annotated segments (we shall discuss the other panels of this figure later). The second phrase segment in this example is a noun phrase segment (NP_SEGMENT) that contains the protein name UBC6 (hence the PROTEIN label). Note that the types are constants that are pre-defined by our representation of Sundance parses, while the labels are defined with respect to the domains of relation we are trying to extract. Also note that the parsing is not always accurate, for instance, the third segment in figure 2a should really be a VP_SEGMENT, but has been typed as an NP_SEGMENT by Sundance.

The states in our HMMs represent the annotated segments of a sentence. Like a segment, each state in the model is annotated with a $\langle$*type*,*label*$\rangle$ pair[1]. A given state can emit only segments whose type is identical to the state's type; for

[1]We can think of segments that do not have a label corresponding a domain of the relation as having an implicit *empty* label.

“This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.”

NP_SEGMENT	this enzyme
NP_SEGMENT:PROTEIN	ubc6
NP_SEGMENT	localizes
PP_SEGMENT	to
NP_SEGMENT:LOCATION	the endoplasmic reticulum
PP_SEGMENT	with
NP_SEGMENT	the catalytic domain
VP_SEGMENT	facing
NP_SEGMENT	the cytosol
	(a)

DET	this
UNK	enzyme
UNK:PROTEIN	ubc6
UNK	localizes
PREP	to
ART	the
N:LOCATION	endoplasmic
UNK:LOCATION	reticulum
PREP	with
ART	the
N	catalytic
UNK	domain
V	facing
ART	the
N	cytosol
	(b)

	this
	enzyme
PROTEIN	ubc6
	localizes
	to
	the
LOCATION	endoplasmic
LOCATION	reticulum
	with
	the
	catalytic
	domain
	facing
	the
	cytosol
	(c)

Figure 2: HMM input representations. (a) The *Phrase* representation: the sentence is segmented into typed phrases. (b) The *POS* representation: the sentence is segmented into words typed with part-of-speech tags. (c) The *Token* representation: the sentence is segmented into untyped words. For each representation, the labels (`PROTEIN`, `LOCATION`) are only present in the training sentences.

example, the segment “`this enzyme`” in figure 2a could be emitted by any state with type `NP_SEGMENT`, regardless of its label. Each state that has a label corresponding to a domain of the relation plays a direct role in extracting tuples.

Figure 3 is a schematic of the architecture of our phrase-based hidden Markov models. The top of the figure shows the *positive model*, which is trained to represent positive instances in the training set. The bottom of the figure shows the *null model*, which is trained to represent negative instances in the training set. Since our *Phrase* representation includes 14 phrase types, both models have 14 states without labels, and the positive model also has five to six additional labeled states (one for each ⟨*type*,*label*⟩ combination that occurs in the training set). We assume a fully connected model, that is, the model may emit a segment of any type at any position within a sentence.

To train and test our *Phrase* models, we have to modify the standard Forward, Backward and Viterbi algorithms [Rabiner, 1989]. The Forward algorithm calculates the probability $\alpha_q(i)$ of a sentence being in state q of the model after having emitted i elements of an instance. When a sentence is represented as a sequence of tokens, the algorithm is based on the following recurrence:

$$\begin{aligned} \alpha_{START}(0) &= 1 \\ \alpha_q(0) &= 0, q \neq START \\ \alpha_q(i) &= M(w_i|q) \sum_r T(q|r)\alpha_r(i-1) \end{aligned} \quad (1)$$

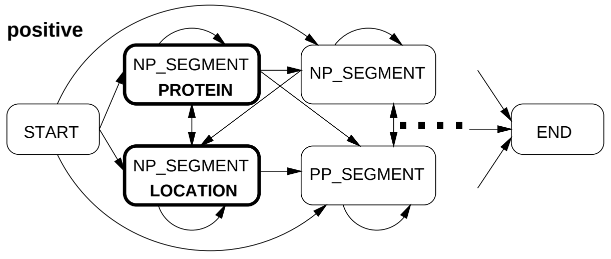
where M and T represent the emission and transition distributions respectively, w_i is the i^{th} element in the instance, and r ranges over the states that transition to q.

Our modification involves changing the last part of this recurrence as follows:

$$\alpha_q(i) = \begin{cases} \left(\prod_{j=1}^{|p_i|} M(w_j|q)\right) \sum_r T(q|r)\alpha_r(i-1), \\ \qquad \text{if } \mathtt{type}(q) = \mathtt{type}(p_i); \\ 0, \qquad \text{otherwise.} \end{cases} \quad (2)$$

Here w_j is the j^{th} word in the i^{th} phrase segment p_i, and `type` is a function that returns the *type* of a segment or state as described above. The two key aspects of this modification are that (i) the type of a segment has to agree with the type of state in order for the state to emit it, and (ii) the emission probability of the words in the segment is computed as the product of the emission probabilities of the individual words. This latter aspect is analogous to having states use a Naïve Bayes model for the words in a phrase. Note that this equation requires a normalization factor to define a proper distribution over sentences. However, since we use these equations to make relative comparisons only, we leave this factor implicit. The modifications to the Viterbi and Backward algorithms are similar to this modification of the Forward algorithm.

Given these modifications to the Forward and Backward algorithm, we could train phrase-based models using the Baum-Welch algorithm [Baum, 1972]. However, for the models we consider here, there is no hidden state for training examples (i.e., there is an unambiguous path through the model for each example), and thus there is no need to use Baum-Welch. Instead, we assume a fully connected model and obtain transition frequencies by considering how often segments with various ⟨*type*, *label*⟩ annotations are adjacent to each other. We smooth these frequencies over the set of possible transitions for every state using m-estimates [Cestnik, 1990]. In a similar manner, we obtain the emission frequencies of the words in each state by summing over all segments with the



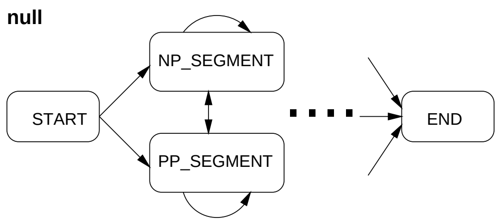


Figure 3: The general architecture of our phrase-based HMMs. The top part of the figure shows the *positive model* and the bottom part of the figure shows the *null model*.

same ⟨*type*, *label*⟩ annotations in our training set. We smooth these frequency counts using m-estimates over the entire vocabulary of words.

Once the model has been constructed, we use it to predict tuples in test sentences. We use the Viterbi algorithm, modified as described above, to determine the most likely path of a sentence through the positive model. We consider a sentence to represent a tuple of the target relation if and only if two conditions hold:

1. The likelihood of emission of the sentence by the positive model is greater than the likelihood of emission by the null model: $\alpha_{P_{END}}(n) > \alpha_{N_{END}}(n)$, where P and N refer to the positive and null models respectively and the sentence has n segments.

2. In the Viterbi path for the positive model, there are segments aligned with states corresponding to *all* the domains of the relation. For example, for the subcellular-localization relation, the Viterbi path for a sentence must pass through a state with the PROTEIN label and a state with the LOCATION label.

Note that even after phrases have been identified in this way, the extraction task is not quite complete, since some of the phrases might contain words other than those that belong in an extracted tuple. Consider the example in Figure 2a. The LOCATION phrase contains the word "the" in addition to the location. Therefore, tuple extraction with these models must include a post-processing phase in which such extraneous words are stripped away before tuples are returned. We do not address this issue here. Instead, we consider a prediction to be correct if the model correctly identifies the phrases containing the target tuple as a subphrase.

It is possible to have multiple predicted segments for each domain of the relation. In this case, we must decide which combinations of segments constitute tuples. We do this using two simple rules:

1. Associate segments in the order in which they occur. Thus for subcellular-localization, the first segment matching a PROTEIN state is associated with the first segment matching a LOCATION state, and so on.

2. If there are fewer segments containing an element of some domain, use the last match of this domain to construct the remaining tuples. For instance, if we predicted one PROTEIN phrase P_1 and two LOCATION phrases L_1 and L_2, we would create two tuples based on $\langle P_1, L_1\rangle$ and $\langle P_1, L_2\rangle$.

3.1 Experiments

In the experiments presented in this section, we test our hypothesis that incorporating phrase-level sentence structure into our model provides improved extraction performance in terms of precision and recall. We test this hypothesis by comparing against several hidden Markov models that represent less information about the grammatical structure of sentences. Henceforth, we refer to the model described above as the *Phrase Model*.

The first model we compare against, which we call the *POS Model*, is based on the representation shown in Figure 2b. This model represents some grammatical information, in that it associates a type with each token indicating the part-of-speech(POS) tag for the word (as determined by Sundance). However, unlike the *Phrase Model*, the *POS model* represents sentences as sequences of tokens, not phrases. This model is comparable in size to the *Phrase Model*. The positive component of this model has 17 states without labels and six to ten states with labels (depending on the training set). The null component of the model has 17 states without labels.

The other models we consider, which we call the *Token Models*, are based on the representation shown in Figure 2c. This representation treats a sentence simply as a sequence of words. We investigate two variants that employ this representation. The simpler of the two hidden Markov models based on this representation, which we refer to as *Token Model 1*, has three states in its positive model and one state in its null model (not counting the START and END states). None of the states in this model have types. Two of the states in the positive model represent the domains of the binary target relation, while the remaining states have no labels. The role of the latter set of states is to model all tokens that do not correspond to the domains of the target relation. A more complex version of this model, which is illustrated in Figure 4, has three unlabeled states in its positive model. We define the transitions and train these models in such a way that these three states can specialize to (i) tokens that come before any relation instances, (ii) tokens that are interspersed between the domains of relation instances, and (iii) tokens that come after relation instances.

The training algorithm used for the *POS Model* is identical to that used for the *Phrase Model*. The training algorithm for the *Token Models* is essentially the same, except that there are no type constraints on either the tokens or states.

Since we consider a prediction made by the *Phrase Model* to be correct if it simply identifies the phrases containing the words of the tuple, we use a similar criterion to decide if the predictions made by the *POS Model* and *Token Models* are

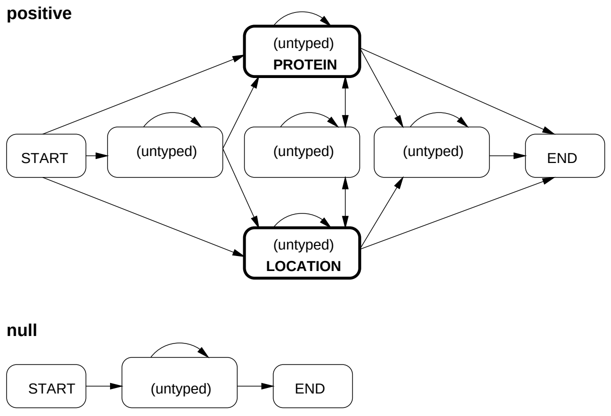


Figure 4: The architecture of *Token Model 2*.

correct. We consider *POS Model* and *Token Model* predictions to be correct if the labeled states of these models identify sequences of tokens that contain the words of the tuple. These models are not penalized for extracting extra adjacent words along with the actual words of a target tuple.

We process the HMM input data (after parsing in cases where Sundance is used) by stemming words with the Porter algorithm [Porter, 1980], and replacing words that occur only once in a training set with a generic `UNKNOWN` token. The statistics for this token are then used by the model while emitting out-of-vocabulary words encountered during prediction. Similarly, numbers are mapped to a generic `NUMBER` token.

Positive predictions are ranked by a confidence measure which is computed as the ratio of the likelihood of the Viterbi path of a sentence through a model to the likelihood of the model to emit that sentence, i.e. confidence$(s) = \delta_{END}(n)/\alpha_{END}(n)$. Here s is a sentence of n segments, $\delta_{END}(n)$ is the likelihood of the most probable path of all n segments threaded through to the `END` state, and $\alpha_{END}(n)$ is the comparable value calculated by the Forward algorithm. We construct precision-recall graphs for our models by varying a threshold on the confidence measures.

For both data sets we measure precision and recall using 5-fold cross validation. The data is partitioned such that all of the sentences from a given MEDLINE abstract are in the same fold. This procedure ensures that our experiments model the nature of the real application setting. For training, we sample the negative instances so that there are an equal number of positive and negative instances per fold. We have observed that we get better recall consistently by doing this.

Figure 5 shows the precision-recall curves for the subcellular-localization data set. The curve for the *Phrase Model* is superior to the curves for both *Token Models*. At low levels of recall, the *POS Model* exhibits slightly higher precision than the *Phrase Model*, but the latter is superior at higher recall levels, and the *Phrase Model* has a significantly higher recall endpoint. These results suggest that there is value in representing grammatical structure in the HMM architectures, but the *Phrase Model* is not definitively more accurate.

Figure 6 shows the precision-recall curves for the

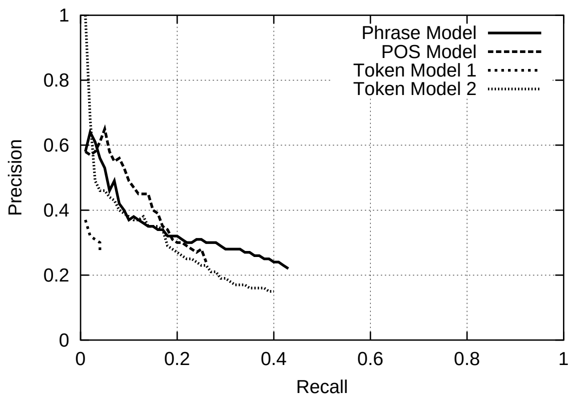


Figure 5: Precision vs. recall for the four models on the subcellular-localization data set.

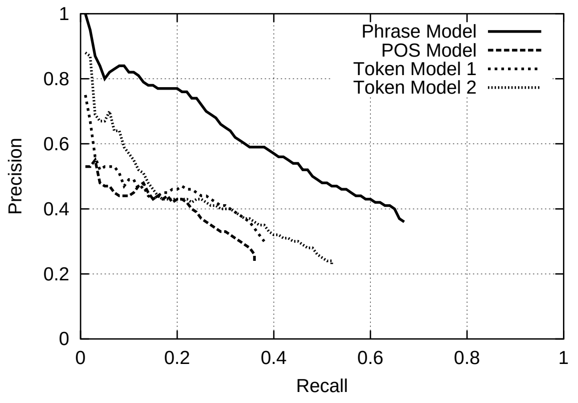


Figure 6: Precision vs. recall for the four models on the disorder-association data set.

disorder-association data set. Here, the differences are much more pronounced. The *Phrase Model* achieves significantly higher levels of precision than any of the other models, including the *POS Model*. The recall endpoint for the *Phrase Model* is also superior to those of the other models. We conclude that the experiments presented here support our hypothesis that incorporating sentence structure into the models we build results in better extraction accuracy.

4 Improving Parameter Estimates

Standard HMM training algorithms, like Baum-Welch, are designed to maximize the likelihood of the data given the model. Specifically, if s_i is a sentence in the training set, Baum-Welch (and the method we used earlier) tries to find parameters $\hat{\theta}$ such that

$$\hat{\theta} = \arg\max_{\theta} \prod_i \Pr(s_i|\theta).$$

We hypothesize that more accurate models can be learned by training with an objective function that aims to maximize the likelihood of predicting the correct sequence of *labels* for a given sentence (as before, we assume that states and phrases

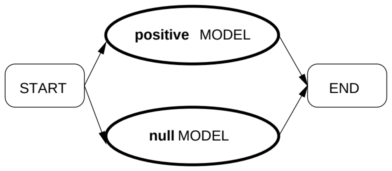


Figure 7: Combined model architecture. The *positive* and *null* models refer to the corresponding models in Figure 3.

without labels have an implicit *empty* label). Let c_i be the known sequence of labels for a sentence s_i in our training set. We would like to estimate parameters $\hat{\theta}$ such that

$$\hat{\theta} = \arg\max_{\theta} \prod_i \Pr(c_i | s_i, \theta) \quad (3)$$

$$= \arg\max_{\theta} \prod_i \frac{\Pr(c_i, s_i | \theta)}{\Pr(s_i | \theta)}. \quad (4)$$

This is similar to the task of optimizing parameters to recover the sequence of states given a set of observations[McCallum *et al.*, 2000]. Krogh[1994] has devised an HMM training algorithm that tries to optimize this criterion. After transforming this objective function into one which aims to minimize the negative log likelihood of the above equation, the following incremental update rule is obtained:

$$\theta_j^{new} = N(\theta_j^{old} + \eta(m_j^i - n_j^i)) \quad (5)$$

where θ_j is the j^{th} parameter, m_j^i is the expected number of times θ_j is used by the i^{th} sentence on *correct* paths through the model, n_j^i is the expected number of times θ_j is used by the i^{th} sentence on *all* paths through the model, N is a normalizing constant, and η is the learning rate. The n and m terms can be calculated using the Forward-Backward procedure. Note that the update rule represents an online training procedure.

In our previous experiments, we used a separate null model to represent negative instances. We would like to use Krogh's algorithm with this configuration to observe if it results in more accurate models. However, the null model as we have described it is a separate entity which is trained separately. With this architecture, Krogh's algorithm would be unable to correct false positives in the training set since doing so might require adjusting the parameters of the positive model in response to a negative instance. To remedy this problem, we propose an alternative to having a separate null model, which we refer to as a *Combined Model*. A *Combined Model* consists of two submodels sharing common START and END states. A schematic is shown in figure 7. The shared START and END states allow the training algorithm to update parameters in both parts of the model in response to a given training sentence.

4.1 Experiments

To evaluate this algorithm, we train the *Combined Model* configuration on the subcellular-localization and the disorder-association data sets. We compare these models against the *Phrase Model* trained on the corresponding data sets in our previous experiments.

The methodology for this experiment is the same as before. Note that for the *Combined Model*, prediction is simpler than with a separate null model, since it suffices to consider the Viterbi path of a sentence through the model to extract tuples, if any. We do not train the *Combined Model* to convergence to avoid overfitting. Instead, we set the number of iterations for which to do gradient descent to a fixed constant value of 100.

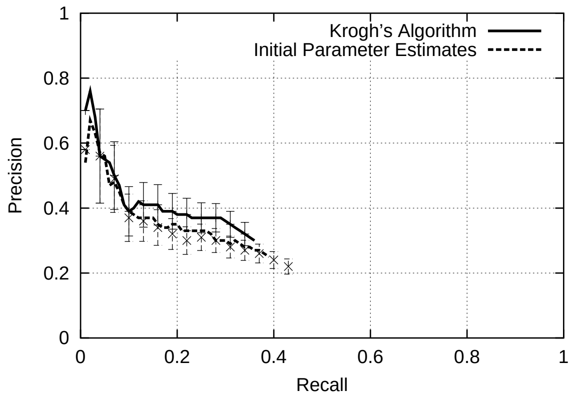


Figure 8: Effect of Krogh's Algorithm on the combined model for the subcellular-localization data set.

Figure 8 shows the precision-recall curves for this experiment for the subcellular-localization data set. For each precision-recall curve, we also show 95% confidence intervals. From the figure, we observe that there is some improvement in the precision of the model on this data set, while recall is held nearly constant. While the improvement is small, we have observed it consistently across the various model architectures we have explored. Figure 9 shows the corresponding precision-recall curves and confidence intervals for the experiment on the disorder-association data set. Here, the difference between the initial model and the trained model is more pronounced. The model trained with Krogh's algo-

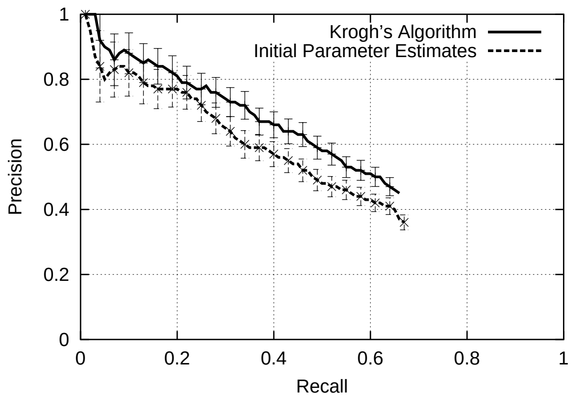


Figure 9: Effect of Krogh's Algorithm on the combined model for the disorder-association data set.

rithm has significantly better precision than the initial model, while maintaining a similar level of recall. We conclude that this training algorithm is appropriate for our task, and can improve accuracy, sometimes significantly.

5 Conclusion

We have presented two contributions to learning Hidden Markov Models for information extraction, and evaluated these contributions on two challenging biomedical domains. We have presented an approach to representing the grammatical structure of sentences in an HMM. Comparative experiments with other models lacking such information shows that this approach learns extractors that have increased precision and recall performance. We have also investigated the application of a training algorithm developed by Krogh to our models. This algorithm consistently provides an accuracy gain over our original models. We believe that these are promising approaches to the task of deriving information extractors for free text domains.

Acknowledgments

This research was supported in part by NIH Grant 1R01 LM07050-01, and NSF CAREER award IIS-0093016. The authors would like to thank Michael Waddell for his work on building the disorder-association data set, and Peter Andreae, Joseph Bockhorst, Tina Eliassi-Rad, and Jude Shavlik for critiquing the initial draft.

References

[Baum, 1972] L. E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[Center for Medical Genetics , 2001] Center for Medical Genetics, Johns Hopkins University and National Center for Biotechnology Information. Online Mendelian inheritance in man, OMIM (TM), 2001. http://www.ncbi.nlm.nih.gov/omim/.

[Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–150, Stockholm, Sweden, 1990. Pitman.

[Freitag and McCallum, 1999] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *Working Notes of the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999. AAAI Press.

[Freitag and McCallum, 2000] D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI Press.

[Hodges *et al.*, 1998] P. E. Hodges, W. E. Payne, and J. I. Garrels. Yeast protein database (YPD): A database for the complete proteome of saccharomyces cerevisiae. *Nucleic Acids Research*, 26:68–72, 1998.

[Krogh, 1994] A. Krogh. Hidden Markov models for labeled sequences. In *Proceedings of the Twelfth International Conference on Pattern Recognition*, pages 140–144, Jerusalem, Israel, 1994. IEEE Computer Society Press.

[Leek, 1997] T. Leek. Information extraction using hidden Markov models. Master's thesis, Department of Computer Science and Engineering, University of California, San Diego, CA, 1997.

[McCallum *et al.*, 2000] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, Stanford, CA, 2000. Morgan Kaufmann.

[National Library of Medicine, 2001] National Library of Medicine. The MEDLINE database, 2001. http://www.ncbi.nlm.nih.gov/PubMed/.

[Porter, 1980] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):127–130, 1980.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Riloff, 1996] E. Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, 85:101–134, 1996.

[Riloff, 1998] E. Riloff. The Sundance sentence analyzer, 1998. http://www.cs.utah.edu/projects/nlp/.

[Seymore *et al.*, 1999] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *Working Notes of the AAAI Workshop on Machine Learning for Information Extraction*, pages 37–42. AAAI Press, 1999.