In Proceedings of HLT/EMNLP, 2005.

Composition of Conditional Random Fields for Transfer Learning

Charles Sutton and Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{sutton,mccallum}@cs.umass.edu

Abstract

Many learning tasks have subtasks for which much training data exists. Therefore, we want to *transfer* learning from the old, general-purpose subtask to a more specific new task, for which there is often less data. While work in transfer learning often considers how the old task should affect learning on the new task, in this paper we show that it helps to take into account how the new task affects the old. Specifically, we perform joint decoding of separately-trained sequence models, preserving uncertainty between the tasks and allowing information from the new task to affect predictions on the old task. On two standard text data sets, we show that joint decoding outperforms cascaded decoding.

1 Introduction

Many tasks in natural language processing are solved by chaining errorful subtasks. Within information extraction, for example, part-of-speech tagging and shallow parsing are often performed before the main extraction task. Commonly these subtasks have their own standard sets of labeled training data: for example, many large data sets exist for learning to extract person names from newswire text; whereas the available training data for new applications, such as extracting appointment information from email, tends to be much smaller. Thus, we need to transfer regularities learned from a well-studied subtask, such as finding person names in newswire text, to a new, related task, such as finding names of speakers in email seminar announcements.

In previous NLP systems, transfer is often accomplished by training a model for the subtask, and using its prediction as a feature for the new task. For example, recent CoNLL shared tasks (Tjong Kim Sang & De Meulder, 2003; Carreras & Marquez, 2004), which are standard data sets for such common NLP tasks as clause identification and named-entity recognition, include predictions from a part-of-phrase tagger and a shallow parser as features. But including only the single most likely subtask prediction fails to exploit useful dependencies between the tasks. First, if the subtask prediction is wrong, the model for the new task may not be able to recover. Often, errors propagate upward through the chain of tasks, causing errors in the final output. This problem can be ameliorated by preserving uncertainty in the subtask predictions, because even if the best subtask prediction is wrong, the distribution over predictions can still be somewhat accurate.

Second, information from the main task can inform the subtask. This is especially important for learning transfer, because the new domain often has different characteristics than the old domain, which is often a standard benchmark data set. For example, named-entity recognizers are usually trained on newswire text, which is more structured and grammatical than email, so we expect an off-the-shelf named-entity recognizer to perform somewhat worse on email. An email task, however, often has domain-specific features, such as PREVIOUS WORD IS *Speaker:*, which were unavailable or uninformative to the subtask on the old training set, but are very informative to the subtask in the new domain. While previous work in transfer learning has considered how the old task can help the new task, in this paper we show how the new task can help itself by improving predictions on the old.

In this paper we address the issue of transfer by training a cascade of models independently on the various training sets, but at test time combining them into a single model in which decoding is performed jointly. For the individual models, we use linear-chain conditional random fields (CRFs), because the great freedom that they allow in feature engineering facilitates the learning of richer interactions between the subtasks. We train a linear chain CRF on each subtask, using the prediction of the previous subtask as a feature. At test time, we combine the learned weights from the original CRFs into a single grid-shaped factorial CRF, which makes predictions for all the tasks at once. Viterbi decoding in this combined model implicitly considers all possible predictions for the subtask when making decisions in the main task.

We evaluate joint decoding for learning transfer on a standard email data set and a standard entity recognition task. On the email data set, we show a significant gain in performance, including new state-of-the-art results. Of particular interest for transfer learning, we also show that using joint decoding, we achieve equivalent results to cascaded decoding with 25% less training data.

2 Linear-chain CRFs

Conditional random fields (CRFs) (Lafferty et al., 2001) are undirected graphical models that are conditionally trained. In this section, we describe CRFs for the linear-chain case. Linear-chain CRFs can be roughly understood as conditionally-trained finite state machines. A linear-chain CRF defines a distribution over state sequences $\mathbf{s} = \{s_1, s_2, \ldots, s_T\}$ given an input sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ by making a first-order Markov assumption on states. These Markov assumptions imply that the distribution over sequences factorizes in terms of pairwise functions $\Phi_t(s_{t-1}, s_t, \mathbf{x})$ as:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \Phi_t(s_{t-1}, s_t, \mathbf{x})}{Z(\mathbf{x})}, \tag{1}$$

The partition function $Z(\mathbf{x})$ is defined to ensure that the distribution is normalized:

$$Z(\mathbf{x}) = \sum_{\mathbf{s}'} \prod_t \Phi_t(s'_{t-1}, s'_t, \mathbf{x}). \tag{2}$$

The *potential functions* $\Phi_t(s_{t-1}, s_t, \mathbf{x})$ can be interpreted as the cost of making a transition from state s_{t-1} to state s_t at time t, similar to a transition probability in an HMM.

Computing the partition function $Z(\mathbf{x})$ requires summing over all of the exponentially many possible state sequences $\mathbf{s}'$. By exploiting Markov assumptions, however, $Z(\mathbf{x})$ (as well as the node marginals $p(s_t|\mathbf{x})$ and the Viterbi labeling) can be calculated efficiently by variants of the standard dynamic programming algorithms used for HMMs.

We assume the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that

$$\Phi(s_{t-1}, s_t, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{x}, t)\right). \tag{3}$$

The model parameters are a set of real weights $\Lambda = \{\lambda_k\}$, one for each feature.

Feature functions can be arbitrary. For example, one feature function could be a binary test $f_k(s_{t-1}, s_t, \mathbf{x}, t)$ that has value 1 if and only if s_{t-1} has the label SPEAKERNAME, s_t has the label OTHER, and the word x_t begins with a capital letter. The chief practical advantage of conditional models, in fact, is that we can include arbitrary highly-dependent features without needing to estimate their distribution, as would be required to learn a generative model.

Given fully-labeled training instances $\{(\mathbf{s}_j, \mathbf{x}_j)\}_{j=1}^{M}$, CRF training is usually performed by maximizing the penalized log likelihood

$$\ell(\Lambda) = \sum_j \sum_t \sum_k \lambda_k f_k(s_{j,t-1}, s_{j,t}, \mathbf{x}, t) - \sum_j \log Z(\mathbf{x}_j) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \tag{4}$$

where the final term is a zero-mean Gaussian prior placed on parameters to avoid overfitting. Although this maximization cannot be done in closed form, it can be optimized numerically. Particularly effective are gradient-based methods that use approximate second-order information, such as conjugate gradient and limited-memory BFGS (Byrd et al., 1994). For more information on current training methods for CRFs, see Sha and Pereira (2003).

3 Dynamic CRFs

Dynamic conditional random fields (Sutton et al., 2004) extend linear-chain CRFs in the same way that dynamic Bayes nets (Dean & Kanazawa, 1989) extend HMMs. Rather than having a single monolithic state variable, DCRFs factorize the state at each time step by an undirected model.

Formally, DCRFs are the class of conditionally-trained undirected models that repeat structure and parameters over a sequence. If we denote by $\Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t)$ the repetition of clique c at time step t, then a DCRF defines the probability of a label sequence $\mathbf{s}$ given the input $\mathbf{x}$ as:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t)}{Z(\mathbf{x})}, \tag{5}$$

where as before, the clique templates are parameterized in terms of input features as

$$\Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t) = \exp\left\{\sum_k \lambda_k f_k(\mathbf{y}_{c,t}, \mathbf{x}_t)\right\}. \tag{6}$$

Exact inference in DCRFs can be performed by forward-backward in the cross product state space, if the cross-product space is not so large as to be infeasible. Otherwise, approximate methods must be used; in our experience, loopy belief propagation is often effective in grid-shaped DCRFs. Even if inference is performed monolithically, however, a factorized state representation is still useful because it requires much fewer parameters than a fully-parameterized linear chain in the cross-product state space.

Sutton et al. (2004) introduced the *factorial CRF* (FCRF), in which the factorized state structure is a grid (Figure 1). FCRFs were originally applied to jointly performing interdependent language processing tasks, in particular part-of-speech tagging and noun-phrase chunking. The previous work on FCRFs used joint training, which requires a single training set that is jointly labeled for all tasks in the cascade. For many tasks such data is not readily available, for example, labeling syntactic parse trees for every new Web extraction task would be prohibitively expensive. In this paper, we train the subtasks separately, which allows us the freedom to use large, standard data sets for well-studied subtasks such as named-entity recognition.

4 Alternatives for Learning Transfer

In this section, we enumerate several classes of methods for learning transfer, based on the amount and type of interaction they allow between the tasks. The principal differences between methods are whether the individual tasks are performed separately in a cascade or jointly; whether a single prediction from the lower task is used, or several; and what kind of confidence information is shared between the subtasks.

The main types of transfer learning methods are:

1. *Cascaded training and testing.* This is the traditional approach in NLP, in which the single best prediction from the old task is used in the new task at training and test time. In this paper, we show that allowing richer interactions between the subtasks can benefit performance.

2. *Joint training and testing.* In this family of approaches, a single model is trained to perform all the subtasks at once. For example, in Caruana's work on multitask learning (Caruana, 1997), a neural network is trained to jointly perform multiple classification tasks, with hidden nodes that form a shared representation among the tasks. Jointly trained methods allow potentially the richest interaction between tasks, but can be expensive in both computation time required for training and in human effort required to label the joint training data.

   Exact inference in a jointly-trained model, such as forward-backward in an FCRF, implicitly considers all possible subtask predictions with confidence given by the model's probability of the prediction. However, for computational efficiency, we can use inference methods such as particle filtering and sparse message-passing (Pal et al., 2005), which communicate only a limited number of predictions between sections of the model.

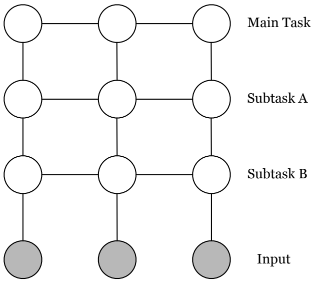

Figure 1: Graphical model for the jointly-decoded CRF. All of the pairwise cliques also have links to the observed input, although we omit these edges in the diagram for clarity.

3. *Joint testing with cascaded training.* Although a joint model over all the subtasks can have better performance, it is often much more expensive to train. One approach for reducing training time is cascaded training, which provides both computational efficiency and the ability to reuse large, standard training sets for the subtasks. At test time, though, the separately-trained models are combined into a single model, so that joint decoding can propagate information between the tasks.

   Even with cascaded training, it is possible to preserve some uncertainty in the subtask's predictions. Instead of using only a single subtask prediction for training the main task, the subtask can pass upwards a lattice of likely predictions, each of which is weighted by the model's confidence. This has the advantage of making the training procedure more similar to the joint testing procedure, in which all possible subtask predictions are considered.

In the next two sections, we describe and evaluate joint testing with cascaded training for transfer learning in linear-chain CRFs. At training time, only the best subtask prediction is used, without any confidence information. Even though this is perhaps the simplest joint-testing/cascaded-training method, we show that it still leads to a significant gain in accuracy.

5 Composition of CRFs

In this section we briefly describe how we combine individually-trained linear-chain CRFs using composition. For a series of N cascaded tasks, we train individual CRFs separately on each task, using the prediction of the previous CRF as a feature. We index the CRFs by i, so that the state of CRF i at time t is denoted s_t^i. Thus, the feature functions for CRF i are of the form $f_k^i(s_{t-1}^i, s_t^i, s_t^{i-1}, \mathbf{x}, t)$—that is, they depend not only on the observed input $\mathbf{x}$ and the transition $(s_{t-1}^i \rightarrow s_t^i)$ but

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t appears in list of first names, last names, honorifics, etc.
w_t appears to be part of a time followed by a dash
w_t appears to be part of a time preceded by a dash
w_t appears to be part of a date
$T_t = T$
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-4, 4]$

Table 1: Input features $q_k(\mathbf{x}, t)$ for the seminars data. In the above w_t is the word at position t, T_t is the POS tag at position t, w ranges over all words in the training data, and T ranges over all Penn Treebank part-of-speech tags. The "appears to be" features are based on hand-designed regular expressions that can span several tokens.

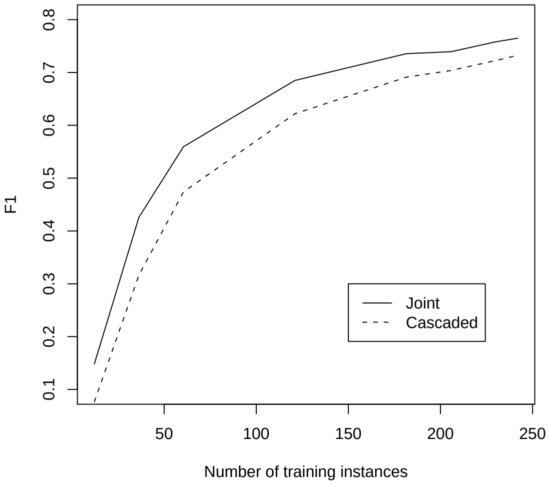

Figure 2: Learning curves for the seminars data set on the speaker field, averaged over 10-fold cross validation. Joint training performs equivalently to cascaded decoding with 25% more data.

also on the state s_t^{i-1} of the previous transducer.

We also add all conjunctions of the input features and the previous transducer's state, for example, a feature that is 1 if the current state is SPEAKERNAME, the previous transducer predicted PERSONNAME, and the previous word is `Host:`.

To perform joint decoding at test time, we form the composition of the individual CRFs, viewed as finite-state transducers. That is, we define a new linear-chain CRF whose state space is the cross product of the states of the individual CRFs, and whose transition costs are the sum of the transition costs of the individual CRFs.

Formally, let $S^1, S^2, \dots S^N$ be the state sets and $\Lambda^1, \Lambda^2, \dots \Lambda^N$ the weights of the individual CRFs. Then the state set of the combined CRF is $\mathbf{S} = S^1 \times S^2 \times \dots \times S^N$. We will denote weight k in an individual CRF i by λ_k^i and a single feature by $f_k^i(s_{t-1}^i, s_t^i, s_t^{i-1}, \mathbf{x}, t)$. Then for $\mathbf{s} \in \mathbf{S}$, the combined model is given by:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \exp\left\{\sum_{i=1}^N \sum_k \lambda_k^i f_k^i(s_{t-1}^i, s_t^i, s_t^{i-1}, \mathbf{x}, t)\right\}}{Z(\mathbf{x})}. \tag{7}$$

The graphical model for the combined model is the factorial CRF in Figure 1.

6 Experiments

6.1 Email Seminar Announcements

We evaluate joint decoding on a collection of 485 e-mail messages announcing seminars at Carnegie Mellon University, gathered by Freitag (1998). The messages are annotated with the seminar's starting time, ending time, location, and speaker. This data set has been the subject of much previous work using a wide variety of learning methods. Despite all this work, however, the best reported systems have precision and recall on speaker names of only about 70%—too low to use in a practical system. This task is so challenging because the messages are written by many different people, who each have different ways of presenting the announcement information.

Because the task includes finding locations and person names, the output of a named-entity tagger is a useful feature. It is not a perfectly indicative feature, however, because many other kinds of person names appear in seminar announcements—for example, names of faculty hosts, departmental secretaries, and sponsors of lecture series. For example, the token *Host:* indicates strongly both that what follows is a person name, but that person is not the seminars' speaker.

Even so, named-entity predictions do improve performance on this task. We use the predictions from a CRF named-entity tagger that we trained on the standard CoNLL 2003 English data set. The CoNLL 2003 data set consists of newswire articles from Reuters labeled as either people, locations, organizations, or miscellaneous entities. It is much larger than the seminar announcements data set. While the named-entity data contains 203,621 tokens for training, the seminar announcements data set contains only slightly over 60,000 training tokens.

Previous work on the seminars data has used a one-field-per-document evaluation. That is, for each field, the CRF selects a single field value from its Viterbi path, and this extraction is counted as correct if it exactly matches any of the true field mentions in the document. We compute precision and recall following this convention, and report their harmonic mean F_1. As in the previous work,

System		stime	etime	location	speaker	overall
WHISK	(Soderland, 1999)	92.6	86.1	66.6	18.3	65.9
SRV	(Freitag, 1998)	98.5	77.9	72.7	56.3	76.4
HMM	(Frietag & McCallum, 1999)	98.5	62.1	78.6	76.6	78.9
RAPIER	(Califf & Mooney, 1999)	95.9	94.6	73.4	53.1	79.3
SNOW-IE	(Roth & Wen-tau Yih, 2001)	**99.6**	96.3	75.2	73.8	86.2
(LP)^2	(Ciravegna, 2001)	99.0	95.5	75.0	**77.6**	86.8
CRF (no transfer)	This paper	99.1	**97.3**	81.0	73.7	87.8
CRF (cascaded)	This paper	99.2	96.0	84.3	74.2	88.4
CRF (joint)	This paper	99.1	96.0	**85.3**	76.3	**89.2**

Table 2: Comparison of F_1 performance on the seminars data. Joint decoding performs significantly better than cascaded decoding. The overall column is the mean of the other four. (This table was adapted from Peshkin and Pfeffer (2003).)

we use 10-fold cross validation with a 50/50 training/test split. We use a spherical Gaussian prior on parameters with variance $\sigma^2 = 0.5$.

We evaluate whether joint decoding with cascaded training performs better than cascaded training and decoding. Table 2 compares cascaded and joint decoding for CRFs with other previous results from the literature.[1] The features we use are listed in Table 1. Although previous work has used very different feature sets, we include a no-transfer CRF baseline to assess the impact of transfer from the CoNLL data set. All the CRF runs used exactly the same features.

On the most challenging fields, location and speaker, cascaded transfer is more accurate than no transfer at all, and joint decoding is more accurate than cascaded decoding. In particular, for speaker, we see an error reduction of 8% by using joint decoding over cascaded. The difference in F1 between cascaded and joint decoding is statistically significant for speaker (paired t-test; p = 0.017) but only marginally significant for location (p = 0.067). Our results are competitive with previous work; for example, on location, the CRF is more accurate than any of the existing systems.

Examining the trained models, we can observe both errors made by the general-purpose named entity tagger, and how they can be corrected by considering the seminars labels. In newswire text, long runs of capitalized words are rare, often indicating the name of an entity. In email announcements, runs of capitalized words are common in formatted text blocks like:

```
Location:  Baker Hall
    Host:  Michael Erdmann
```

In this type of situation, the named entity tagger often mistakes *Host:* for the name of an entity, especially because the word preceding *Host* is also capitalized. On one of the cross-validated testing sets, of 80 occurrences of the word *Host:*, the named-entity tagger labels 52 as some kind of entity. When joint decoding is used, however, only 20 occurrences are labeled as entities. Recall that the joint model uses exactly the same weights as the cascaded model; the only difference is that the joint model takes into account information about the seminar labels when choosing named-entity labels. This is an example of how domain-specific information from the main task can improve performance on a more standard, general-purpose subtask.

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t is punctuation
w_t appears in list of first names, last names, honorifics, etc.
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-2, 2]$
Conjunction $q_k(\mathbf{x}, t)$ and $q_{k'}(\mathbf{x}, t)$ for all features k, k'
Conjunction $q_k(\mathbf{x}, t)$ and $q_{k'}(\mathbf{x}, t + 1)$ for all features k, k'

Table 3: Input features $q_k(\mathbf{x}, t)$ for the ACE named-entity data. In the above w_t is the word at position t, and w ranges over all words in the training data.

Figure 2 shows the difference in performance between joint and cascaded decoding as a function of training set size. Cascaded decoding with the full training set of 242 emails performs equivalently to joint decoding on only 181 training instances, a 25% reduction in the training set.

In summary, even with a simple cascaded training method on a well-studied data set, joint decoding performs better for transfer than cascaded decoding.

6.2 Entity Recognition

In this section we give results on joint decoding for transfer between two newswire data sets with similar but overlapping label sets. The Automatic Content Extraction (ACE) data set is another standard entity recognition data

[1] We omit one relevant paper (Peshkin & Pfeffer, 2003) because its evaluation method differs from all the other previous work.

	Transfer Type		
	none	cascaded	joint
Person name	81.0	86.9	87.3
Person nominal	34.9	36.1	42.4
Organization name	53.9	62.6	61.1
Organization nominal	33.7	35.3	40.8
GPE name	78.5	84.0	84.0
GPE nominal	51.2	54.1	59.2

Table 4: Comparison of F_1 performance between joint and cascaded training on the ACE entity recognition task. GPE means geopolitical entities, such as countries. Joint decoding helps most on the harder nominal (common noun) references. These results were obtained using a small subset of the training set.

set, containing 422 stories from newspaper, newswire, and broadcast news. Unlike the CoNLL entity recognition data set, in which only proper names of entities are annotated, the ACE data includes annotation both for named entities like *United States*, and also nominal mentions of entities like *the nation*. Thus, although the input text has similar distribution in the CoNLL NER and ACE data set, the label distributions are very different.

Current state-of-the-art systems for the ACE task (Florian et al., 2004) use the predictions of other named-entity recognizers as features, that is, they use cascaded transfer. In this experiment, we test whether the transfer between these datasets can be further improved using joint decoding. We train a CRF entity recognizer on the ACE dataset, with the output of a named-entity entity recognizer trained on the CoNLL 2003 English data set. The CoNLL recognizer is the same CRF as was used in the previous experiment. In these results, we use a subset of 10% of the ACE training data. Table 3 lists the features we use. Table 4 compares the results on some representative entity types. Again, cascaded decoding for transfer is better than no transfer at al, and joint decoding is better than cascaded decoding. Interestingly, joint decoding has most impact on the harder nominal references, showing marked improvement over the cascaded approach.

7 Related Work

Researchers have begun to accumulate experimental evidence that joint training and decoding yields better performance than the cascaded approach. As mentioned earlier, the original work on dynamic CRFs (Sutton et al., 2004) demonstrated improvement due to joint training in the domains of part-of-speech tagging and noun-phrase chunking. Also, Carreras and Marquez (Carreras & Màrquez, 2004) have obtained increased performance in clause finding by training a cascade of perceptrons to minimize a single global error function. Finally, Miller et al. (Miller et al., 2000) have combined entity recognition, parsing, and relation extraction into a jointly-trained single statistical parsing model that achieves improved performance on all the subtasks.

Part of the contribution of the current work is to suggest that joint decoding can be effective even when joint training is not possible because jointly-labeled data is unavailable. For example, Miller et al. report that they originally attempted to annotate newswire articles for all of parsing, relations, and named entities, but they stopped because the annotation was simply too expensive. Instead they hand-labeled relations only, assigning parse trees to the training set using a standard statistical parser, which is potentially less flexible than the cascaded training, because the model for main task is trained explicitly to match the noisy subtask predictions, rather than being free to correct them.

In the speech community, it is common to compose separately trained weighted finite-state transducers (Mohri et al., 2002) for joint decoding. Our method extends this work to conditional models. Ordinarily, higher-level transducers depend only on the output of the previous transducer: a transducer for the lexicon, for example, consumes only phonemes, not the original speech signal. In text, however, such an approach is not sensible, because there is simply not enough information in the named-entity labels, for example, to do extraction if the original words are discarded. In a conditional model, weights in higher-level transducers are free to depend on arbitrary features of the original input without any additional complexity in the finite-state structure.

Finally, stacked sequential learning (Cohen & Carvalho, 2005) is another potential method for combining the results of the subtask transducers. In this general meta-learning method for sequential classification, first a base classifier predicts the label at each time step, and then a higher-level classifier makes the final prediction, including as features a window of predictions from the base classifier. For transfer learning, this would correspond to having an independent base model for each subtask (e.g., independent CRFs for named-entity and seminars), and then having a higher-level CRF that includes as a feature the predictions from the base models.

8 Conclusion

In this paper we have shown that joint decoding improves transfer between interdependent NLP tasks, even when the old task is named-entity recognition, for which highly accurate systems exist. The rich features afforded by a conditional model allow the new task to influence the predictions of the old task, an effect that is only possible with joint decoding.

It is now common for researchers to publicly release trained models for standard tasks such as part-of-speech tagging, named-entity recognition, and parsing. This paper has implications for how such standard tools are packaged. Our results suggest that off-the-shelf NLP tools will need not only to provide a single-best prediction, but also to be engineered so that they can easily communicate distributions over predictions to models for higher-level tasks.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grants #IIS-0326249 and #IIS-0427594, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Byrd, R. H., Nocedal, J., & Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Program.*, *63*, 129–156.

Califf, M. E., & Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 328–334).

Carreras, X., & Marquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. *Proceedings of CoNLL-2004*.

Carreras, X., & Màrquez, L. (2004). Online learning via global feedback for phrase recognition. In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Ciravegna, F. (2001). Adaptive information extraction from text by rule induction and generalisation. *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*.

Cohen, W. W., & Carvalho, V. R. (2005). Stacked sequential learning. *International Joint Conference on Artificial Intelligence* (pp. 671–676).

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, *5(3)*, 142–150.

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., Roukos, S., & Zhang, T. (2004). A statistical model for multilingual entity detection and tracking. *In* HLT/NAACL 2004.

Freitag, D. (1998). *Machine learning for information extraction in informal domains*. Doctoral dissertation, Carnegie Mellon University.

Frietag, D., & McCallum, A. (1999). Information extraction with HMMs and shrinkage. *AAAI Workshop on Machine Learning for Information Extraction*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*.

Miller, S., Fox, H., Ramshaw, L. A., & Weischedel, R. M. (2000). A novel use of statistical parsing to extract information from text. *ANLP 2000* (pp. 226–233).

Mohri, M., Pereira, F., & Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, *16*, 69–88.

Pal, C., Sutton, C., & McCallum, A. (2005). *Fast inference and learning with sparse belief propagation* (Technical Report IR-433). Center for Intelligent Information Retrieval, University of Massachusetts.

Peshkin, L., & Pfeffer, A. (2003). Bayesian information extraction network. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Roth, D., & Wen-tau Yih (2001). Relational learning via propositional algorithms: An information extraction case study. *International Joint Conference on Artificial Intelligence* (pp. 1257–1263).

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proceedings of HLT-NAACL 2003*.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning*, 233–272.

Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*.

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *Proceedings of CoNLL-2003* (pp. 142–147). Edmonton, Canada.