Learning Representations for Weakly Supervised Natural Language Processing Tasks

Fei Huang*
Temple University

Arun Ahuja**
Northwestern University

Doug Downey†
Northwestern University

Yi Yang‡
Northwestern University

Yuhong Guo*
Temple University

Alexander Yates*
Temple University

Finding the right representations for words is critical for building accurate NLP systems when domain-specific labeled data for the task is scarce. This article investigates novel techniques for extracting features from n-gram models, Hidden Markov Models, and other statistical language models, including a novel Partial Lattice Markov Random Field model. Experiments on part-of-speech tagging and information extraction, among other tasks, indicate that features taken from statistical language models, in combination with more traditional features, outperform traditional representations alone, and that graphical model representations outperform n-gram models, especially on sparse and polysemous words.

* 1805 N. Broad St., Wachman Hall 324, Philadelphia, PA 19122, USA.
E-mail: {fei.huang,yuhong,yates}@temple.edu.
** 2133 Sheridan Road, Evanston, IL, 60208. E-mail: ahuja@eecs.northwestern.edu.
† 2133 Sheridan Road, Evanston, IL, 60208. E-mail: ddowney@eecs.northwestern.edu.
‡ 2133 Sheridan Road, Evanston, IL, 60208. E-mail: yya518@eecs.northwestern.edu.

Submission received: 13 June 2012; revised submission received: 25 November 2012, accepted for publication: 15 January 2013.

doi:10.1162/COLI_a_00167

1. Introduction

NLP systems often rely on hand-crafted, carefully engineered sets of features to achieve strong performance. Thus, a part-of-speech (POS) tagger would traditionally use a feature like, "the previous token is `the`" to help classify a given token as a noun or adjective. For supervised NLP tasks with sufficient domain-specific training data, these traditional features yield state-of-the-art results. However, NLP systems are increasingly being applied to the Web, scientific domains, personal communications like e-mails and tweets, among many other kinds of linguistic communication. These texts have very different characteristics from traditional training corpora in NLP. Evidence from POS tagging (Blitzer, McDonald, and Pereira 2006; Huang and Yates 2009), parsing (Gildea 2001; Sekine 1997; McClosky 2010), and semantic role labeling (SRL) (Pradhan, Ward, and Martin 2007), among other NLP tasks (Daumé III and Marcu 2006; Chelba and Acero 2004; Downey, Broadhead, and Etzioni 2007; Chan and Ng 2006; Blitzer, Dredze, and Pereira 2007), shows that the accuracy of supervised NLP systems degrades significantly when tested on domains different from those used for training. Collecting labeled training data for each new target domain is typically prohibitively expensive. In this article, we investigate representations that can be applied to **weakly supervised** learning, that is, learning when domain-specific labeled training data are scarce.

A growing body of theoretical and empirical evidence suggests that traditional, manually crafted features for a variety of NLP tasks limit systems' performance in this weakly supervised learning for two reasons. First, feature *sparsity* prevents systems from generalizing accurately, because many words and features are not observed in training. Also because word frequencies are Zipf-distributed, this often means that there is little relevant training data for a substantial fraction of parameters (Bikel 2004b), especially in new domains (Huang and Yates 2009). For example, word-type features form the backbone of most POS-tagging systems, but types like "gene" and "pathway" show up frequently in biomedical literature, and rarely in newswire text. Thus, a classifier trained on newswire data and tested on biomedical data will have seen few training examples related to sentences with features "gene" and "pathway" (Blitzer, McDonald, and Pereira 2006; Ben-David et al. 2010).

Further, because words are *polysemous*, word-type features prevent systems from generalizing to situations in which words have different meanings. For instance, the word type "signaling" appears primarily as a present participle (VBG) in Wall Street Journal (WSJ) text, as in, "Interest rates rose, signaling that . . . " (Marcus, Marcinkiewicz, and Santorini 1993). In biomedical text, however, "signaling" appears primarily in the phrase "signaling pathway," where it is considered a noun (NN) (PennBioIE 2005); this phrase *never* appears in the WSJ portion of the Penn Treebank (Huang and Yates 2010).

Our response to the sparsity and polysemy challenges with traditional NLP representations is to seek new representations that allow systems to generalize to previously unseen examples. That is, we seek representations that permit classifiers to have close to the same accuracy on examples from other domains as they do on the domain of the training data. Our approach depends on the well-known **distributional hypothesis**, which states that a word's meaning is identified with the contexts in which it appears (Harris 1954; Hindle 1990). Our goal is to develop probabilistic statistical language models that describe the contexts of individual words accurately. We then construct **representations**, or mappings from word tokens and types to real-valued vectors, from statistical language models. Because statistical language models are designed to model words' contexts, the features they produce can be used to combat problems with polysemy. And by careful design of the statistical language models, we can limit

the number of features that they produce, controlling how sparse those features are in training data.

Our specific contributions are as follows:

1. We show how to generate representations from a variety of language models, including n-gram models, Brown clusters, and Hidden Markov Models (HMMs). We also introduce a Partial-Lattice Markov Random Field (PL-MRF), which is a tractable variation of a Factorial Hidden Markov Model (Ghahramani and Jordan 1997) for language modeling, and we show how to produce representations from it.
2. We quantify the performance of these representations in experiments on POS tagging in a domain adaptation setting, and weakly supervised information extraction (IE). We show that the graphical models outperform n-gram representations, even when the n-gram models leverage larger corpora for training. The PL-MRF representation achieves a state-of-the-art 93.8% accuracy on a biomedical POS tagging task, which represents a 5.5 percentage point absolute improvement over more traditional POS tagging representations, a 4.8 percentage point improvement over a tagger using an n-gram representation, and a 0.7 percentage point improvement over a tagger with an n-gram representation using several orders of magnitude more training data. The HMM representation improves over the n-gram model by 7 percentage points on our IE task.
3. We analyze how sparsity, polysemy, and differences between domains affects the performance of a classifier using different representations. Results indicate that statistical language model representations, and especially graphical model representations, provide the best features for sparse and polysemous words.

The next section describes background material and related work on representation learning for NLP. Section 3 presents novel representations based on statistical language models. Sections 4 and 5 discuss evaluations of the representations, first on sequence-labeling tasks in a domain adaptation setting, and second on a weakly supervised set-expansion task. Section 6 concludes and outlines directions for future work.

2. Background and Previous Work on Representation Learning

2.1 Terminology and Notation

In a traditional machine learning task, the goal is to make predictions on test data using a hypothesis that is optimized on labeled training data. In order to do so, practitioners predefine a set of features and try to estimate classifier parameters from the observed features in the training data. We call these feature sets **representations** of the data.

Formally, let $\mathcal{X}$ be an instance space for a learning problem. Let $\mathcal{Z}$ be the space of possible labels for an instance, and let $f\colon \mathcal{X} \to \mathcal{Z}$ be the target function to be learned. A *representation* is a function $R\colon \mathcal{X} \to \mathcal{Y}$, for some suitable feature space $\mathcal{Y}$ (such as $\mathbb{R}^{\mathrm{d}}$). We refer to dimensions of $\mathcal{Y}$ as **features**, and for an instance $x \in \mathcal{X}$ we refer to values for particular dimensions of $R(x)$ as *features* of x. Given a set of training examples, a learning machine's task is to select a **hypothesis** h from the *hypothesis space* $\mathcal{H}$, a subset of $\mathcal{Z}^{R(\mathcal{X})}$. Errors by the hypothesis are measured using a **loss function** $\mathcal{L}(x, R, f, h)$ that

measures the cost of the mismatch between the target function $f(x)$ and the hypothesis $h(R(x))$.

As an example, the instance set for POS tagging in English is the set of all English sentences, and $\mathcal{Z}$ is the space of POS sequences containing labels like NN (for noun) and VBG (for present participle). The target function f is the mapping between sentences and their correct POS labels. A traditional representation in NLP converts sentences into sequences of vectors, one for each word position. Each vector contains values for features like, "+1 if the word at this position ends with `-tion`, and 0 otherwise." A typical loss function would count the number of words that are tagged differently by $f(x)$ and $h(R(x))$.

2.2 Representation-Learning Problem Formulation

Machine learning theory assumes that there is a distribution D over $\mathcal{X}$ from which data is sampled. Given a training set $S = \{(x_1, z_1), \ldots, (x_N, z_N)\} \sim (D(\mathcal{X}), \mathcal{Z})^N$, a fixed representation R, a hypothesis space $\mathcal{H}$, and a loss function $\mathcal{L}$, a machine learning algorithm seeks to identify the hypothesis in $\mathcal{H}$ that will minimize the expected loss over samples from distribution D:

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \mathbf{E}_{x \sim D(\mathcal{X})} \mathcal{L}(x, R, f, h) \tag{1}$$

The representation-learning paradigm breaks the traditional notion of a fixed representation R. Instead, we allow a space of possible representations $\mathcal{R}$. The full learning problem can then be formulated as the task of identifying the best $R \in \mathcal{R}$ and $h \in \mathcal{H}$ simultaneously:

$$R^*, h^* = \operatorname*{argmin}_{R \in \mathcal{R}, h \in \mathcal{H}} \mathbf{E}_{x \sim D(\mathcal{X})} \mathcal{L}(x, R, f, h) \tag{2}$$

The representation-learning problem formulation in Equation (2) can in fact be reduced to the general learning formulation in Equation (1) by setting the fixed representation R to be the identity function, and setting the hypothesis space to be $\mathcal{R} \times \mathcal{H}$ from the representation-learning task. We introduce the new formulation primarily as a way of changing the perspective on the learning task: most NLP systems consider a fixed, manually crafted transformation of the original data to some new space, and investigate hypothesis classes over that space. In the new formulation, systems learn the transformation to the feature space, and then apply traditional classification or regression algorithms.

2.3 Theory on Domain Adaptation

We refer to the distribution D over the instance space $\mathcal{X}$ as a **domain**. For example, the newswire domain is a distribution over sentences that gives high probability to sentences about governments and current events; the biomedical literature domain gives high probability to sentences about proteins and regulatory pathways. In **domain adaptation**, a system observes a set of training examples $(R(x), f(x))$, where instances $x \in \mathcal{X}$ are drawn from a **source** domain $\mathcal{D}_S$, to learn a hypothesis for classifying examples drawn from a separate **target** domain $\mathcal{D}_T$. We assume that large quantities of unlabeled data are available for the source domain and target domain, and call these

samples U_S and U_T, respectively. For any domain $\mathcal{D}$, let $R(\mathcal{D})$ represent the induced distribution over the feature space $\mathcal{Y}$ given by $Pr_{R(\mathcal{D})}[y] = Pr_{\mathcal{D}}[\{x \text{ such that } R(x) = y\}]$.

Previous work by Ben-David et al. (2007, 2010) proves theoretical bounds on an open-domain learning machine's performance. Their analysis shows that the choice of representation is crucial to domain adaptation. A good choice of representation must allow a learning machine to achieve low error rates on the source domain. Just as important, however, is that the representation must simultaneously make the source and target domains look as similar to one another as possible. That is, if the labeling function f is the same on the source and target domains, then for every $h \in H$, we can provably bound the error of h on the target domain by its error on the source domain plus a measure of the distance between $\mathcal{D}_S$ and $\mathcal{D}_T$:

$$\mathrm{E}_{x\sim\mathcal{D}_T}\mathcal{L}(x, R, f, h) \leq \mathrm{E}_{x\sim\mathcal{D}_S}\mathcal{L}(x, R, f, h) + d_1(R(\mathcal{D}_S), R(\mathcal{D}_T)) \tag{3}$$

where the variation divergence d_1 is given by

$$d_1(\mathcal{D}, \mathcal{D}') = 2 \sup_{B\in\mathcal{B}} |Pr_{\mathcal{D}}[B] - Pr_{\mathcal{D}'}[B]| \tag{4}$$

where $\mathcal{B}$ is the set of measurable sets under $\mathcal{D}$ and $\mathcal{D}'$ (Ben-David et al. 2007, 2010).

Crucially, the distance between domains depends on the features in the representation. The more that features appear with different frequencies in different domains, the worse this bound becomes. In fact, one lower bound for the d_1 distance is the accuracy of the best classifier for predicting whether an unlabeled instance $y = R(x)$ belongs to domain S or T (Ben-David et al. 2010). Thus, if R provides one set of common features for examples from S, and another set of common features for examples from T, the domain of an instance becomes easy to predict, meaning the distance between the domains grows, and the bound on our classifier's performance grows worse.

In light of Ben-David et al.'s theoretical findings, traditional representations in NLP are inadequate for domain adaptation because they contribute to the d_1 distance between domains. Although many previous studies have shown that lexical features allow learning systems to achieve impressively low error rates during training, they also make texts from different domains look very dissimilar. For instance, a feature based on the word "bank" or "CEO" may be common in a domain of newswire text, but scarce or nonexistent in, say, biomedical literature. Ben David et al.'s theory predicts greater variance in the error rate of the target domain classifier as the distance grows.

At the same time, traditional representations contribute to **data sparsity**, a lack of sufficient training data for the relevant parameters of the system. In traditional supervised NLP systems, there are parameters for each word type in the data, or perhaps even combinations of word types. Because vocabularies can be extremely large, this leads to an explosion in the number of parameters. As a consequence, for many of their parameters, supervised NLP systems have zero or only a handful of relevant labeled examples (Bikel 2004a, 2004b). No matter how sophisticated the learning technique, it is difficult to estimate parameters without relevant data. Because vocabularies differ across domains, domain adaptation greatly exacerbates this issue of data sparsity.

2.4 Problem Formulation for the Domain Adaptation Setting

Formally, we define the task of representation learning for domain adaptation as the following optimization problem: Given a set of unlabeled instances U_S drawn from the

source domain and unlabeled instances U_T from the target domain, as well as a set of labeled instances L_S drawn from the source domain, identify a function R^* from the space of possible representations $\mathcal{R}$ that minimizes

$$R^*, h^* = \underset{R \in \mathcal{R}, h \in \mathcal{H}}{\operatorname{argmin}} \left(\mathrm{E}_{x \sim \mathcal{D}_S} \mathcal{L}(x, R, f, h) \right) + \lambda d_1(R(\mathcal{D}_S), R(\mathcal{D}_T)) \quad (5)$$

where λ is a free parameter.

Note that there is an underlying tension between the two terms of the objective function: The best representation for the source domain would naturally include domain-specific features, and allow a hypothesis to learn domain-specific patterns. We are aiming, however, for the best *general* classifier, which happens to be trained on training data from one domain (or a few domains). The domain-specific features contribute to distance between domains, and to classifier errors on data taken from domains not seen in training. By optimizing for this combined objective function, we allow the optimization method to trade off between features that are best for classifying source-domain data and features that allow generalization to new domains.

Unlike the representation-learning problem-formulation in Equation (2), Equation (5) does *not* reduce to the standard machine-learning problem (Equation (1)). In a sense, the d_1 term acts as a regularizer on $\mathcal{R}$, which also affects $\mathcal{H}$. Representation learning for domain adaptation is a fundamentally novel learning task.

2.5 Tractable Representation Learning: Statistical Language Models as Representations

For most hypothesis classes and any interesting space of representations, Equations (2) and (5) are completely intractable to optimize exactly. Even given a fixed representation, it is intractable to compute the best hypothesis for many hypothesis classes. And the d_1 metric is intractable to compute from samples of a distribution, although Ben-David et al. (2007, 2010) propose some tractable bounds. We view these problem formulations as high-level goals rather than as computable objectives.

As a tractable objective, in this work we describe an investigation into the use of statistical language models as a way to represent the meanings of words. This approach depends on the well-known **distributional hypothesis**, which states that a word's meaning is identified with the contexts in which it appears (Harris 1954; Hindle 1990). From this hypothesis, we can formulate the following testable prediction, which we call the **statistical language model representation hypothesis**, or LMRH:

> To the extent that a model accurately describes a word's possible contexts, parameters of that model are highly informative descriptors of the word's meaning, and are therefore useful as features in NLP tasks like POS tagging, chunking, NER, and information extraction.

The LMRH says, essentially, that for NLP tasks, we can decouple the task of optimizing a representation from the task of optimizing a hypothesis. To learn a representation, we can train a statistical language model on unlabeled text, and then use parameters or latent states from the statistical language model to create a representation function. Optimizing a hypothesis then follows the standard learning framework, using the representation from the statistical language model.

The LMRH is similar to the manifold and cluster assumptions behind other semi-supervised approaches to machine learning, such as Alternating Structure Optimization (ASO) (Ando and Zhang 2005) and Structural Correspondence Learning (SCL) (Blitzer, McDonald, and Pereira 2006). All three of these techniques use predictors built on unlabeled data as a way to harness the manifold and cluster assumptions. However, the LMRH is distinct from at least ASO and SCL in important ways. Both ASO and SCL create multiple "synthetic" or "pivot" prediction tasks using unlabeled data, and find transformations of the input feature space that perform well on these tasks. The LMRH, on the other hand, is more specific — it asserts that for language problems, if we optimize word representations on a single task (the language modeling task), this will lead to strong performance on weakly supervised tasks. In reported experiments on NLP tasks, both ASO and SCL use certain synthetic predictors that are essentially language modeling tasks, such as the task of predicting whether the next token is of word type w. To the extent that these techniques' performance relies on language-modeling tasks as their "synthetic predictors," they can be viewed as evidence in support of the LMRH.

One significant consequence of the LMRH is that it allows us to leverage well-developed techniques and models from statistical language modeling. Section 3 presents a series of statistical language models that we investigate for learning representations for NLP.

2.6 Previous Work

There is a long tradition of NLP research on representations, mostly falling into one of four categories: 1) vector space models of meaning based on document-level lexical co-occurrence statistics (Salton and McGill 1983; Sahlgren 2006; Turney and Pantel 2010); 2) dimensionality reduction techniques for vector space models (Deerwester et al. 1990; Ritter and Kohonen 1989; Honkela 1997; Kaski 1998; Sahlgren 2001, 2005; Blei, Ng, and Jordan 2003; Väyrynen and Honkela 2004, 2005; Väyrynen, Honkela, and Lindqvist 2007); 3) using clusters that are induced from distributional similarity (Brown et al. 1992; Pereira, Tishby, and Lee 1993; Martin, Liermann, and Ney 1998) as non-sparse features (Miller, Guinness, and Zamanian 2004; Ratinov and Roth 2009; Lin and Wu 2009; Candito and Crabbe 2009; Koo, Carreras, and Collins 2008; Suzuki et al. 2009; Zhao et al. 2009); and, recently, 4) neural network statistical language models (Bengio 2008; Bengio et al. 2003; Morin and Bengio 2005; Mnih, Yuecheng, and Hinton 2009; Mnih and Hinton 2007, 2009) as representations (Weston, Ratle, and Collobert 2008; Collobert and Weston 2008; Bengio et al. 2009). Our work is a form of distributional clustering for representations, but where previous work has used bigram and trigram statistics to form clusters, we build sophisticated models that attempt to capture the context of a word, and hence its similarity to other words, more precisely. Our experiments show that the new graphical models provide representations that outperform those from previous work on several tasks.

Neural network statistical language models have recently achieved state-of-the-art perplexity results (Mnih and Hinton 2009), and representations based on them have improved in-domain chunking, NER, and SRL (Weston, Ratle, and Collobert 2008; Turian, Bergstra, and Bengio 2009; Turian, Ratinov, and Bengio 2010). As far as we are aware, Turian, Ratinov, and Bengio (2010) is the only other work to test a learned representation on a domain adaptation task, and they show improvement on out-of-domain NER with their neural net representations. Though promising, the neural network models are computationally expensive to train, and these statistical language models work only on fixed-length histories (n-grams) rather than full observation sequences. Turian,

Ratinov, and Bengio's (2010) tests also show that Brown clusters perform as well or better than neural net models on all of their chunking and NER tests. We concentrate on probabilistic graphical models with discrete latent states instead. We show that HMM-based and other representations significantly outperform the more commonly used Brown clustering (Brown et al. 1992) as a representation for domain adaptation settings of sequence-labeling tasks.

Most previous work on domain adaptation has focused on the case where some labeled data are available in both the source and target domains (Chan and Ng 2006; Daumé III and Marcu 2006; Blitzer, Dredze, and Pereira 2007; Daumé III 2007; Jiang and Zhai 2007a, 2007b; Dredze and Crammer 2008; Finkel and Manning 2009; Dredze, Kulesza, and Crammer 2010). Learning bounds for this domain-adaptation setting are known (Blitzer et al. 2007; Mansour, Mohri, and Rostamizadeh 2009). Approaches to this problem setting have focused on appropriately weighting examples from the source and target domains so that the learning algorithm can balance the greater relevance of the target-domain data with the larger source-domain data set. In some cases, researchers combine this approach with semi-supervised learning to include unlabeled examples from the target domain as well (Daumé III, Kumar, and Saha 2010). These techniques do not handle open-domain corpora like the Web, where they require expert input to acquire labels for each new single-domain corpus, and it is difficult to come up with a representative set of labeled training data for each domain. Our technique requires only unlabeled data from each new domain, which is significantly easier and cheaper to acquire. Where target-domain labeled data is available, however, these techniques can in principle be combined with ours to improve performance, although this has not yet been demonstrated empirically.

A few researchers have considered the more general case of domain adaptation without labeled data in the target domain. Perhaps the best known is Blitzer, McDonald, and Pereira's (2006) Structural Correspondence Learning (SCL). SCL uses "pivot" words common to both source and target domains, and trains linear classifiers to predict these pivot words from their context. After an SVD reduction of the weight vectors for these linear classifiers, SCL projects the original features through these weight vectors to obtain new features that are added to the original feature space. Like SCL, our language modeling techniques attempt to predict words from their context, and then use the output of these predictions as new features. Unlike SCL, we attempt to predict all words from their context, and we rely on traditional probabilistic methods for language modeling. Our best learned representations, which involve significantly different techniques from SCL, especially latent-variable probabilistic models, significantly outperform SCL in POS tagging experiments.

Other approaches to domain adaptation without labeled data from the target domain include Satpal and Sarawagi (2007), who show that by changing the optimization function during conditional random field (CRF) training, they can learn classifiers that port well to new domains. Their technique selects feature subsets that minimize the distance between training text and unlabeled test text, but unlike our techniques, theirs cannot learn representations with features that do not appear in the original feature set. In contrast, we learn hidden features through statistical language models. McClosky, Charniak, and Johnson (2010) use classifiers from multiple source domains and features that describe how much a target document diverges from each source domain to determine an optimal weighting of the source-domain classifiers for parsing the target text. However, it is unclear if this "source-combination" technique works well on domains that are not mixtures of the various source domains. Dai et al. (2007) use KL-divergence between domains to directly modify the parameters of their naive Bayes model for a

text classification task trained purely on the source domain. These last two techniques are not representation learning, and are complementary to our techniques.

Our representation-learning approach to domain adaptation is an instance of semi-supervised learning. Of the vast number of semi-supervised approaches to sequence labeling in NLP, the most relevant ones here include Suzuki and Isozaki's (2008) combination of HMMs and CRFs that uses over a billion words of unlabeled text to achieve the current best performance on in-domain chunking, and semi-supervised approaches to improving in-domain SRL with large quantities of unlabeled text (Weston, Ratle, and Collobert 2008; Deschacht and Moens 2009; and Fürstenau and Lapata 2009). Ando and Zhang's (2005) semi-supervised sequence labeling technique has been tested on a domain adaptation task for POS tagging (Blitzer, McDonald, and Pereira 2006); our representation-learning approaches outperform it. Unlike most semi-supervised techniques, we concentrate on a particularly simple task decomposition: unsupervised learning for new representations, followed by standard supervised learning. In addition to our task decomposition being simple, our learned representations are also task-independent, so we can learn the representation once, and then apply it to any task.

One of the best-performing representations that we consider for domain adaptation is based on the HMM (Rabiner 1989). HMMs have of course also been used for supervised, semi-supervised, and unsupervised POS tagging on a single domain (Banko and Moore 2004; Goldwater and Griffiths 2007). Recent efforts on improving unsupervised POS tagging have focused on incorporating prior knowledge into the POS induction model (Graça et al. 2009; Toutanova and Johnson 2007), or on new training techniques like contrastive estimation (Smith and Eisner 2005) for alternative sequence models. Despite the fact that completely connected, standard HMMs perform poorly at the POS induction task (Johnson 2007), we show that they still provide very useful features for a supervised POS tagger. Experiments in information extraction have previously also shown that HMMs provide informative features for this quite different, semantic processing task (Downey, Schoenmackers, and Etzioni 2007; Ahuja and Downey 2010).

This article extends our previous work on learning representations for domain adaptation (Huang and Yates 2009, 2010) by investigating new language representations—the naive Bayes representation and PL-MRF representation (Huang et al. 2011)—by analyzing results in terms of polysemy, sparsity, and domain divergence; by testing on new data sets including a Chinese POS tagging task; and by providing an empirical comparison with Brown clusters as representations.

3. Learning Representations of Distributional Similarity

In this section, we will introduce several representation learning models.

3.1 Traditional POS-Tagging Representations

As an example of our terminology, we begin by describing a representation used in traditional POS taggers (this representation will later form a baseline for our POS tagging experiments). The instance set $\mathcal{X}$ is the set of English sentences, and $\mathcal{Z}$ is the set of POS tag sequences. A traditional representation TRAD-R maps a sentence $\mathbf{x} \in \mathcal{X}$ to a sequence of boolean-valued vectors, one vector per word x_i in the sentence. Dimensions for each latent vector include indicators for the word type of x_i and various orthographic features. Table 1 presents the full list of features in TRAD-R. Because our IE task classifies word types rather than tokens, this baseline is not appropriate for that task. Herein, we

Table 1
Summary of features provided by our representations. $\forall_a \mathbf{1}[g(a)]$ represents a set of boolean features, one for each value of a, where the feature is true iff $g(a)$ is true. x_i represents a token at position i in sentence $\mathbf{x}$, w represents a word type, Suffixes = {-ing,-ogy,-ed,-s,-ly,-ion,-tion,-ity}, k (and $\mathbf{k}$) represents a value for a latent state (set of latent states) in a latent-variable model, $\mathbf{y}^*$ represents the maximum a posteriori sequence of states $\mathbf{y}$ for $\mathbf{x}$, y_i is the latent variable for x_i, and $y_{i,j}$ is the latent variable for x_i at layer j. prefix(y,p) is the p-length prefix of the Brown cluster y.

Representation	Features
TRAD-R	$\forall_w \mathbf{1}[x_i = w]$ $\forall_{s \in \text{Suffixes}} \mathbf{1}[x_i \text{ ends with } s]$ $\mathbf{1}[x_i \text{ contains a digit}]$
n-GRAM-R	$\forall_{\mathbf{w}',\mathbf{w}''} P(\mathbf{w}' w \mathbf{w}'')/P(w)$
LSA-R	$\forall_{w,j} \{\mathbf{v}'_{left}(w)\}_j$ $\forall_{w,j} \{\mathbf{v}'_{right}(w)\}_j$
NB-R	$\forall_k \mathbf{1}[y_i^* = k]$
HMM-TOKEN-R	$\forall_k \mathbf{1}[y_i^* = k]$
HMM-TYPE-R	$\forall_k P(y = k \mid x = w)$
I-HMM-TOKEN-R	$\forall_{j,k} \mathbf{1}[y_{i,j}^* = k]$
I-HMM-TYPE-R	$\forall_{j,k} P(y_{.,j} = k \mid x = w)$
BROWN-TOKEN-R	$\forall_{j \in \{-2,-1,0,1,2\}}$ $\forall_{p \in \{4,6,10,20\}} \text{prefix}(y_{i+j}, p)$
BROWN-TYPE-R	$\forall_p \text{prefix}(y, p)$
LATTICE-TOKEN-R	$\forall_{j,k} \mathbf{1}[y_{i,j}^* = k]$
LATTICE-TYPE-R	$\forall_{\mathbf{k}} P(\mathbf{y} = \mathbf{k} \mid x = w)$

describe how we can learn representations R by using a variety of statistical language models, for use in both our IE and POS tagging tasks. All representations for POS tagging inherit the features from TRAD-R; all representations for IE do not.

3.2 *n*-gram Representations

n-gram representations, which we call n-GRAM-R, model a word type w in terms of the n-gram contexts in which w appears in a corpus. Specifically, for word w we generate the vector $P(\mathbf{w}' w \mathbf{w}'')/P(w)$, the conditional probability of observing the word sequence $\mathbf{w}'$ to the left and $\mathbf{w}''$ to the right of w. Each dimension in this vector represents a combination of the left and right words. The experimental section describes the particular corpora and statistical language modeling methods used for estimating probabilities. Note that these features depend only on the word type w, and so for every token $x_i = w$, n-GRAM-R provides the same set of features regardless of local context.

One drawback of n-GRAM-R is that it does not handle sparsity well—the features are as sparsely observed as the lexical features in TRAD-R, except that n-GRAM-R features can be obtained from larger corpora. As an alternative, we apply latent semantic analysis (LSA) (Deerwester et al. 1990) to compute a reduced-rank representation. For word w, let $\mathbf{v}_{right}(w)$ represent the right context vector of w, which in each dimension contains the value of $P(ww'')/P(w)$ for some word w'', as observed in the n-gram model. Similarly, let $\mathbf{v}_{left}(w)$ be the left context vector of w. We apply LSA to the set

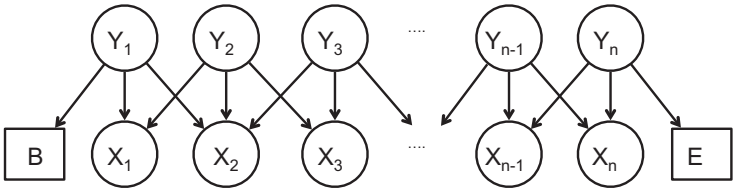


Figure 1
A graphical representation of the naive Bayes statistical language model. The *B* and *E* are special dummy words for the beginning and end of the sentence.

of right context vectors and the set of left context vectors separately,[1] to find reduced-rank versions $\mathbf{v}'_{right}(w)$ and $\mathbf{v}'_{left}(w)$, where each dimension represents a combination of several context word types. We then use each component of $\mathbf{v}'_{right}(w)$ and $\mathbf{v}'_{left}(w)$ as features. After experimenting with different choices for the number of dimensions to reduce our vectors to, we choose a value of 10 dimensions as the one that maximizes the performance of our supervised sequence labelers on held-out data. We call this model LSA-R.

3.3 A Context-Dependent Representation Using Naive Bayes

The *n*-GRAM-R and LSA-R representations always produce the same features **F** for a given word type w, regardless of the local context of a particular token $x_i = w$. Our remaining representations are all context-dependent, in the sense that the features provided for token x_i depend on the local context around x_i. We begin with a statistical language model based on the Naive Bayes model with categorical latent states $S = \{1, \ldots, K\}$. First, we form trigrams from our sentences. For each trigram, we form a separate Bayes net in which each token from the trigram is conditionally independent given the latent state. For tokens x_{i-1}, x_i, and x_{i+1}, the probability of this trigram given latent state $Y_i = y$ is given by:

$$P(x_{i-1}, x_i, x_{i+1}|y_i) = P_{left}(x_{i-1}|y_i)P_{mid}(x_i|y_i)P_{right}(x_{i+1}|y_i) \tag{6}$$

where P_{left}, P_{mid}, and P_{right} are multinomial distributions conditioned on the latent state. The probability of a whole sentence is then given by the product of the probabilities of its trigrams. Figure 1 shows a graphical representation of this model. We train our models using standard expectation-maximization (Dempster, Laird, and Rubin 1977) with random initialization of the parameters.

Because our factorization of the sentence does not take into account the fact that the trigrams overlap, the resulting statistical language model is mass-deficient. Worse still, it is throwing away information from the dependencies among trigrams which might help make better clustering decisions. Nevertheless, this model closely mirrors many of the clustering algorithms used in previous approaches to representation learning for sequence labeling (Ushioda 1996; Miller, Guinness, and Zamanian 2004; Koo, Carreras,

1 Compare with Dhillon, Foster, and Ungar (2011), who use canonical correlation analysis to find a simultaneous reduction of the left and right context vectors, a significantly more complex undertaking.

and Collins 2008; Lin and Wu 2009; Ratinov and Roth 2009), and therefore serves as an important benchmark.

Given a naive Bayes statistical language model, we construct an NB-R representation that produces $|S|$ boolean features $F_s(x_i)$ for each token x_i and each possible latent state $s \in S$:

$$F_s(x_i) = \begin{cases} \text{true} & \text{if } s = \arg\max_{s' \in S} P(x_{i-1}, x_i, x_{i+1} | y_i = s'), \\ \text{false} & \text{otherwise.} \end{cases}$$

For a reasonable choice of S (i.e., $|S| \ll |V|$), each feature should be observed often in a sufficiently large training data set. Therefore, compared with n-GRAM-R, NB-R produces far fewer features. On the other hand, its features for x_i depend not just on the contexts in which x_i has appeared in the statistical language model's training data, but also on x_{i-1} and x_{i+1} in the current sentence. Furthermore, because the range of the features is much more restrictive than real-valued features, it is less prone to data sparsity or variations across domains than real-valued features.

3.4 Context-Dependent, Structured Representations: The Hidden Markov Model

In previous work, we have implemented several representations based on hidden Markov models (Rabiner 1989), which we used for both sequential labeling (like POS tagging [Huang et al. 2011] and NP chunking [Huang and Yates 2009]) and IE (Downey, Schoenmackers, and Etzioni 2007). Figure 2 shows a graphical model of an HMM. An HMM is a generative probabilistic model that generates each word x_i in the corpus conditioned on a latent variable y_i. Each y_i in the model takes on integral values from 1 to K, and each one is generated by the latent variable for the preceding word, y_{i-1}. The joint distribution for a corpus $\mathbf{x} = (x_1, \ldots, x_N)$ and a set of state vectors $\mathbf{y} = (y_1, \ldots, y_N)$ is given by: $P(\mathbf{x}, \mathbf{y}) = \prod_i P(x_i|y_i)P(y_i|y_{i-1})$. Using expectation-maximization (EM) (Dempster, Laird, and Rubin 1977), it is possible to estimate the distributions for $P(x_i|y_i)$ and $P(y_i|y_{i-1})$ from unlabeled data.

We construct two different representations from HMMs, one for sequence-labeling tasks and one for IE. For sequence labeling, we use the Viterbi algorithm to produce the optimal setting $\mathbf{y}^*$ of the latent states for a given sentence $\mathbf{x}$, or $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} P(\mathbf{x}, \mathbf{y})$. We use the value of y_i^* as a new feature for x_i that represents a cluster of distributionally similar words. For IE, we require features for word types w, rather than tokens x_i. Applying Bayes' rule to the HMM parameters, we compute a distribution $P(Y|x = w)$, where Y is a single latent node, x is a single token, and w is its word type. We then use each of the K values for $P(Y = k|x = w)$, where k ranges from 1 to K, as features. This set

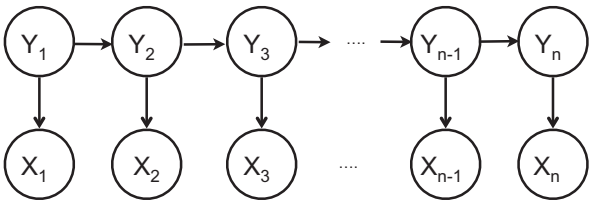


Figure 2
The Hidden Markov Model.

of features represents a "soft clustering" of w into K different clusters. We refer to these representations as HMM-TOKEN-R and HMM-TYPE-R, respectively.

We also compare against a multi-layer variation of the HMM from our previous work (Huang and Yates 2010). This model trains an ensemble of M independent HMM models on the same corpus, initializing each one randomly. We can then use the Viterbi-optimal decoded latent state of each independent HMM model as a separate feature for a token, or the posterior distribution for $P(Y|x = w)$ from each HMM as a separate set of features for each word type. We refer to this statistical language model as an I-HMM, and the representations as I-HMM-TOKEN-R and I-HMM-TYPE-R, respectively.

Finally, we compare against Brown clusters (Brown et al. 1992) as learned features. Although not traditionally described as such, Brown clustering involves constructing an HMM model in which each word type is restricted to having exactly one latent state that may generate it. Brown et al. describe a greedy agglomerative clustering algorithm for training this model on unlabeled text. Following Turian, Ratinov, and Bengio (2010), we use Percy Liang's implementation of this algorithm for our comparison, and we test runs with 100, 320, 1,000 and 3,200 clusters. We use features from these clusters identical to Turian et al.'s.[2] Turian et al. have shown that Brown clusters match or exceed the performance of neural network-based statistical language models in domain adaptation experiments for named-entity recognition, as well as in-domain experiments for NER and chunking.

Because HMM-based representations offer a small number of discrete states as features, they have a much greater potential to combat sparsity than do n-gram models. Furthermore, for token-based representations, these models can potentially handle polysemy better than n-gram statistical language models by providing different features in different contexts.

3.5 A Novel Lattice Statistical Language Model Representation

Our final statistical language model is a novel latent-variable statistical language model, called a Partial Lattice MRF (PL-MRF), with rich latent structure, shown in Figure 3. The model contains a lattice of $M \times N$ latent states, where N is the number of words in a sentence and M is the number of layers in the model. The dotted and solid lines in the figure together form a complete lattice of edges between these nodes; the PL-MRF uses only the solid edges. Formally, let $c = \lfloor \frac{N}{2} \rfloor$, where N is the length of the sentence; let i denote a position in the sentence, and let j denote a layer in the lattice. If $i < c$ and j is odd, or if j is even and $i > c$, we delete edges between $y_{i,j}$ and $y_{i,j+1}$ from the complete lattice. The same set of nodes remains, but the partial lattice contains fewer edges and paths between the nodes. A central "trunk" at $i = c$ connects all layers of the lattice, and branches from this trunk connect either to the branches in the layer above or the layer below (but not both).

The result is a model that retains most of the edges of the complete lattice, but unlike the complete lattice, it supports tractable inference. As $M, N \to \infty$, five out of every six edges from the complete lattice appear in the PL-MRF. However, the PL-MRF makes the branches conditionally independent from one another, except through the trunk. For instance, the left branch between layers 1 and 2 (($y_{1,1}, y_{1,2}$) and ($y_{2,1}, y_{2,2}$)) in Figure 3 are disconnected; similarly, the right branch between layers 2 and 3 (($y_{4,2}, y_{4,3}$) and ($y_{5,2}, y_{5,3}$)) are disconnected, except through the trunk and the observed nodes. As

2 Percy Liang's implementation is available at `http://metaoptimize.com/projects/wordreprs/`.

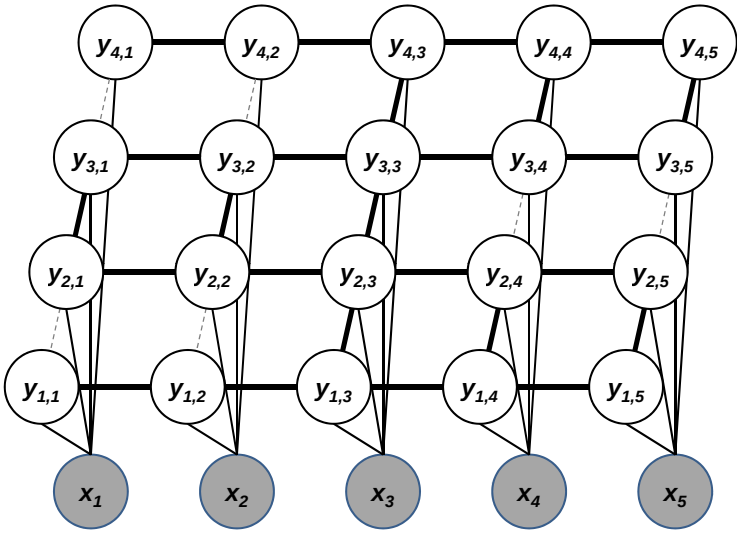


Figure 3
The PL-MRF model for a five-word sentence and a four-layer lattice. Dashed gray edges are part of a complete lattice, but not part of the PL-MRF.

a result, excluding the observed nodes, this model has a low **tree-width** of 2 (excluding observed nodes), and a variety of efficient dynamic programming and message-passing algorithms for training and inference can be readily applied (Bodlaender 1988). Our inference algorithm passes information from the branches inwards to the trunk, and then upward along the trunk, in time $O(K^4MN)$. In contrast, a fully connected lattice model has tree-width $= \min(M, N)$, making inference and learning intractable (Sutton, McCallum, and Rohanimanesh 2007), partly because of the difficulty in enumerating and summing over the exponentially-many configurations $\mathbf{y}$ for a given $\mathbf{x}$.

We can justify the choice of this model from a linguistic perspective as a way to capture the multi-dimensional nature of words. Linguists have long argued that words have many different features in a high dimensional space: They can be separately described by part of speech, gender, number, case, person, tense, voice, aspect, mass vs. count, and a host of semantic categories (agency, animate vs. inanimate, physical vs. abstract, etc.), to name a few (Sag, Wasow, and Bender 2003). In the PL-MRF, each layer of nodes is intended to represent some latent dimension of words.

We represent the probability distribution for PL-MRFs as log-linear models that decompose over cliques in the MRF graph. Let $Cliq(\mathbf{x}, \mathbf{y})$ represent the set of all maximal cliques in the graph of the MRF model for $\mathbf{x}$ and $\mathbf{y}$. Expressing the lattice model in log-linear form, we can write the marginal probability $P(\mathbf{x})$ of a given sentence $\mathbf{x}$ as:

$$\frac{\sum_{\mathbf{y}} \prod_{c \in Cliq(\mathbf{x},\mathbf{y})} \text{score}(c, \mathbf{x}, \mathbf{y})}{\sum_{\mathbf{x}',\mathbf{y}'} \prod_{c \in Cliq(\mathbf{x}',\mathbf{y}')} \text{score}(c, \mathbf{x}', \mathbf{y}')}$$

where $\text{score}(c, \mathbf{x}, \mathbf{y}) = \exp(\theta_c \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c))$. Our model includes parameters for transitions between two adjacent latent variables on layer j: $\theta^{trans}_{i,s,i+1,s',j}$ for $y_{i,j} = s$ and $y_{i+1,j} = s'$. It also includes observation parameters for latent variables and tokens, as well as for pairs of adjacent latent variables in different layers and their tokens: $\theta^{obs}_{i,j,s,w}$ and $\theta^{obs}_{i,j,s,j+1,s',w}$ for $y_{i,j} = s$, $y_{i,j+1} = s'$, and $x_i = w$.

As with our HMM models, we create two representations from PL-MRFs, one for tokens and one for types. For tokens, we decode the model to compute $\mathbf{y}^*$, the matrix of optimal latent state values for sentence $\mathbf{x}$. For each layer j and and each possible latent state value k, we add a boolean feature for token x_i that is true iff $\mathbf{y}^*_{i,j} = k$. For word types, we compute distributions over the latent state space. Let $\mathbf{y}$ be a column vector of latent variables for word type w. For a PL-MRF model with M layers of binary variables, there are 2^M possible values for $\mathbf{y}$. Our type representation computes a probability distribution over these 2^M possible values, and uses each probability as a feature for w.[3] We refer to these two representations as LATTICE-TOKEN-R and LATTICE-TYPE-R, respectively.

We train the PL-MRF using contrastive estimation (Smith and Eisner 2005), which iteratively optimizes the following objective function on a corpus $\mathbf{X}$:

$$\sum_{\mathbf{x} \in \mathbf{X}} \log \frac{\sum_{\mathbf{y}} \prod_{c \in Cliq(\mathbf{x},\mathbf{y})} \text{score}(c, \mathbf{x}, \mathbf{y})}{\sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x}), \mathbf{y}'} \prod_{c \in Cliq(\mathbf{x}',\mathbf{y}')} \text{score}(c, \mathbf{x}', \mathbf{y}')} \tag{7}$$

where $\mathcal{N}(\mathbf{x})$, the neighborhood of $\mathbf{x}$, indicates a set of perturbed variations of the original sentence $\mathbf{x}$. Contrastive estimation seeks to move probability mass away from the perturbed neighborhood sentences and onto the original sentence. We use a neighborhood function that includes all sentences which can be obtained from the original sentence by swapping the order of a consecutive pair of words. Training uses gradient descent over this non-convex objective function with a standard software package (Liu and Nocedal 1989) and converges to a local maximum or saddle point.

For tractability, we modify the training procedure to train the PL-MRF one layer at a time. Let θ_i represent the set of parameters relating to features of layer i, and let $\theta_{\neg i}$ represent all other parameters. We fix $\theta_{\neg 0} = \mathbf{0}$, and optimize θ_0 using contrastive estimation. After convergence, we fix $\theta_{\neg 1}$, and optimize θ_1, and so on. For training each layer, we use a convergence threshold of 10^{-6} on the objective function in Equation (7), and each layer typically converges in under 100 iterations.

4. Domain Adaptation with Learned Representations

We evaluate the representations described earlier on POS tagging and NP chunking tasks in a domain adaptation setting.

4.1 A Rich Problem Setting for Representation Learning

Existing supervised NLP systems are domain-dependent: There is a substantial drop in their performance when tested on data from a new domain. Domain adaptation is the task of overcoming this domain dependence. The aim is to build an accurate system for

3 This representation is only feasible for small numbers of layers, and in our experiments that require type representations, we used $M = 10$. For larger values of M, other representations are also possible. We also experimented with a representation which included only M possible values: For each layer l, we included $P(y_l = 0|w)$ as a feature. We used the less-compact representation in our experiments because results were better.

a target domain by training on labeled examples from a separate source domain. This problem is sometimes also called **transfer learning** (Raina et al. 2007).

Two of the challenges for NLP representations, sparsity and polysemy, are exacerbated by domain adaptation. New domains come with new words and phrases that appear rarely (or even not at all) in the training domain, thus increasing problems with data sparsity. And even for words that do appear commonly in both domains, the contexts around the words will change from the training domain to the target domain. As a result, domain adaptation adds to the challenge of handling polysemous words, whose meaning depends on context.

In short, domain adaptation is a challenging setting for testing NLP representations. We now present several experiments testing our representations against state-of-the-art POS taggers in a variety of domain adaptation settings, showing that the learned representations surpass the previous state-of-the-art, without requiring any labeled data from the target domain.

4.2 Experimental Set-up

For domain adaptation, we test our representations on two sequence labeling tasks: POS tagging and chunking. To incorporate learned representation into our models, we follow this general procedure, although the details vary by experiment and are given in the following sections. First, we collect a set of unannotated text from both the training domain and test domain. Second, we learn representations on the unannotated text. We then automatically annotate both the training and test data with features from the learned representation. Finally, we train a supervised linear-chain CRF model on the annotated training set and apply it to the test set.

A linear-chain CRF is a Markov random field (Darroch, Lauritzen, and Speed 1980) in which the latent variables form a path with edges only between consecutive nodes in the path, and all latent variables are globally conditioned on the observations. Let $\mathbf{X}$ be a random variable over data sequences, and $\mathbf{Z}$ be a random variable over corresponding label sequences. The conditional distribution over the label sequence $\mathbf{Z}$ given $\mathbf{X}$ has the form

$$p_\theta(\mathbf{Z} = \mathbf{z}|\mathbf{X} = \mathbf{x}) \propto \exp\left(\sum_i \sum_j \theta_j f_j(z_{i-1}, z_i, \mathbf{x}, i)\right) \quad (8)$$

where $f_j(z_{i-1}, z_i, \mathbf{x}, i)$ is a real-valued feature function of the entire observation sequence and the labels at positions i and $i-1$ in the label sequence, and θ_j is a parameter to be estimated from training data.

We use an open source CRF software package designed by Sunita Sarawagi to train and apply our CRF models.[4] As is standard, we use two kinds of feature functions: transition and observation. Transition feature functions indicate, for each pair of labels l and l', whether $z_i = l$ and $z_{i-1} = l'$. Boolean observation feature functions indicate, for each label l and each feature f provided by a representation, whether $z_i = l$ and x_i has feature f. For each label l and each real-valued feature f in representation R, real-valued observation feature functions have value $f(\mathbf{x})$ if $z_i = l$, and are zero otherwise.

4 Available from `http://sourceforge.net/projects/crf/`.

4.3 Domain Adaptation for POS Tagging

Our first experiment tests the performance of all the representations we introduced earlier on an English POS tagging task, trained on newswire text, to tag biomedical research literature. We follow Blitzer et al.'s experimental set-up. The labeled data consists of the WSJ portion of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993) as source domain data, and 561 labeled sentences (9,576 tokens) from the biomedical research literature database MEDLINE as target domain data (PennBioIE 2005). Fully 23% of the tokens in the labeled test text are never seen in the WSJ training data. The unlabeled data consists of the WSJ text plus 71,306 additional sentences of MEDLINE text (Blitzer, McDonald, and Pereira 2006). As a preprocessing step, we replace **hapax legomena** (defined as words that appear once in our unlabeled training data) with the special symbol `*UNKNOWN*`, and do the same for words in the labeled test sets that never appeared in any of our unlabeled training text.

For representations, we tested TRAD-R, *n*-GRAM-R, LSA-R, NB-R, HMM-TOKEN-R, I-HMM-TOKEN-R (between 2 and 8 layers), and LATTICE-TOKEN-R (8, 12, 16, and 20 layers). Each latent node in the I-HMMs had 80 possible values, creating $80^8 \approx 10^{15}$ possible configurations of the eight-layer I-HMM for a single word. Each node in our PL-MRF is binary, creating a much smaller number ($2^{20} \approx 10^6$) of possible configurations for each word in a 20-layer representation. To give the *n*-gram model the largest training data set available, we trained it on the Web 1Tgram corpus (Brants and Franz 2006). We included the top 500 most common *n*-grams for each word type, and then used mutual information on the training data to select the top 10,000 most relevant *n*-gram features for all word types, in order to keep the number of features manageable. We incorporated *n*-gram features as binary values indicating whether x_i appeared with the *n*-gram or not. For comparison, we also report on the performance of Brown clusters (100, 320, 1,000, and 3,200 possible clusters), following Turian, Ratinov, and Bengio (2010). Finally, we compare against Blitzer, McDonald, and Pereira (2006) SCL technique, described in Section 2.6, and the standard semi-supervised learning algorithm ASO (Ando and Zhang 2005), whose results on this task were previously reported by Blitzer, McDonald, and Pereira (2006).

Table 2 shows the results for the best variation of each kind of model—20 layers for the PL-MRF, 7 layers for the I-HMM, and 3,200 clusters for the Brown clustering. All statistical language model representations outperform the TRAD-R baseline.

In nearly all cases, learned representations significantly outperformed TRAD-R. The best representation, the 20-layer LATTICE-TOKEN-R, reduces error by 47% (35% on OOV) relative to the baseline TRAD-R, and by 44% (24% on out-of-vocabulary words (OOV)) relative to the benchmark SCL system. For comparison, this model achieved a 96.8% in-domain accuracy on Sections 22–24 of the Penn Treebank, about 0.5 percentage point shy of a state-of-the-art in-domain system with more sophisticated supervised learning (Shen, Satta, and Joshi 2007). The BROWN-TOKEN-R representation, which Turian, Ratinov, and Bengio (2010) demonstrated performed as well or better than a variety of neural network statistical language models as representations, achieved accuracies between the SCL system and the HMM-TOKEN-R. The WEB1T-*n*-GRAM-R, I-HMM-TOKEN-R, and LATTICE-TOKEN-R all performed quite close to one another, but the I-HMM-TOKEN-R and LATTICE-TOKEN-R were trained on many orders of magnitude less text. The LSA-R and NB-R outperformed the TRAD-R baseline but not the SCL system. The *n*-GRAM-R, which was trained on the same text as the other representations except the WEB1T-*n*-GRAM-R, performed far worse than the WEB1T-*n*-GRAM-R.

Table 2
Learned representations, and especially latent-variable statistical language model representations, significantly outperform a traditional CRF system on domain adaptation for POS tagging. Percent error is shown for all words and out-of-vocabulary (OOV) words. The SCL+500bio system was given 500 labeled training sentences from the biomedical domain. 1.8% of tokens in the biomedical test set had POS tags like 'HYPHENATED', which are not part of the tagset for the training data, and were labeled incorrectly by all systems without access to labeled data from the biomedical domain. As a result, an error rate of 1.8 + 3.9 = 5.7 serves as a reasonable lower bound for a system that has never seen labeled examples from the biomedical domain.

Model	All words	OOV words
TRAD-R	11.7	32.7
n-GRAM-R	11.7	32.2
LSA-R	11.6	31.1
NB-R	11.6	30.7
ASO	11.6	29.1
SCL	11.1	28
BROWN-TOKEN-R	10.0	25.2
HMM-TOKEN-R	9.5	24.8
WEB1T-*n*-GRAM-R	6.9	24.4
I-HMM-TOKEN-R	6.7	24
LATTICE-TOKEN-R	**6.2**	**21.3**
SCL+500bio	3.9	–

The amount of unlabeled training data has a significant impact on the performance of these representations. This is apparent in the difference between WEB1T-*n*-GRAM-R and *n*-GRAM-R, but it is also true for our other representations. Figure 4 shows the accuracy of a representative subset of our taggers on words not seen in labeled training data, as we vary the amount of unlabeled training data available to the language

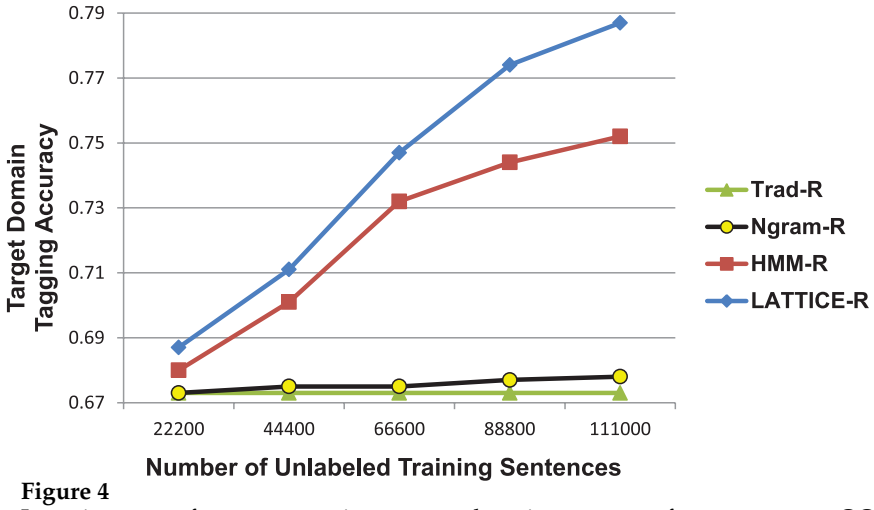


Figure 4
Learning curve for representations: target domain accuracy of our taggers on OOV words (not seen in *labeled* training data), as a function of the number of *unlabeled* examples given to the language models.

models. Performance grows steadily for all representations we measured, and none of the learning curves appears to have peaked. Furthermore, the margin between the more complex graphical models and the simpler *n*-gram models grows with increasing amounts of training data.

4.3.1 Sparsity and Polysemy. We expected that statistical language model representations would perform well in part because they provide meaningful features for sparse and polysemous words. For sparse tokens, these trends are already evident in the results in Table 2: Models that provide a constrained number of features, like HMM-based models, tend to outperform models that provide huge numbers of features (each of which, on average, is only sparsely observed in training data), like TRAD-R.

As for polysemy, HMM models significantly outperform naive Bayes models and the *n*-GRAM-R. The *n*-GRAM-R's features do not depend on a token type's context at all, and the NB-R's features depend only on the tokens immediately to the right and left of the current word. In contrast, the HMM takes into account all tokens in the surrounding sentence (although the strength of the dependence on more distant words decreases rapidly). Thus the performance of the HMM compared with *n*-GRAM-R and NB-R, as well as the performance of the LATTICE-TOKEN-R compared with the WEB1T-*n*-GRAM-R, suggests that representations that are sensitive to the context of a word produce better features.

To test these effects more rigorously, we selected 109 polysemous word types from our test data, along with 296 non-polysemous word types. The set of polysemous word types was selected by filtering for words in our labeled data that had at least two POS tags that began with distinct letters (e.g., VBZ and NNS). An initial set of non-polysemous word types was selected by filtering for types that appeared with just one POS tag. We then manually inspected these initial selections to remove obvious cases of word types that were in fact polysemous within a single part-of-speech, such as "bank." We further define sparse word types as those that appear five times or fewer in all of our unlabeled data, and we define non-sparse word types as those that appear at least 50 times in our unlabeled data. Table 3 shows our POS tagging results on the tokens of our labeled biomedical data with word types matching these four categories.

As expected, all of our statistical language models outperform the baseline by a larger margin on polysemous words than on non-polysemous words. The margin between graphical model representations and the WEB1T-*n*-GRAM-R model also increases on polysemous words, except for the NB-R. The WEB1T-*n*-GRAM-R uses none of the local context to decide which features to provide, and the NB-R uses only the immediate left and right context, so both models ignore most of the context. In contrast, the remaining graphical models use Viterbi decoding to take into account all tokens in the surrounding sentence, which helps to explain their relative improvement over WEB1T-*n*-GRAM-R on polysemous words.

The same behavior is evident for sparse words, as compared with non-sparse words: All of the statistical language model representations outperform the baseline by a larger margin on sparse words than not-sparse words, and all of the graphical models perform better relative to the WEB1T-*n*-GRAM-R on sparse words than not-sparse words. By reducing the feature space from millions of possible *n*-gram features to L categorical features, these models ensure that each of their features will be observed often in a reasonably sized training data set. Thus representations based

Table 3
Graphical models consistently outperform *n*-gram models by a larger margin on sparse words than not-sparse words, and by a larger margin on polysemous words than not-polysemous words. One exception is the NB-R, which performs worse relative to WEB1T-*n*-GRAM-R on polysemous words than non-polysemous words. For each graphical model representation, we show the difference in performance between that representation and WEB1T-*n*-GRAM-R in parentheses. For each representation, differences in accuracy on polysemous and non-polysemous subsets were statistically significant at $p < 0.01$ using a two-tailed Fisher's exact test. Likewise for performance on sparse vs. non-sparse categories.

	polysemous	not polysemous	sparse	not sparse
tokens	159	4,321	463	12,194
TRAD-R	59.5	78.5	52.5	89.6
WEB1T-*n*-GRAM-R	68.2	85.3	61.8	94.0
NB-R	64.5	88.7	57.8	89.4
(-WEB1T-*n*-GRAM-R)	(–3.7)	(**+3.4**)	(**–4.0**)	(–4.6)
HMM-TOKEN-R	67.9	83.4	60.2	91.6
(-WEB1T-*n*-GRAM-R)	(**–0.3**)	(–1.9)	(**–1.6**)	(–2.4)
I-HMM-TOKEN-R	75.6	85.2	62.9	94.5
(-WEB1T-*n*-GRAM-R)	(**+7.4**)	(–0.1)	(**+1.1**)	(+0.5)
LATTICE-TOKEN-R	70.5	86.9	65.2	94.6
(-WEB1T-*n*-GRAM-R)	(**+2.3**)	(+1.6)	(**+3.4**)	(+0.6)

on graphical models help address two key issues in building representations for POS tagging.

4.3.2 Domain Divergence. Besides sparsity and polysemy, Ben-David et al.'s (2007, 2010) theoretical analysis of domain adaptation shows that the distance between two domains under a representation R of the data is crucial for a good representation. We test their predictions using learned representations.

Ben-David et al.'s (2007, 2010) analysis depends on a particular notion of distance, the d_1 divergence, that is computationally intractable to calculate. For our analysis, we resort instead to two different computationally efficient approximations of this measure. The first uses a more standard notion of distance: the Jensen-Shannon Divergence (d_{JS}), a distance metric for probability distributions:

$$d_{JS}(p||q) = \frac{1}{2}\sum_i \left[p_i \log\left(\frac{p_i}{m_i}\right) + q_i \log\left(\frac{q_i}{m_i}\right)\right]$$

where $m_i = \frac{p_i+q_i}{2}$.

Intuitively, we aim to measure the distance between two domains by measuring whether features appear more commonly in one domain than in the other. For instance, the biomedical domain is far from the newswire domain under the TRAD-R representation because word-based features like *protein*, *gene*, and *pathway* appear far more commonly in the biomedical domain than the newswire domain. Likewise, *bank* and *president* appear far more commonly in newswire text. Since the d_1 distance is related to the optimal classifier for distinguishing two domains, it makes sense to measure the distance by comparing the frequencies of these features: a classifier can easily use the occurrence of words like *bank* and *protein* to accurately predict whether a given sentence belongs to the newswire or biomedical domain.

More formally, let S and T be two domains, and let f be a feature[5] in representation R—that is, a dimension of the image space of R. Let V be the set of possible values that f can take on. Let U_S be an unlabeled sample drawn from S, and likewise for U_T. We first compute the relative frequencies of the different values of f in $R(U_S)$ and $R(U_T)$, and then compute d_{JS} between these empirical distributions. Let p_f represent the empirical distribution over V estimated from observations of feature f in $R(U_S)$, and let q_f represent the same distribution estimated from $R(U_T)$.

Definition 1
JS domain divergence for a feature or $d_f(U_S, U_T)$ is the domain divergence between domains S and T under feature f from representation R, and is given by

$$d_f(U_S, U_T) = d_{JS}(p_f||q_f)$$

For a multidimensional representation, we compute the full domain divergence as a weighted sum over the domain divergences for its features. Because individual features may vary in their relevance to a sequence-labeling task, we use weights to indicate their importance to the overall distance between the domains. We set the weight w_f for feature f proportional to the $\mathcal{L}_1$ norm of CRF parameters related to f in the trained POS tagger. That is, let θ be the CRF parameters for our trained POS tagger, and let $\theta_f = \{\theta_{l,v}|l \text{ be the state for } z_i \text{ and } v \text{ be the value for } f\}$. We set $w_f = \frac{||\theta_f||_1}{||\theta||_1}$.

Definition 2
JS Domain Divergence or $d_R(U_S, U_T)$, is the distance between domains S and T under representation R, and is given by

$$d_R(U_S, U_T) = \sum_f w_f d_f(U_S, U_T)$$

Blitzer (2008) uses a different notion of domain divergence to approximate the d_1 divergence, which we also experimented with. He trains a CRF classifier on examples labeled with a tag indicating which domain the example was drawn from. We refer to this type of classifier as a **domain classifier**. Note that these should not be confused with our CRFs used for POS tagging, which take as input examples which are labeled with POS sequences. For the domain classifier, we tag every token from the WSJ domain as 0, and every token from the biomedical domain as 1. Blitzer then uses the accuracy of his domain classifier on a held-out test set as his measure of domain divergence. A high accuracy for the domain classifier indicates that the representation makes the two domains easy to separate, and thus high accuracy signifies a high domain divergence. To measure domain divergence using a domain classifier, we trained our representations on all of the unlabeled data for this task, as before. We then used 500 randomly sampled sentences from the WSJ domain, and 500 randomly sampled biomedical sentences, and labeled these with 0 for the WSJ data and 1 for the biomedical data. We measured the error rate of our domain-classifier CRF as the average error rate across folds when performing three-fold cross-validation on these 1,000 sentences.

5 For simplicity, the definition we provide here works only for discrete features, although it is possible to extend this definition to continuous-valued features.

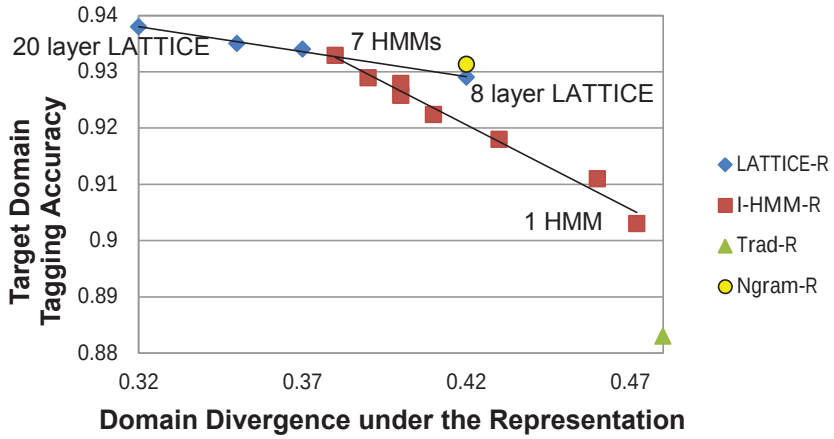


Figure 5
Target-domain POS tagging accuracy for a model developed using a representation *R* correlates strongly with lower JS domain divergence between WSJ and biomedical text under each representation *R*. The correlation coefficients r^2 for the linear regressions drawn in the figure are both greater than 0.97.

Figure 5 plots the accuracies and JS domain divergences for our POS taggers. Figure 6 shows the difference between target-domain error and source-domain error as a function of JS domain divergence. Figures 7 and 8 show the same information, except that the *x* axis plots the accuracy of a domain classifier as the way of measuring domain divergence. These results give empirical support to Ben-David et al.'s (2007, 2010) theoretical analysis: Smaller domain divergence—whether measured by JS domain divergence or by the accuracy of a domain classifier—correlates strongly with better target-domain accuracy. Furthermore, smaller domain divergence correlates strongly with a smaller difference in the accuracy of the taggers on the source and target domains.

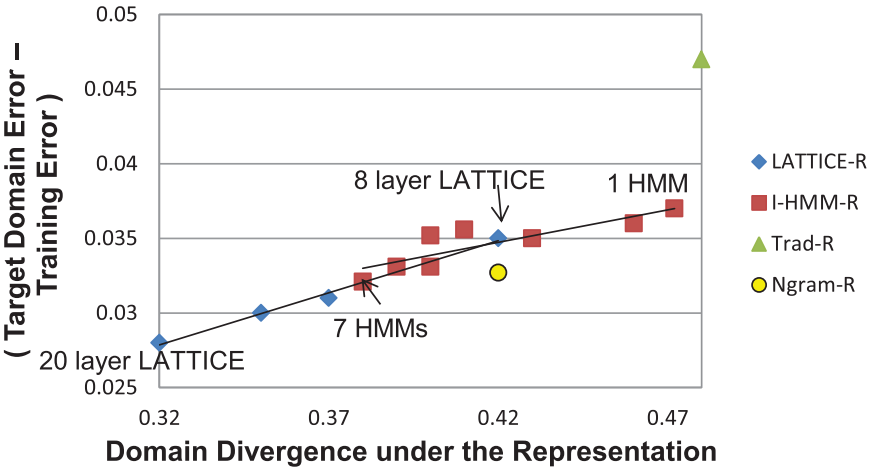


Figure 6
Smaller JS domain divergence correlates with a smaller difference between target-domain error and source-domain error.

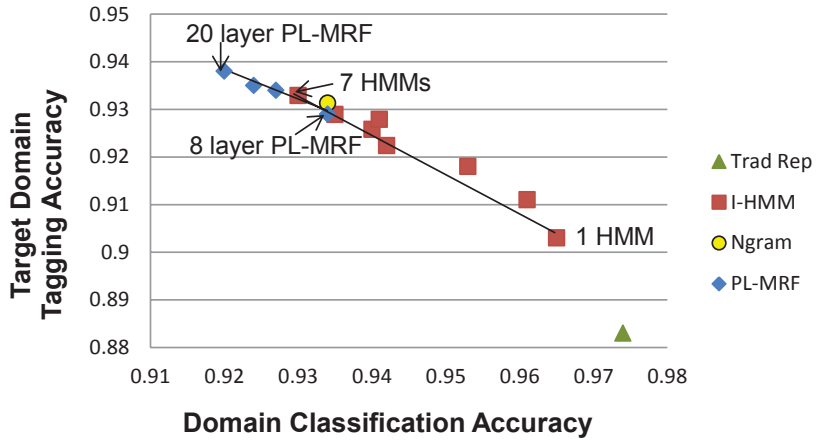


Figure 7
Target-domain tagging accuracy decreases with the accuracy of a CRF domain classifier. Intuitively, this means that training data from a source domain is less helpful for tagging in a target domain when source-domain data is easy to distinguish from target-domain data.

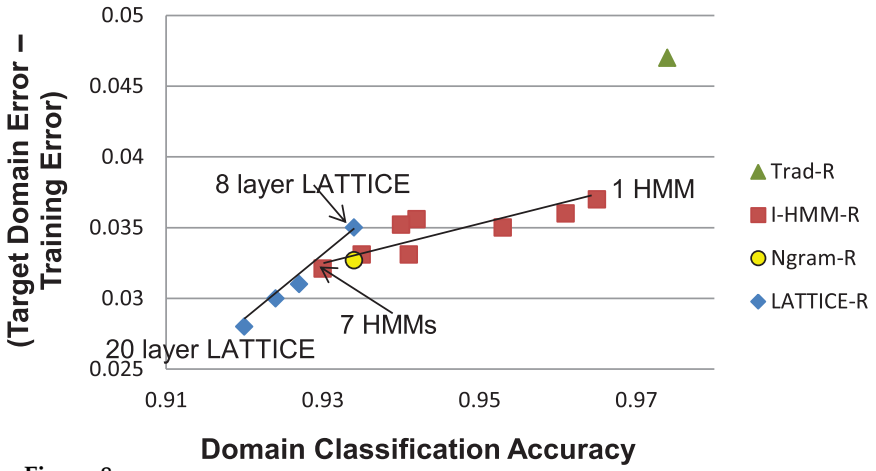


Figure 8
Better domain classification correlates with a larger difference between target-domain error and source-domain error.

Although both the JS domain divergence and the domain classifier provide only approximations of the d_1 metric for domain divergence, they agree very strongly: In both cases, the LATTICE-TOKEN-R representations had the lowest domain divergence, followed by the I-HMM-TOKEN-R representations, followed by TRAD-R, with n-GRAM-R somewhere between LATTICE-TOKEN-R and I-HMM-TOKEN-R. The main difference between the two metrics appears to be that the JS domain divergence gives a greater domain divergence to the eight-layer LATTICE-TOKEN-R model and the n-GRAM-R, placing them past the four- through eight-layer I-HMM-TOKEN-R representations. The domain classifier places these models closer to the other LATTICE-TOKEN-R representations, just past the seven-layer I-HMM-TOKEN-R representation.

The domain divergences of all models, using both techniques for measuring divergence, remain significantly far from zero, even under the best representation. As a result, there is ample room to experiment with even less-divergent representations of the two domains, to see if they might yield ever-increasing target-domain accuracies. Note that this is not simply a matter of adding more layers to the layered models. The I-HMM-TOKEN-R model performed best with seven layers, and the eight-layer representation had about the same accuracy and domain divergence as the five-layer model. This may be explained by the fact that the I-HMM layers are trained independently, and so additional layers may be duplicating other ones, and causing the supervised classifier to overfit. But it also shows that our current methodology has no built-in technique for constraining the domain divergence in our representations—the decrease in domain divergence from our more sophisticated representations is a coincidental byproduct of our training methodology, but there is no guarantee that our current mechanisms will continue to decrease domain divergence simply by increasing the number of layers. An important consideration for future research is to devise explicit learning mechanisms that guide representations towards smaller domain divergences.

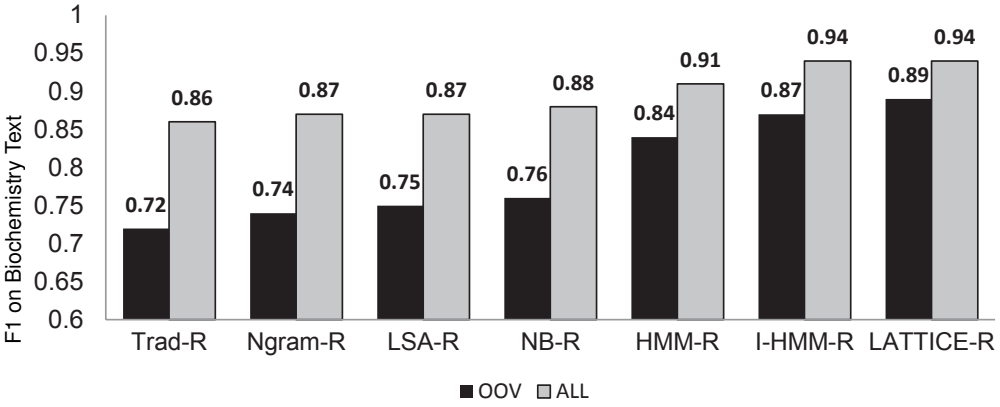
4.4 Domain Adaptation for Noun-Phrase Chunking and Chinese POS Tagging

We test the generality of our representations by using them for other tasks, domains, and languages. Here, we report on further sequence-labeling tasks in a domain adaptation setting: noun phrase chunking for adaptation from news text to biochemistry journals, and POS tagging in Mandarin for a variety of domains. In the next section, we describe the use of our representations in a weakly supervised information extraction task.

For chunking, the training set consists of the CoNLL 2000 shared task data for source-domain labeled data (Sections 15–18 of the WSJ portion of the Penn Treebank, labeled with chunk tags) (Tjong, Sang, and Buchholz 2000). For test data, we used biochemistry journal data from the Open American National Corpus[6] (OANC). One of the authors manually labeled 198 randomly selected sentences (5,361 tokens) from the OANC biochemistry text with noun-phrase chunk information.[7] We focus on noun phrase chunks because they are relatively easy to annotate manually, but contain a large variety of open-class words that vary from domain to domain. The labeled training set consists of 8,936 sentences and 211,726 tokens. Twenty-three percent of chunks in the test set begin with an OOV word (especially adjective-noun constructions like "aqueous formation" and "angular recess"), and 29% begin with a word seen at most twice in training data; we refer to these as OOV chunks and rare chunks. For our unlabeled data, we use 15,000 sentences (358,000 tokens; Sections 13–19) of the Penn Treebank and 45,000 sentences (1,083,000 tokens) from the OANC's biochemistry section. We tested TRAD-R (augmented with features for automatically generated POS tags), LSA-R, *n*-GRAM-R, NB-R, HMM-TOKEN-R, I-HMM-TOKEN-R (7 layers, which performed best for POS tagging) and LATTICE-TOKEN-R (20 layers) representations.

Figure 9 shows our NP chunking results for this domain adaptation task. The performance improvements for the HMM-based chunkers are impressive: LATTICE-TOKEN-R reduces error by 57% with respect to TRAD-R, and comes close to state-of-the-art results for chunking on newswire text. The results suggest that this representation allows the CRF to generalize almost as well to out-of-domain text as in-domain text.

6 Available from `http://www.anc.org/OANC/`.
7 The labeled data for this experiment are available from the first author's Web site.



Freq:	0		1		2		all	
Chunks:	284		39		39		1,258	
	R	P	R	P	R	P	R	P
TRAD-R	.74	.70	.85	.87	.79	.86	.86	.87
n-GRAM-R	.74	.74	.85	.85	.79	.86	.87	.87
LSA-R	.76	.74	.82	.83	.78	.85	.87	.88
NB-R	.73	.78	.86	.73	.86	.75	.88	.88
HMM-TOKEN-R	.80	.89	.92	.88	.92	.90	.91	.90
I-HMM-TOKEN-R	.90	.86	.92	.95	.87	.97	.95	.92
LATTICE-TOKEN-R	.92	.85	.94	.95	.87	.97	.95	.93

Figure 9
On biomedical journal data from the OANC, our best NP chunker outperforms the baseline CRF chunker by 0.17 F1 on chunks that begin with OOV words, and by 0.08 on all chunks. The table shows performance breakdowns (recall and precision) for chunks whose first word has frequency 0, 1, and 2 in training data, and the number of chunks in test data that fall into each of these categories.

Improvements are greatest on OOV and rare chunks, where LATTICE-TOKEN-R made absolute improvements over TRAD-R by 0.17 and 0.09 F1, respectively. Improvements for the single-layer HMM-TOKEN-R were smaller but still significant: 36% relative reduction in error overall, and 32% for OOV chunks.

The improved performance from our HMM-based chunker caused us to wonder how well the chunker could work without some of its other features. We removed all tag features and orthographic features and all features for word types that appear fewer than 20 times in training. This chunker still achieves 0.91 F1 on OANC data, and 0.93 F1 on WSJ data (Section 20), outperforming the TRAD-R system in both cases. It has only 20% as many features as the baseline chunker, greatly improving its training time. Thus these features are more valuable to the chunker than features from automatically produced tags and features for all but the most common words.

For Chinese POS tagging, we use text from the UCLA Corpus of Written Chinese (Tao and Xiao 2007), which is part of the Lancaster Corpus of Mandarin Chinese (LCMC). The UCLA Corpus consists of 11,192 sentences of word-segmented and POS-tagged text in 13 genres (see Table 4). We use gold-standard word segmentation labels for training and testing. The LCMC tagset consists of 50 Chinese POS tags. On average, each genre contains 5,284 word tokens, for a total of 68,695 tokens among all genres. We use the 'news' genre as our source domain, which we use for training and development

Table 4
POS tagging accuracy: the LATTICE-TOKEN-R and other graphical model representations outperform TRAD-R and state-of-the-art Chinese POS taggers on all target domains. For target domains, * indicates the performance is statistically significantly better than the Stanford and TRAD-R baselines at $p < 0.05$, using a two-tailed χ^2 test; ** indicates significance at $p < 0.01$. On the news domain, the Stanford tagger is significantly different from all other systems using a two-tailed χ^2 test with $p < 0.01$.

Domain	Stanford	TRAD	NGR	LSA	NB	HMM	I-H	LAT
lore	88.4	84.0	84.2	85.3	85.3	89.7	89.9	**90.1***
religion	83.5	79.1	79.4	79.8	80.0	85.2	85.6	**85.9***
humour	89.0	84.2	84.5	86.2	86.8	89.6	89.6	**89.9***
general-fic	87.5	84.5	85.0	85.3	85.7	89.4	89.7	**89.9***
essay	88.4	83.2	83.7	84.0	84.3	89.0	89.1	**90.1***
mystery	87.4	82.4	83.4	84.3	85.3	90.1	91.1	**91.3****
romance	87.5	84.2	84.5	85.3	86.1	89.0	89.5	**89.8****
science-fic	88.6	82.1	82.5	83.0	83.0	87.0	88.3	**88.6**
skills	82.7	77.3	77.7	78.2	78.4	84.9	85.0	**85.1****
science	86.0	82.0	82.3	82.4	82.4	87.8	87.8	**87.9***
adventure-fic	82.1	74.3	75.2	76.1	77.8	81.7	82.0	**82.2**
report	91.7	84.2	85.1	85.3	86.1	**91.9**	**91.9**	**91.9**
news	**98.8****	96.9	92.3	93.4	94.3	94.2	97.0	97.1
all but news	87.0	81.2	82.0	82.8	83.6	88.1	88.4	**88.8****
all domains	88.7	83.2	83.6	84.4	85.5	89.5	89.7	**90.0****

data. For test data, we randomly select 20% of every other genre. For our unlabeled data, we use all of the 'news' text, plus the remaining 80% of the texts from the other genres. As before, we replace hapax legomena in the unlabeled data with the special symbol `*UNKNOWN*`, and do the same for word types in the labeled test sets that never appear in our unlabeled training texts. We compare against a state-of-the-art Chinese POS tagger for in-domain text, the CRF-based Stanford tagger (Tseng, Jurafsky, and Manning 2005). We obtained the code for this tagger,[8] and retrained it on our training data set.

The Chinese POS tagging results are shown in Table 4. The LATTICE-TOKEN-R outperforms the state-of-the-art Stanford tagger on all target domains. Overall, on all out-of-domain tests, LATTICE-TOKEN-R provides a relative reduction in error of 13.8% compared with the Stanford tagger. The best performance is on the 'mystery' domain, where the LATTICE-TOKEN-R model reaches 91.3% accuracy, a 3.9 percentage points improvement over the Stanford tagger. Its performance on the in-domain 'news' test set is significantly worse (1.7 percentage points) than the Stanford tagger, suggesting that the Stanford tagger relies on domain-dependent features that are helpful for tagging news, but not for tagging in general. The LATTICE-TOKEN-R's accuracy is still significantly worse on out-of-domain text than in-domain text, but the gap between the two (8.3 percentage points) is better than the gap for the Stanford tagger (11.8 percentage points). We believe that the lower out-of-domain performance of our Chinese POS tagger, compared with our English POS tagger and our chunker, was at least in part due to having far less unlabeled text available for this task.

8 Available at `http://nlp.stanford.edu/software/tagger.shtml`.

5. Information Extraction Experiments

In this section, we evaluate our learned representations on their ability to capture semantic, rather than syntactic, information. Specifically, we investigate a *set-expansion* task in which we're given a corpus and a few "seed" noun phrases from a semantic category (e.g., Superheroes), and our goal is to identify other examples of the category in the corpus. This is a different type of weakly supervised task from the earlier domain adaptation tasks because we are given only a handful of positive examples from a category, rather than a large sample of positively and negatively labeled training examples from a separate domain.

Existing set-expansion techniques utilize the distributional hypothesis: Candidate noun phrases for a given semantic class are ranked based on how similar their contextual distributions are to those of the seeds. Here, we measure how performance on the set-expansion task varies when we employ different representations for the contextual distributions.

5.1 Methods

The set-expansion task we address is formalized as follows. Given a corpus, a set of seeds from some semantic category C, and a separate set of candidate phrases P, output a ranking of the phrases in P in decreasing order of likelihood of membership in the semantic category C.

For any given representation R, the set-expansion algorithm we investigate is straightforward: We rank candidate phrases in increasing order of the distance between their feature vectors and those of the seeds. The particular distance metrics utilized are detailed subsequently.

Because set expansion is performed at the level of word types rather than tokens, it requires type-based representations. We compare HMM-TYPE-R, n-GRAM-R, LATTICE-TYPE-R, and BROWN-TYPE-R in this experiment. We used a 25-state HMM, and the LATTICE-TYPE-R as described in the previous section. Following previous set-expansion experiments with n-grams (Ahuja and Downey 2010), we use a trigram model with Kneser-Ney smoothing for n-GRAM-R.

The distances between the candidate phrases and the seeds for HMM-TYPE-R, n-GRAM-R, and LATTICE-TYPE-R representations are calculated by first creating a prototypical "seed feature vector" equal to the mean of the feature vectors for each of the seeds in the given representation. Then, we rank candidate phrases in order of increasing distance between their feature vector and the seed feature vector. As a distance measure between vectors (in this case, probability distributions), we compute the average of five standard distance measures, including KL and JS divergence, and cosine, Euclidean, and L1 distance. In experiments, we found that improving upon this simple averaging was not easy—in fact, tuning a weighted average of the distance measures for each representation did not improve results significantly on held-out data.

For Brown clusters, we use prefixes of all possible lengths as features. We define the similarity between two Brown representation feature vectors to be the number of features they share in common (this is equivalent to the length of the longest common prefix between the two original Brown cluster labels). The candidate phrases are then ranked in decreasing order of the sum of their similarity scores to each of the seeds. We experimented with normalizing the similarity scores by the longer of the two vector lengths, and found this to decrease results slightly. We use unnormalized (integer) similarity scores for Brown clusters in our experiments.

5.2 Data Sets

We utilized a set of approximately 100,000 sentences of Web text, joining multi-word named entities in the corpus into single tokens using the Lex algorithm (Downey, Broadhead, and Etzioni 2007). This process enables each named entity (the focus of the set-expansion experiments) to be treated as a single token, with a single representation vector for comparison. We developed all word type representations using this corpus.

To obtain examples of multiple semantic categories, we utilized selected Wikipedia "listOf" pages from Pantel et al. (2009) and augmented these with our own manually defined categories, such that each list contained at least ten distinct examples occurring in our corpus. In all, we had 432 examples across 16 distinct categories such as Countries, Greek Islands, and Police TV Dramas.

5.3 Results

For each semantic category, we tested five different random selections of five seed examples, treating the unselected members of the category as positive examples, and all other candidate phrases as negative examples. We evaluate using the area under the precision-recall curve (AUC) metric.

The results are shown in Table 5. All representations improve performance over a random baseline, equal to the average AUC over five random orderings for each category, and the graphical models outperform the *n*-gram representation. I-HMM-TYPE-R and Brown clustering in the particular case of 1,000 clusters perform best, with HMM-TYPE-R performing nearly as well. Brown clusters give somewhat lower results as the number of clusters varies.

As with POS tagging, we expect that language model representations improve performance on the IE task by providing informative features for sparse word types. However, because the IE task classifies word types rather than tokens, we expect the representations to provide less benefit for polysemous word types. To test these hypotheses, we measured how IE performance changed in sparse or polysemous settings. We identified polysemous categories as those for which fewer than 90% of the category members had the category as a clear dominant sense (estimated manually); other categories were considered non-polysemous. Categories whose members had a median number of occurrences in the corpus of less than 30 were deemed sparse, and others non-sparse.

Table 5
I-HMM-TYPE-R outperforms the other methods, improving performance over a random baseline by twice as much as either *n*-GRAM-R or LATTICE-TYPE-R.

model	AUC
I-HMM-TYPE-R	**0.18**
HMM-TYPE-R	0.17
BROWN-TYPE-R-3200	0.16
BROWN-TYPE-R-1000	**0.18**
BROWN-TYPE-R-320	0.15
BROWN-TYPE-R-100	0.13
LATTICE-TYPE-R	0.11
n-GRAM-R baseline	0.10
Random baseline	0.10

Table 6
Graphical models as representations for IE consistently perform better relative to *n*-gram models on sparse words, but not necessarily polysemous words.

	polysemous	not-polysemous	sparse	not-sparse
types	222	210	266	166
categs.	12	4	13	3
n-GRAM-R	0.07	0.17	0.06	0.25
LATTICE-TYPE-R	0.09	0.15	0.1	0.19
-*n*-GRAM-R	+0.02	−0.02	+0.04	−0.06
HMM-TYPE-R	0.14	0.26	0.15	0.32
-*n*-GRAM-R	+0.07	+0.09	+0.09	+0.07

IE performance on these subsets of the data are shown in Table 6. Both graphical model representations outperform the *n*-gram representation more on sparse words, as expected. For polysemy, the picture is mixed: The LATTICE-TYPE-R outperforms *n*-GRAM-R on polysemous categories, whereas HMM-TYPE-R's performance advantage over *n*-GRAM-R decreases.

One surprise on the IE task is that the LATTICE-TYPE-R performs significantly less well than the HMM-TYPE-R, whereas the reverse is true on POS tagging. We suspect that the difference is due to the issue of classifying types vs. tokens. Because of their more complex structure, PL-MRFs tend to depend more on transition parameters than do HMMs. Furthermore, our decision to train the PL-MRFs using contrastive estimation with a neighborhood that swaps consecutive pairs of words also tends to emphasize transition parameters. As a result, we believe the posterior distribution over latent states given a word type is more informative in our HMM model than the PL-MRF model. We measured the entropy of these distributions for the two models, and found that $H(P_{\text{PL-MRF}}(\mathbf{y}|x = w)) = 9.95$ bits, compared with $H(P_{\text{HMM}}(\mathbf{y}|x = w)) = 2.74$ bits, which supports the hypothesis that the drop in the PL-MRF's performance on IE is due to its dependence on transition parameters. Further experiments are warranted to investigate this issue.

5.4 Testing the Language Model Representation Hypothesis in IE

The language model representation hypothesis (Section 2) suggests that all else being equal, more accurate language models will provide features that lead to better performance on NLP tasks. Here, we test this hypothesis on the set expansion IE task.

Figures 10 and 11 show how the performance of the HMM-TYPE-R varies with the language modeling accuracy of the underlying HMM. Language modeling accuracy is measured in terms of perplexity on held-out text. Here, we use set expansion data sets from previous work (Ahuja and Downey 2010). The first two are composed of extractions from the TextRunner information extraction system (Banko et al. 2007) and are denoted as Unary (361 examples) and Binary (265 examples). The second, Wikipedia (2,264 examples), is a sample of Wikipedia concept names. We evaluate the performance of several different trained HMMs with numbers of latent states K ranging from 5 to 1,600 (to help illustrate how IE and LM performance varies even when model capacity is fixed, we include three distinct models with $K = 100$ states trained separately over the full corpus). We used a distributed implementation of HMM training and corpus

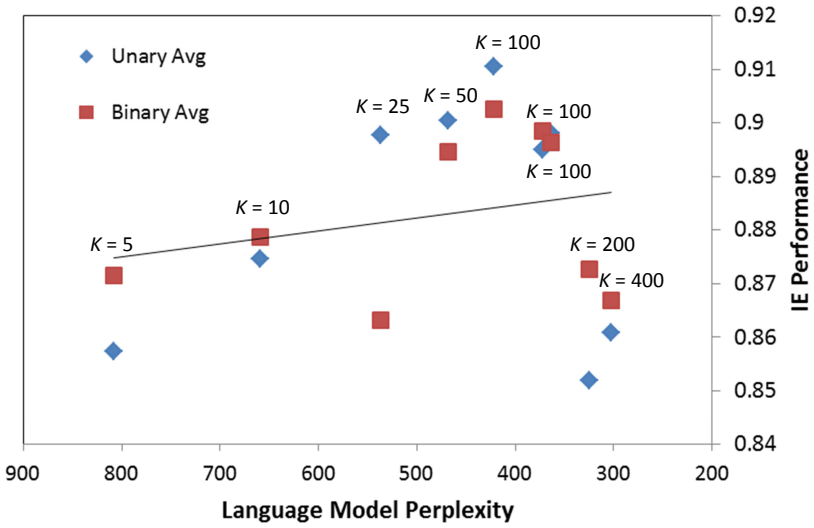


Figure 10
Information extraction (IE) performance of HMM-TYPE-R as the language modeling accuracy of the HMM varies, on TextRunner data sets. IE accuracy (in terms of area under the precision-recall curve) tends to increase as language modeling accuracy improves (i.e., perplexity decreases).

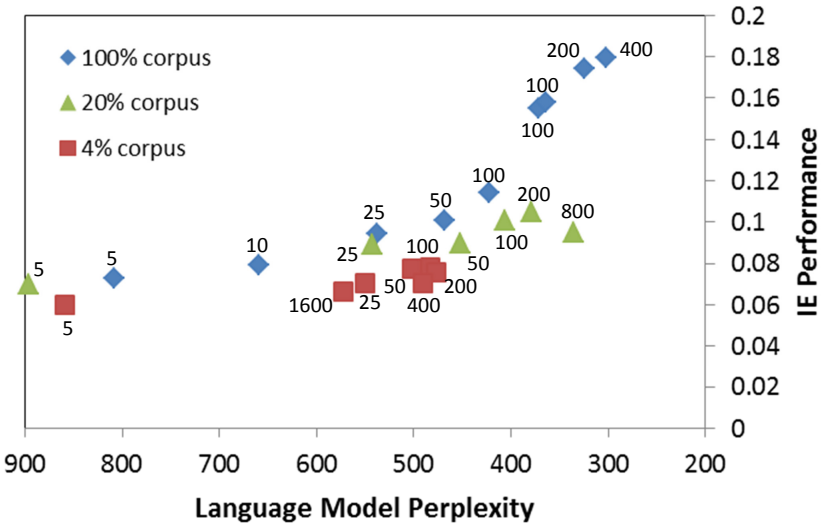


Figure 11
Information extraction (IE) performance of HMM-TYPE-R as the language modeling accuracy of the HMM varies on the Wikipedia data set. Number labels indicate the number of latent states *K*, and performance is shown for three training corpus sizes (the full corpus consists of approximately 60 million tokens). IE accuracy (in terms of area under the precision-recall curve) tends to increase as language modeling accuracy improves (i.e., perplexity decreases).

partitioning techniques (Yang, Yates, and Downey 2013) to enable training of our larger capacity HMM models on large data sets.

The results provide support for the language model representation hypothesis, showing that IE performance does tend to improve as language model perplexity decreases. On the smaller Unary and Binary sets (Figure 10), although IE accuracy

does decrease for the lowest-perplexity models, overall language model perplexity exhibits a negative correlation with IE area under the precision-recall curve (the Pearson correlation coefficient is −0.18 for Unary, and −0.28 for Binary). For Wikipedia (Figure 11), the trend is more consistent, with IE performance increasing monotonically as perplexity decreases for models trained on the full training corpus (the Pearson correlation coefficient is −0.90).

Figure 11 also illustrates how LM and IE performance changes as the amount of training text varies. In general, increasing the training corpus size increases IE performance and decreases perplexity. Over all data points in the figure, IE performance correlates most strongly with model perplexity (−0.68 Pearson correlation, −0.88 Spearman correlation), followed by corpus size (0.66, 0.71) and model capacity (−0.05, 0.38). The small negative Pearson correlation between model capacity and IE performance is primarily due to the model with 1,600 states trained on 4% of the corpus. This model has a large parameter space and sparse training data, and thus suffers from overfitting in terms of both model perplexity and IE performance. If we ignore this overfit model, the Pearson correlation between model capacity and IE performance for the other models in the Figure is 0.24.

Our results show that IE based on distributional similarity tends to improve as the quality of the latent variable model used to measure distributional similarity improves. A similar trend was exhibited in our previous work (Ahuja and Downey 2010); here, we extend the previous results to models with more latent states and a larger, more reliable test set (Wikipedia). The results suggest that scaling up the training of latent variable models to utilize larger training corpora and more latent states may be a promising direction for improving IE capabilities.

6. Conclusion and Future Work

Our study of representation learning demonstrates that by using statistical language models to aggregate information across many unannotated examples, it is possible to find accurate distributional representations that can provide highly informative features to weakly supervised sequence labelers and named-entity classifiers. For both domain adaptation and weakly supervised set expansion, our results indicate that graphical models outperform *n*-gram models as representations, in part for their greater ability to handle sparsity and polysemy. Our IE task provides important evidence to support the Language Model Representation Hypothesis, showing that the AUC of the IE system correlates more with language model perplexity than the size of the training data or the capacity of the language model. Finally, our sequence labeling experiments provide empirical evidence in support of theoretical work on domain adaptation, showing that target-domain tagging accuracy is highly correlated with two different measures of domain divergence.

Representation learning remains a promising area for finding further improvements in various NLP tasks. The representations we have described are trained in an unsupervised fashion, so a natural extension is to investigate supervised or semi-supervised representation-learning techniques. As mentioned previously, our current techniques have no built-in methods for enforcing that they provide similar features in different domains; devising a mechanism that enforces this could allow for less domain-divergent and potentially more accurate representations. We have considered sequence labeling, but another promising direction is to apply these techniques to more complex structured prediction tasks, like parsing or relation extraction. Our current approach to sequence labeling requires retraining of a CRF for every new domain; incremental

retraining techniques for new domains would speed up the process. Finally, models that combine our representation learning approach with instance weighting and other forms of supervised domain adaptation may take better advantage of labeled data in target domains, when it is available.

Acknowledgments

This material is based on work supported by the National Science Foundation under grant no. IIS-1065397.

References

Ahuja, Arun and Doug Downey. 2010. Improved extraction assessment through better language models. In *Proceedings of the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*, pages 225–228, Los Angeles, CA.

Ando, Rie Kubota and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the ACL*, pages 1–9, Ann Arbor, MI.

Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the IJCAI*, pages 2670–2676, Hyderabad.

Banko, Michele and Robert C. Moore. 2004. Part of speech tagging in context. In *Proceedings of the COLING*, pages 556–561, Geneva.

Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. 2010. A theory of learning from different domains. *Machine Learning*, 79:151–175.

Ben-David, Shai, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 20*, pages 127–144, Vancouver.

Bengio, Yoshua. 2008. Neural net language models. *Scholarpedia*, 3(1):3,881.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1,137–1,155.

Bengio, Yoshua, Jerome Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 41–48, Montreal.

Bikel, Daniel M. 2004a. A distributional analysis of a lexicalized statistical parsing model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 182–189, Barcelona.

Bikel, Daniel M. 2004b. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1,022.

Blitzer, John. 2008. *Domain Adaptation of Natural Language Processing Systems*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Blitzer, John, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. 2007. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 129–136, Vancouver.

Blitzer, John, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*, pages 40–47, Prague.

Blitzer, John, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the EMNLP*, pages 120–128, Sydney.

Bodlaender, Hans L. 1988. Dynamic programming on graphs with bounded treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, pages 105–118, Tampere.

Brants, Thorsten and Alex Franz. 2006. Web 1t 5-gram version 1. www.ldc.upenn.edu/catalog/.

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18:467–479.

Candito, Marie and Benoit Crabbe. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the IWPT*, pages 138–141, Paris.

Chan, Yee Seng and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 89–96, Sydney.

Chelba, Ciprian and Alex Acero. 2004. Adaptation of maximum entropy classifier: Little data can help a lot. In *Proceedings of the EMNLP*, pages 285–292, Barcelona.

Collobert, Robert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 160–167, Helsinki.

Dai, Wenyuan, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Transferring naive Bayes classifiers for text classification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 540–545, Vancouver.

Darroch, J. N., S. L. Lauritzen, and T. P. Speed. 1980. Markov fields and log-linear interaction models for contingency tables. *The Annals of Statistics*, 8(3):522–539.

Daumé III, Hal. 2007. Frustratingly easy domain adaptation. In *Proceedings of the ACL*, pages 256–263, Prague.

Daumé III, Hal, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the ACL Workshop on Domain Adaptation (DANLP)*, pages 53–59, Uppsala.

Daumé III, Hal and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

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

Dempster, Arthur, Nan Laird, and Donald Rubin. 1977. Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Deschacht, Koen and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 21–29, Singapore.

Dhillon, Paramveer S., Dean Foster, and Lyle Ungar. 2011. Multi-View Learning of Word Embeddings via CCA. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 886–874, Granada.

Downey, Doug, Matthew Broadhead, and Oren Etzioni. 2007. Locating complex named entities in web text. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2,733–2,739, Hyderabad.

Downey, Doug, Stefan Schoenmackers, and Oren Etzioni. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *Proceedings of the ACL*, pages 696–703, Prague.

Dredze, Mark and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of EMNLP*, pages 689–697, Honolulu, HI.

Dredze, Mark, Alex Kulesza, and Koby Crammer. 2010. Multi-domain learning by confidence weighted parameter combination. *Machine Learning*, 79:123–149.

Finkel, Jenny Rose and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *Proceedings of HLT-NAACL*, pages 602–610, Boulder, CO.

Fürstenau, Hagen and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 220–228, Athens.

Ghahramani, Zoubin and Michael I. Jordan. 1997. Factorial hidden Markov models. *Machine Learning*, 29(2-3):245–273.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *Conference on Empirical Methods in Natural Language Processing*, pages 167–202, Pittsburgh, PA.

Goldwater, Sharon and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751, Prague.

Graça, João V., Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs. parameter sparsity in latent variable models. In *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, pages 664–672, Vancouver.

Harris, Z. 1954. Distributional structure. *Word*, 10(23):146–162.

Hindle, Donald. 1990. Noun classification from predicage-argument structures. In *Proceedings of the ACL*, pages 268–275, Pittsburgh, PA.

Honkela, Timo. 1997. Self-organizing maps of words for natural language processing applications. In *Proceedings of the International ICSC Symposium on Soft Computing*, pages 401–407, Millet, Alberta.

Huang, Fei and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 495–503, Singapore.

Huang, Fei and Alexander Yates. 2010. Exploring representation-learning approaches to domain adaptation. In *Proceedings of the ACL 2010 Workshop on Domain Adaptation for Natural Language Processing (DANLP)*, pages 23–30, Uppsala.

Huang, Fei, Alexander Yates, Arun Ahuja, and Doug Downey. 2011. Language models as representations for weakly supervised NLP tasks. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 125–134, Portland, OR.

Jiang, Jing and ChengXiang Zhai. 2007a. Instance weighting for domain adaptation in NLP. In *Proceedings of ACL*, pages 264–271, Prague.

Jiang, Jing and ChengXiang Zhai. 2007b. A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, pages 401–410, Lisbon.

Johnson, Mark. 2007. Why doesn't EM find good HMM POS-taggers. In *Proceedings of the EMNLP*, pages 296–305, Prague.

Kaski, S. 1998. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the IJCNN*, pages 413–418, Washington, DC.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 595–603, Columbus, OH.

Lin, Dekang and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the ACL-IJCNLP*, pages 1,030–1,038, Singapore.

Liu, Dong C. and Jorge Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

Mansour, Y., M. Mohri, and A. Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1,041–1,048, Vancouver.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Martin, Sven, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24:19–37.

McClosky, David. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Parsing*. Ph.D. thesis, Brown University, Providence, RI.

McClosky, David, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies 2010 Conference (NAACL-HLT 2010)*, pages 28–36, Los Angeles, CA.

Miller, Scott, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 337–342, Boston, MA.

Mnih, Andriy and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, Corvallis, OR.

Mnih, Andriy and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 1,081–1,088, Vancouver.

Mnih, Andriy, Zhang Yuecheng, and Geoffrey Hinton. 2009. Improving a statistical language model through non-linear prediction. *Neurocomputing*, 72(7-9):1414–1418.

Morin, Frederic and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252, Barbados.

Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the EMNLP*, pages 938–947, Singapore.

PennBioIE. 2005. Mining the bibliome project. http://bioie.ldc.upenn.edu/.
Pereira, Fernando, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 183–190, Columbus, OH.
Pradhan, Sameer, Wayne Ward, and James H. Martin. 2007. Towards robust semantic role labeling. In *Proceedings of NAACL-HLT*, pages 556–563, Rochester, NY.
Rabiner, Lawrence R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.
Raina, Rajat, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 759–766, Corvallis, OR.
Ratinov, Lev and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 147–155, Boulder, CO.
Ritter, H. and T. Kohonen. 1989. Self-organizing semantic maps. *Biological Cybernetics*, 61(4):241–254.
Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Synactic Theory: A Formal Introduction*. CSLI Publications, Stanford, CA, second edition.
Sahlgren, Magnus. 2001. Vector-based semantic analysis: Representing word meanings based on random labels. In *Proceedings of the Semantic Knowledge Acquisition and Categorization Workshop*, pages 1–12, Helsinki.
Sahlgren, Magnus. 2005. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*, 87:1–9.
Sahlgren, Magnus. 2006. *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.
Salton, Gerard and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.
Satpal, Sandeep and Sunita Sarawagi. 2007. Domain adaptation of conditional probability models via feature subsetting. In *Proceedings of ECML/PKDD*, pages 224–235, Warsaw.
Sekine, Satoshi. 1997. The domain dependence of parsing. In *Proceedings of Applied Natural Language Processing (ANLP)*, pages 96–102, Washington, DC.
Shen, Libin, Giorgio Satta, and Aravind K. Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the ACL*, pages 760–767, Prague.
Smith, Noah A. and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 354–362, Ann Arbor, MI.
Sutton, Charles, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723.
Suzuki, Jun and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-HLT)*, pages 665–673, Columbus, OH.
Suzuki, Jun, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the EMNLP*, pages 551–560, Singapore.
Tao, Hongyin and Richard Xiao. 2007. The UCLA Chinese corpus. UCREL. www.lancaster.ac.uk/fass/projects/corpus/UCLA/.
Tjong, Erik F., Kim Sang, and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on Computational Natural Language Learning*, pages 127–132, Lisbon.
Toutanova, Kristina and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of the NIPS*, pages 1,521–1,528, Vancouver.
Tseng, Huihsin, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help POS tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN Workshop*, pages 32–39, Jeju Island.

Turian, Joseph, James Bergstra, and Yoshua Bengio. 2009. Quadratic features and deep architectures for chunking. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 245–248, Boulder, CO.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 384–394, Uppsala.

Turney, Peter D. and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Ushioda, Akira. 1996. Hierarchical clustering of words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1,159–1,162, Copenhagen.

Väyrynen, Jaakko and Timo Honkela. 2004. Word category maps based on emergent features created by ICA. In *Proceedings of the STePs 2004 Cognition + Cybernetics Symposium*, pages 173–185, Tikkurila.

Väyrynen, Jaakko and Timo Honkela. 2005. Comparison of independent component analysis and singular value decomposition in word context analysis. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR)*, pages 135–140, Espoo.

Väyrynen, Jaakko, Timo Honkela, and Lasse Lindqvist. 2007. Towards explicit semantic features using independent component analysis. In *Proceedings of the Workshop Semantic Content Acquisition and Representation (SCAR)*, pages 20–27, Stockholm.

Weston, Jason, Frederic Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1,168–1,175, Helsinki.

Yang, Yi, Alexander Yates, and Doug Downey. 2013. Overcoming the memory bottleneck in distributed training of latent variable models of text. In *Proceedings of the NAACL-HLT*, pages 579–584, Atlanta, GA.

Zhao, Hai, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the CoNLL 2009 Shared Task*, pages 55–60, Boulder, CO.