Weakly-Supervised Learning with Cost-Augmented Contrastive Estimation

Kevin Gimpel **Mohit Bansal**
Toyota Technological Institute at Chicago, IL 60637, USA
{kgimpel,mbansal}@ttic.edu

Abstract

We generalize contrastive estimation in two ways that permit adding more knowledge to unsupervised learning. The first allows the modeler to specify not only the *set* of corrupted inputs for each observation, but also *how bad* each one is. The second allows specifying structural preferences on the latent variable used to explain the observations. They require setting additional hyperparameters, which can be problematic in unsupervised learning, so we investigate new methods for unsupervised model selection and system combination. We instantiate these ideas for part-of-speech induction without tag dictionaries, improving over contrastive estimation as well as strong benchmarks from the PASCAL 2012 shared task.

1 Introduction

Unsupervised NLP aims to discover useful structure in unannotated text. This structure might be part-of-speech (POS) tag sequences (Merialdo, 1994), morphological segmentation (Creutz and Lagus, 2005), or syntactic structure (Klein and Manning, 2004), among others. Unsupervised systems typically improve when researchers incorporate knowledge to bias learning to capture characteristics of the desired structure.[1]

There are many successful examples of adding knowledge to improve learning without labeled examples, including: sparsity in POS tag distributions (Johnson, 2007; Ravi and Knight, 2009; Ganchev et al., 2010), short attachments for dependency parsing (Smith and Eisner, 2006), agreement of word alignment models (Liang et al., 2006), power law effects in lexical distributions (Blunsom and Cohn, 2010; Blunsom and Cohn, 2011), multilingual constraints (Smith and Eisner, 2009; Ganchev et al., 2009; Snyder et al., 2009; Das and Petrov, 2011), and orthographic cues (Spitkovsky et al., 2010c; Spitkovsky et al., 2011b), *inter alia*.

Contrastive estimation (CE; Smith and Eisner, 2005) is a general approach to weakly-supervised learning with a particular way of incorporating knowledge. CE increases the likelihood of the observations at the expense of those in a particular **neighborhood** of each observation. The neighborhood typically contains corrupted versions of the observations. The latent structure is marginalized out for both the observations and their corruptions; the intent is to learn latent structure that helps to explain why the observation was generated rather than any of the corrupted alternatives.

In this paper, we present a new objective function for weakly-supervised learning that generalizes CE by including two types of **cost functions**, one on observations and one on output structures. The first (§4) allows us to specify not only the set of corrupted observations, but also *how bad* each corruption was. We use n-gram language models to measure the severity of each corruption.

The second (§5) allows us to specify preferences on desired output structures, regardless of the input sentence. For POS tagging, we attempt to learn language-independent tag frequencies by computing counts from treebanks for 11 languages not used in our POS induction experiments. For example, we encourage tag sequences that contain adjacent nouns and penalize those that contain adjacent adpositions.

We consider several unsupervised ways to set hyperparameters for these cost functions (§7), including the recently-proposed log-likelihood estimator of Bengio et al. (2013). We also circumvent

[1] We note that doing so strains the definition of the term *unsupervised*. Hence we will use the term *weakly-supervised* to refer to methods that do not explicitly train on labeled examples for the task of interest, but do use some form of task-specific knowledge.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1329–1341,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

hyperparameter selection via system combination, developing a novel voting scheme for POS induction that aligns tag identifiers across runs.

We evaluate our approach, which we call **cost-augmented contrastive estimation (CCE)**, on POS induction without tag dictionaries for five languages from the PASCAL shared task (Gelling et al., 2012). We find that CCE improves over both standard CE as well as strong baselines from the shared task. In particular, our final average accuracies are better than all entries in the shared task that use the same number of tags.

2 Related Work

Weakly-supervised techniques can be roughly categorized in terms of whether they influence the model, the learning procedure, or explicitly target the output structure. Examples abound in NLP; we focus on those that have been applied to POS tagging.

There have been many efforts at biasing models, including features (Smith and Eisner, 2005a; Berg-Kirkpatrick et al., 2010), sparse priors (Johnson, 2007; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2007), sparsity in tag transition distributions (Ravi and Knight, 2009), small models via minimum description length criteria (Vaswani et al., 2010; Poon et al., 2009), a one-tag-per-type constraint (Blunsom and Cohn, 2011), and power law effects via Bayesian nonparametrics (Van Gael et al., 2009; Blunsom and Cohn, 2010; Blunsom and Cohn, 2011).

We focus below on efforts that induce bias into the learning (§2.1) or more directly in the output structure (§2.2), as they are more closely related to our contributions in this paper.

2.1 Biasing Learning

Some unsupervised methods do not change the model or attempt to impose structural bias; rather, they change the learning. This may involve optimizing a different objective function for the same model, e.g., by switching from soft to hard EM (Spitkovsky et al., 2010b). Or it may involve changing the objective during learning via annealing (Smith and Eisner, 2004) or more general multi-objective techniques (Spitkovsky et al., 2011a; Spitkovsky et al., 2013).

Other learning modifications relate to automatic data selection, e.g., choosing examples for generative learning (Spitkovsky et al., 2010a) or automatically generating negative examples for discriminative unsupervised learning (Li et al., 2010; Xiao et al., 2011).

CE does both, automatically generating negative examples and changing the objective function to include them. Our observation cost function alters CE's objective function, sharpening the effective distribution of the negative examples.

2.2 Structural Bias

Our output cost function is used to directly specify preferences on desired output structures. Several others have had similar aims. For dependency grammar induction, Smith and Eisner (2006) favored short attachments using a fixed-weight feature whose weight was optionally annealed during learning. Their bias could be implemented as an output cost function in our framework.

Posterior regularization (PR; Ganchev et al., 2010) is a general framework for declaratively specifying preferences on model outputs. Naseem et al. (2010) proposed universal syntactic rules for unsupervised dependency parsing and used them in a PR regime; we use analogous universal tag sequences in our cost function.

Our output cost is similar to posterior regularization. The difference is that we specify preferences via an arbitrary cost function on output structures, while PR uses expectation constraints on posteriors of the model. We compare to the PR tag induction system of Graça et al. (2011) in our experiments, improving over it in several settings.

2.3 Exploiting Resources

Much of the work mentioned above also benefits from leveraging existing resources. These may be curated or crowdsourced resources like the Wiktionary (Li et al., 2012), or traditional annotated treebanks for languages other than those under investigation (Cohen et al., 2011). In this paper, we use tag statistics from treebanks for 11 languages to impose our structural bias for a different set of languages used in our POS induction experiments.

Substantial recent work has improved many NLP tasks by leveraging multilingual or parallel text (Cohen and Smith, 2009; Snyder et al., 2009; Wang and Manning, 2014), including unsupervised POS tagging (Naseem et al., 2009; Das and Petrov, 2011; Täckström et al., 2013; Ganchev and Das, 2013). This sort of multilingual guidance could also be captured by particular output cost functions, though we leave this to future work.

3 Unsupervised Structure Learning

We consider a structured unsupervised learning setting. We use $\mathcal{X}$ to denote our set of possible structured inputs, and for a particular $\boldsymbol{x} \in \mathcal{X}$, we use $\mathcal{Y}(\boldsymbol{x})$ to denote the set of valid structured outputs for $\boldsymbol{x}$. We are given a dataset of inputs $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$. To map inputs to outputs, we start by building a model of the joint probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$. We use a log-linear parameterization with feature vector $\boldsymbol{f}$ and weight vector $\boldsymbol{\theta}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\right\}}{\sum_{\boldsymbol{x}' \in \mathcal{X}, \boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}}$$

where the sum in the denominator ranges over all possible inputs and all valid outputs for them.

In this paper, we consider ways of learning the parameters $\boldsymbol{\theta}$. Given $\boldsymbol{\theta}$, at test time we output a $\boldsymbol{y}$ for a new $\boldsymbol{x}$ using, e.g., Viterbi or minimum Bayes risk decoding; we use the latter in this paper.

3.1 EM and Contrastive Estimation

We start by reviewing two ways of choosing $\boldsymbol{\theta}$. The expectation-maximization algorithm (EM; Dempster et al., 1977) finds a local optimum of the marginal (log-)likelihood of the observations $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$. The marginal log-likelihood is a sum over all $\boldsymbol{x}^{(i)}$ of the **gain function** $\gamma_{\text{EM}}(\boldsymbol{x}^{(i)})$:

$$\begin{aligned}\gamma_{\text{EM}}(\boldsymbol{x}^{(i)}) &= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) \\ &= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} \\ &\quad - \underbrace{\log \sum_{\boldsymbol{x}' \in \mathcal{X}, \boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}}_{Z(\boldsymbol{\theta})}\end{aligned}$$

The difficulty is the final term, $\log Z(\boldsymbol{\theta})$, which requires summing over all possible inputs and all valid outputs for them. This summation is typically intractable for structured problems, and may even diverge. For this reason, EM is typically only used to train log-linear model weights when $Z(\boldsymbol{\theta}) = 1$, e.g., for hidden Markov models, probabilistic context-free grammars, and models composed of locally-normalized log-linear models (Berg-Kirkpatrick et al., 2010), among others.

There have been efforts at approximating the summation over elements of $\mathcal{X}$, whether by limiting sequence length (Haghighi and Klein, 2006), only summing over observations in the training data (Riezler, 1999), restricting the observation space based on the task (Dyer et al., 2011), or using Gibbs sampling to obtain an unbiased sample of the full space (Della Pietra et al., 1997; Rosenfeld, 1997).

Contrastive estimation (CE) addresses this challenge by using a **neighborhood** function $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$ that generates a set of inputs that are "corruptions" of an input $\boldsymbol{x}$; $\mathcal{N}(\boldsymbol{x})$ always includes $\boldsymbol{x}$. Using shorthand $\mathcal{N}_i$ for $\mathcal{N}(\boldsymbol{x}^{(i)})$, CE corresponds to maximizing the sum over inputs $\boldsymbol{x}^{(i)}$ of the gain

$$\begin{aligned}\gamma_{\text{CE}}(\boldsymbol{x}^{(i)}) &= \log\ \Pr(\boldsymbol{x}^{(i)} \mid \mathcal{N}_i) \\ &= \log \frac{\sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{y})}{\sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} p_{\boldsymbol{\theta}}(\boldsymbol{x}', \boldsymbol{y}')} \\ &= \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} - \\ &\qquad \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\}\end{aligned}$$

Two $\log Z(\boldsymbol{\theta})$ terms cancel out, leaving the summation over input/output pairs in the neighborhood instead of the full summation over pairs.

Two desiderata govern the choice of $\mathcal{N}$. One is to make the summation over its elements computationally tractable. If $\mathcal{N}(\boldsymbol{x}) = \mathcal{X}$ for all $\boldsymbol{x} \in \mathcal{X}$, we obtain EM, so a smaller neighborhood typically must be used in practice. The second consideration is to target learning for the task of interest. For POS tagging and dependency parsing, Smith and Eisner (2005a, 2005b) used neighborhood functions that corrupted the observations in systematic ways, e.g., their TRANS1 neighborhood contains the original sentence along with those that result from transposing a single pair of adjacent words. The intent was to force the learner to explain why the given sentences were observed at the expense of the corrupted sentences.

Next we present our modifications to contrastive estimation. Both can be viewed as adding specialized cost functions that penalize some part of the structured input/output pair.

4 Modeling Corruption Costs

While CE allows us to specify a set of corrupted $\boldsymbol{x}$ for each $\boldsymbol{x}^{(i)}$ via the neighborhood function $\mathcal{N}$, it says nothing about how bad each corruption is. The same type of corruption might be harmful in one context and not harmful in another.

This fact was suggested as the reason why certain neighborhoods did not work as well for POS tagging as others (Smith and Eisner, 2005a). One poorly-performing neighborhood consisted of sentences in which a single word of the original was deleted. Deleting a single word in a sentence might not harm grammaticality. By contrast, neighborhoods that transpose adjacent words led to better results. These kinds of corruptions are expected to be more frequently harmful, at least for languages with relatively rigid word order. However, there may still be certain transpositions that are benign, at least for grammaticality.

To address this, we introduce an **observation cost function** $\Delta : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$ that indicates how much two observations differ. Using Δ, we define the following gain function $\gamma_{\text{CCE}_1}(\boldsymbol{x}^{(i)}) =$

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} -$$

$$\log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \Delta(\boldsymbol{x}^{(i)}, \boldsymbol{x}')\right\}$$

The function Δ inflates the score of neighborhood entries with larger differences from the observed $\boldsymbol{x}^{(i)}$. This gain function is inspired by ideas from structured large-margin learning (Taskar et al., 2003; Tsochantaridis et al., 2005), specifically softmax-margin (Povey et al., 2008; Gimpel and Smith, 2010). Softmax-margin extends conditional likelihood by allowing the user to specify a cost function to give partial credit for structures that are partially correct. Conditional likelihood, by contrast, treats all incorrect structures equally.

While softmax-margin uses a cost function to specify how two *output* structures differ, our gain function γ_{CCE_1} uses a cost function Δ to specify how two *inputs* differ. But the motivations are similar: since poor structures have their scores artificially inflated by Δ, learning pays more attention to them, choosing weights that penalize them more than the lower-cost structures.

4.1 Observation Cost Functions

What types of cost functions should we consider? For efficient inference, we want to ensure that Δ decomposes additively across parts of the corrupted input $\boldsymbol{x}'$ in the same way as the features; we assume unigram and bigram features in this paper.

In addition, the choice of the observation cost function Δ is tied to the choice of neighborhood function. In our experiments, we use neighborhoods that change the *order* of words in the observation but not the *set* of words. Our first cost function simply counts the number of novel bigrams introduced when corrupting the original:

$$\Delta_I(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = \alpha \sum_{j=1}^{|\boldsymbol{x}|+1} \mathbb{I}\left[x_{j-1}x_j \notin \text{2grams}(\boldsymbol{x}^{(i)})\right]$$

where x_j is the jth word of sentence $\boldsymbol{x}$, x_0 is the start-of-sentence marker, $x_{|\boldsymbol{x}|+1}$ is the end-of-sentence marker, $\text{2grams}(\boldsymbol{x})$ returns the set of bigrams in $\boldsymbol{x}$, $\mathbb{I}[]$ returns 1 if its argument is true and 0 otherwise, and α is a constant to be tuned. We call this cost function MATCH. Only $\boldsymbol{x}^{(i)}$ (which is always contained in $\mathcal{N}_i$) is guaranteed to have cost 0. In the TRANS1 neighborhood, corrupted sequences will be penalized more if their transpositions occur in the middle of the sentence rather than at the beginning or end.

We also consider a version that weights the indicator by the negative log probability of the novel bigram: $\Delta_{LM}(\boldsymbol{x}^{(i)}, \boldsymbol{x}) =$

$$\alpha \sum_{j=1}^{|\boldsymbol{x}|+1} -\log \mathrm{P}(x_j | x_{j-1}) \mathbb{I}\left[x_{j-1}x_j \notin \text{2grams}(\boldsymbol{x}^{(i)})\right]$$

where $\mathrm{P}(x_j | x_{j-1})$ is obtained from a bigram language model. Among novel bigrams in the corruption $\boldsymbol{x}$, if the second word is highly surprising conditioned on the first, the bigram will incur high cost. We refer to $\Delta_{LM}(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ as MATLM.

5 Expressing Structural Preferences

Our second modification to CE allows us to specify structural preferences for outputs $\boldsymbol{y}$. We first note that there exist objective functions for *supervised* structure prediction that never require computing the feature vector for the true output $\boldsymbol{y}^{(i)}$. Examples include Bayes risk (Kaiser et al., 2000; Povey and Woodland, 2002) and structured ramp loss (Do et al., 2008). These two objectives do, however, need to compute a cost function $\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$, which requires the true output $\boldsymbol{y}^{(i)}$. We start with the following form of structured ramp loss from Gimpel and Smith (2012), transformed here to a gain function:

$$\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \left(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\right) -$$

$$\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \left(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}') + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}')\right) \quad (1)$$

Maximizing this gain function for supervised learning corresponds to increasing the model score of outputs that have both high model score ($\boldsymbol{\theta}^\top \boldsymbol{f}$) and low cost, while decreasing the model score of outputs with high model score and *high* cost.

For unsupervised learning, we do not have $\boldsymbol{y}^{(i)}$, so we simply drop $\boldsymbol{y}^{(i)}$ from the cost function. The result is an **output cost function** $\pi : \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ which captures our *a priori* knowledge about desired output structures. The value of $\pi(\boldsymbol{y})$ should be large for outputs $\boldsymbol{y}$ that are far from the ideal. In this paper, we consider POS induction and use intrinsic evaluation; however, in a real-world scenario, the output cost function could use signals derived from the downstream task in which the tags are being used.

Given π, we convert each max to a $\log \sum \exp$ in Eq. 1 and introduce the contrastive neighborhood into the second term, defining our new gain function $\gamma_{\text{CCE}_2}(\boldsymbol{x}^{(i)}) =$

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp \left\{ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \pi(\boldsymbol{y}) \right\} -$$
$$\log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp \left\{ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \pi(\boldsymbol{y}') \right\}$$

Gimpel (2012) found that using such "softened" versions of the ramp losses worked better than the original versions (e.g., Eq. 1) when training machine translation systems.

5.1 Output Cost Functions

The output cost π should capture our desiderata about $\boldsymbol{y}$ for the task of interest. We consider universal POS tag subsequences analogous to the universal syntactic rules of Naseem et al. (2010). In doing so, we use the universal tags of Petrov et al. (2012): NOUN, VERB, ADJ (adjective), ADV (adverb), PRON (pronoun), DET (determiner), ADP (pre/postposition), NUM (numeral), CONJ (conjunction), PRT (particle), '.' (punctuation), and X (other).

We aimed for a set of rules that would be robust across languages. So, we used treebanks for 11 languages from the CoNLL 2006/2007 shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007) other than those used in our POS induction experiments. In particular, we used Arabic, Bulgarian, Catalan, Czech, English, Spanish, German, Hungarian, Italian, Japanese, and Turkish. We replicated shorter treebanks a sufficient number of times until they were a similar size as the largest treebank. Then we counted gold POS tag unigrams and bigrams from the concatenation.

tag unigram	count	cost
X	50783	3.83
NUM	174613	2.59
PRT	179131	2.57
ADV	330210	1.96
CONJ	436649	1.68
PRON	461880	1.62
DET	615284	1.33
ADJ	694685	1.21
ADP	906922	0.95
VERB	1018989	0.83
.	1042662	0.81
NOUN	2337234	0

tag bigram	count	cost
DET PRT	109	84.41
DET CONJ	518	68.82
NUM ADV	1587	57.63
NOUN NOUN	409828	2.09
DET NOUN	454980	1.04
NOUN .	504897	0

Table 1: Counts and costs for universal tags based on treebanks for 11 languages not used in POS induction experiments.

Where $\#(y)$ is the count of tag y in the treebank concatenation, the cost of y is

$$u(y) = \log \left(\frac{\max_{y'} \#(y')}{\#(y)} \right)$$

and, where $\#(\langle y_1, y_2 \rangle)$ is the count of tag bigram $\langle y_1, y_2 \rangle$, the cost of $\langle y_1, y_2 \rangle$ is

$$u(\langle y_1, y_2 \rangle) = 10 \times \log \left(\frac{\max_{\langle y_1', y_2' \rangle} \#(\langle y_1', y_2' \rangle)}{\#(\langle y_1, y_2 \rangle)} \right)$$

We use a multiplier of 10 in order to exaggerate count differences among bigrams, which generally are closer together than unigram counts. In Table 1, we show counts and costs for all tag unigrams and selected tag bigrams.[2]

Given these costs for individual tag unigrams and bigrams, we use the following π function, which we call Univ:

$$\pi(\boldsymbol{y}) = \beta \sum_{j=1}^{|\boldsymbol{y}|+1} u(y_j) + u(\langle y_{j-1}, y_j \rangle)$$

where β is a constant to be tuned and y_j is the jth tag of $\boldsymbol{y}$. We define y_0 to be the beginning-of-sentence marker and $y_{|\boldsymbol{y}|+1}$ to be the end-of-sentence marker (which has unigram cost 0).

Many POS induction systems use one-tag-per-type constraints (Blunsom and Cohn, 2011; Gelling et al., 2012), which often lead to higher

[2] The complete tag bigram list is provided in the supplementary material.

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right\} - \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}')\right\} \tag{2}$$

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^{(i)})} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \pi(\boldsymbol{y})\right\} - \log \sum_{\boldsymbol{x}' \in \mathcal{N}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}')} \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{y}') + \Delta(\boldsymbol{x}^{(i)}, \boldsymbol{x}') + \pi(\boldsymbol{y}')\right\} \tag{3}$$

Figure 1: Contrastive estimation (Eq. 2) and cost-augmented contrastive estimation (Eq. 3). L2 regularization terms ($\frac{C}{2}\sum_{j=1}^{|\boldsymbol{\theta}|}\theta_j^2$) are not shown here but were used in our experiments.

accuracies even though the gold standard is not constrained in this way. This constraint can be encoded as an output cost function, though it would require approximate inference (Poon et al., 2009).

6 Cost-Augmented CE

We extended the objective function underlying CE by defining two new types of cost functions, one on observations (§4) and one on outputs (§5). We combine them into a single objective, which we call **cost-augmented contrastive estimation** (CCE), shown as Eq. 3 in Figure 1.

If the cost functions Δ and π factor in the same way as the features $\boldsymbol{f}$, then it is straightforward to implement CCE atop an existing CE implementation. The additional terms in the cost functions can be implemented as features with fixed weights (albeit where the weight differs depending on the context).

7 Model Selection

Our modifications give increased flexibility, but require setting new hyperparameters. In addition to the choice of the cost functions, each has a weight: α for Δ and β for π. We need ways to set these weights that do not require labeled data.

Smith and Eisner (2005a) chose the hyperparameter values that yielded the best CE objective on held-out development data. We use their strategy, though we experiment with two others as well.[3] In particular, we estimate held-out data log-likelihood via the method of Bengio et al. (2013) and also consider ways of combining outputs from multiple models.

7.1 Estimating Held-Out Log-Likelihood

Bengio et al. (2013) recently proposed ways to efficiently estimate held-out data log-likelihood for generative models. They showed empirically that a simple, biased version of their conservative sampling-based log-likelihood (CSL) estimator can be useful for model selection.

The biased CSL requires a Markov chain on the variables in the model (i.e., $\boldsymbol{x}$ and $\boldsymbol{y}$) as well as the ability to compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$. It generates consecutive samples of $\boldsymbol{y}$ from a Markov chain initialized at each $\boldsymbol{x}$ in a development set D, with S Markov chains run for each $\boldsymbol{x}$. We compute and sum $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}^j)$ for each sampled $\boldsymbol{y}^j$, then sum over all $\boldsymbol{x}$ in D. The result is a biased estimate for the log-likelihood of D. Bengio et al. showed that these biased estimates could give the same model ranking as unbiased estimates, though more efficiently. They also showed that taking the single, initial sample from the S Markov chains resulted in the same model ranking as using many samples from each chain. We follow suit here.

Our Markov chain is a blocked Gibbs sampler in which we alternate between sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$. Since we only use a single sample from each Markov chain and initialize each chain to $\boldsymbol{x}$, this simply amounts to drawing S samples from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. To sample from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, we use the exact algorithm obtained by running the backward algorithm and then performing left-to-right sampling of tags using the local features and requisite backward terms to define the local tag distributions.

We then compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$ for each sampled $\boldsymbol{y}$. If there are no features in $\boldsymbol{f}$ that look at more than one word (which is the case with the features used in our experiments), then this probability factors:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) = \prod_{k=1}^{|\boldsymbol{y}|} p_{\boldsymbol{\theta}}(x_k|y_k)$$

This is easily computable assuming that we have normalization constants $Z(y)$ cached for each tag y. To compute each $Z(y)$, we sum over all words observed in the training data (replacing some with a special UNK token; see below). We can then compute likelihoods for individual words and mul-

[3] When using their strategy for CCE, we compute the CE criterion only, omitting the costs. We do so because the weights of the cost terms can have a large impact on the magnitude of the objective, making it difficult to do a fair comparison of models with different cost weights.

tiply them across the words in the sentence to compute $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$.

To summarize, we get a log-likelihood estimate for development set $D = \{\boldsymbol{x}^{(i)}\}_{i=1}^{|D|}$ by sampling S times from $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(i)})$ for each $\boldsymbol{x}^{(i)}$, getting samples $\{\{\boldsymbol{y}^{(i),j}\}_{j=1}^{S}\}_{i=1}^{|D|}$, then we compute

$$\textstyle\sum_{i=1}^{|D|}\sum_{j=1}^{S}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{y}^{(i),j})$$

We used values of $S \in \{1, 10, 100\}$, finding that the ranking of models was consistent across S values. We used $S = 10$ in all results reported below.

We note that this estimator was originally presented for generative models, and that (C)CE is not a generative training criterion. It seeks to maximize the conditional probability of an observation given its neighborhood. Nonetheless, when implementing our log-likelihood estimator, we treat the model as a generative model, computing the $Z(y)$ constants by summing over all words in the vocabulary.

7.2 System Combination

We can avoid choosing a single model by combining the outputs of multiple models via system combination. We decode test data by using posterior decoding. To combine the outputs of multiple models, we find the max-posterior tag under each model, then choose the highest vote-getter, breaking ties arbitrarily.

However, when doing POS induction without a tag dictionary, the tags are simply unique identifiers and may not have consistent meaning across runs. To address this, we propose a novel voting scheme that is inspired by the widely-used 1-to-1 accuracy metric for POS induction (Haghighi and Klein, 2006). This metric maps system tags to gold tags to maximize accuracy with the constraint that each gold tag is mapped to at most once. The optimal mapping can be found by solving a maximum weighted bipartite matching problem.

We adapt this idea to map tags between two systems, rather than between system tags and gold tags. Given k systems that we want to combine, we choose one to be the **backbone** and map the remaining $k - 1$ systems' outputs to the backbone.[4] After mapping each system's output to the backbone system, we perform simple majority voting among all k systems. To choose the backbone, we consider each of the k systems in turn as backbone and maximize the sum of the weights of the weighted bipartite matching solutions found. This is a heuristic that attempts to choose a backbone that is similar to all other systems. We found that highly-weighted matchings often led to high POS tagging accuracy metrics. We call this voting scheme ALIGN. To see the benefit of ALIGN, we also compare to a simple scheme (NAÏVE) that performs majority voting without any tag mapping.

8 Experiments

Task and Datasets We consider POS induction without tag dictionaries using five freely-available datasets from the PASCAL shared task (Gelling et al., 2012).[5] These include Danish (DA), using the Copenhagen Dependency Treebank v2 (Buch-Kromann et al., 2007); Dutch (NL), using the Alpino treebank (Bouma et al., 2001); Portuguese (PT), using the Floresta Sintá(c)tica treebank (Afonso et al., 2002); Slovene (SL), using the jos500k treebank (Erjavec et al., 2010); and Swedish (SV), using the Talbanken treebank (Nivre et al., 2006). We use their provided training, development, and test sets.

Evaluation We fix the number of tags in our models to 12, which matches the number of universal tags from Petrov et al. (2012). We use both many-to-1 (M-1) and 1-to-1 (1-1) accuracy as our evaluation metrics, using the universal tags for the gold standard (which was done for the official evaluation for the shared task).[6] We note that our π function assigns identities to tags (e.g., tag 1 is assumed to be NOUN), so we could use actual tagging accuracy when training with the π cost function. But we use M-1 and 1-1 accuracy to enable easier comparison both among different settings and to prior work.

Baselines From the shared task, we compare to all entries that used 12 tags. These include

[4] We use the LEMON C++ toolkit (Dezs et al., 2011) to solve the maximum weighted bipartite matching problems.

[5] http://wiki.cs.ox.ac.uk/InducingLinguisticStructure/SharedTask

[6] It is common to use a greedy algorithm to compute 1-to-1 accuracy, e.g., as in the shared task scoring script (http://www.dcs.shef.ac.uk/~tcohn/wils/eval.tar.gz), though the optimal mapping can be computed efficiently via the maximum weighted bipartite matching algorithm, as stated above. We use the shared task scorer for all results here for ease of comparison. When we instead evaluate using the optimal mapping, we find that accuracies are usually only slightly higher than those found by the greedy algorithm.

BROWN clusters (Brown et al., 1992), clusters obtained using the `mkcls` tool (Och, 1995), and the featurized HMM with sparsity constraints trained using posterior regularization (PR), described by Graça et al. (2011). The PR system achieved the highest average 1-1 accuracy in the shared task.

We restrict our attention to systems that use 12 tags because the M-1 and 1-1 metrics are highly dependent upon the number of hypothesized tags. In general, using more tags leads to higher M-1 and lower 1-1 (Gelling et al., 2012). By keeping the number of tags fixed, we hope to provide a cleaner comparison among approaches.

We compare to two other baselines: an HMM trained with 500 iterations of EM and an HMM trained with 100 iterations of stepwise EM (Liang and Klein, 2009). We used random initialization as done by Liang and Klein: we set each parameter in each multinomial to $\exp\{1 + c\}$, where $c \sim U[0, 1]$, then normalized to get probability distributions. For stepwise EM, we used minibatch size 3 and stepsize reduction power 0.7.

For all models we trained, including both baselines and CCE, we used only the training data during training and used the unannotated development data for certain model selection criteria. No labels were used except for final evaluation on the test data. Therefore, we need a way to handle unknown words in test data. When running EM and stepwise EM, while reading in the final 10% of sentences in the training set, we replace novel words with the special token UNK. We then replace unknown words in test data with UNK.

8.1 CCE Setup

Features We use standard indicator features on tag-tag transitions and tag-word emissions, the spelling features from Smith and Eisner (2005a), and additional emission features based on Brown clusters. The latter features are simply indicators for tag-cluster pairs—analogous to tag-word emissions in which the word is replaced by its Brown cluster identifier. We run Brown clustering (Liang, 2005) on the POS training data for each language, once with 12 clusters and once with 40, then add tag-cluster emission features for each clustering and one more for their conjunction.[7]

Learning We solve Eq. 2 and Eq. 3 by running LBFGS until convergence on the training data, up to 100 iterations. We tag the test data with minimum Bayes risk decoding and evaluate.

We use two neighborhood functions:

- TRANS1: the original sentence along with all sentences that result from doing a single transposition of adjacent words.
- SHUFF10: the original sentence along with 10 random permutations of it.

We use L2 regularization, adding $\frac{C}{2}\sum_{j=1}^{|\boldsymbol{\theta}|}\theta_j^2$ to the objectives shown in Figure 1. We use a fixed (untuned) $C = 0.0001$ for all experiments reported below.[8] We initialize each CE model by sampling weights from $\mathcal{N}(0, 1)$.

Cost Functions The cost functions Δ and π have constants α and β which balance their contributions relative to the model score and must be tuned. We consider the ways proposed in Section 7, namely tuning based on the contrastive estimation criterion computed on development data (CE), the log-likelihood estimate on development data with $S = 10$ (LL), and our two system combination algorithms: naïve voting (NAÏVE) and aligned voting (ALIGN), both of which use as input the 4 system outputs whose hyperparameters led to the highest values for the CE criterion on development data.

We used $\alpha \in \{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 0.01, 0.03, 0.1, 0.3\}$ and $\beta \in \{3 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$. Setting $\alpha = \beta = 0$ gives us CE, which we also compare to. When using both MATLM and UNIV simultaneously, we first choose the best two α values by the LL criterion and the best two β values by the CE criterion when using only those individual costs. This gives us 4 pairs of values; we run experiments with these pairs and choose the pair to report using each of the model selection criteria. For system combination, we use the 4 system outputs resulting from these 4 pairs.

For training bigram language models for the MATLM cost, we use the language's POS training data concatenated with its portion of the Europarl v7 corpus (Koehn, 2005) and the text of its

[7] To handle unknown words: for words that only appear in the final 10% of training sentences, we replace them with UNK when firing their tag-word emission features. We use special Brown cluster identifiers reserved for UNK. But we still use all spelling features derived from the actual word type. For unknown words at test time, we use the UNK emission feature, the Brown cluster features with the special UNK cluster identifiers, and the word's actual spelling features.

[8] In subsequent experiments we tried $C \in \{0.01, 0.001\}$ for the baseline CE setting and found minimal differences.

neigh-borhood	cost	mod. sel.	DA		NL		PT		SL		SV		avg	
			M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1
Shuff10	none	N/A	45.0	38.0	55.1	45.7	**54.2**	38.0	54.7	45.7	47.4	31.3	51.3	39.7
	Match	CE	48.9	31.5	56.5	46.4	**54.2**	37.7	55.9	46.8	**48.9**	**33.8**	52.9	39.2
		LL	49.9	34.4	56.5	46.4	54.1	**38.9**	57.2	**48.9**	**48.9**	**33.8**	53.3	40.5
	MatLM	CE	49.1	34.3	**59.6**	**50.4**	53.6	37.1	55.0	46.2	48.8	33.1	53.2	40.2
		LL	**50.2**	**40.0**	**59.6**	**50.4**	53.1	36.0	**58.0**	48.4	48.8	33.1	**53.9**	**41.6**
Trans1	none	N/A	58.5	42.7	62.5	49.5	**70.7**	**43.8**	58.6	46.1	58.7	53.8	61.8	47.2
	Match	CE	58.5	42.5	66.3	53.3	70.6	43.3	59.1	45.6	59.3	54.2	62.7	47.8
		LL	58.8	42.8	66.3	53.3	70.6	43.3	**60.3**	43.7	**59.8**	**54.9**	**63.1**	47.6
	MatLM	CE	**59.4**	**43.5**	63.8	50.1	70.2	43.0	58.5	46.1	59.2	54.8	62.2	47.5
		LL	58.7	42.8	**66.5**	**60.4**	70.5	43.6	59.1	**47.7**	59.2	54.8	62.8	**49.9**

Table 2: Results for observation cost functions. The CE baseline corresponds to rows where cost="none". Other rows are CCE. Best score for each column and each neighborhood is bold.

Wikipedia. The word counts for the Wikipedias used range from 18M for Slovene to 1.9B for Dutch. We used modified Kneser-Ney smoothing as implemented by SRILM (Stolcke, 2002).

8.2 Results

We present two sets of results. First we compare our Match and MatLM observation cost functions for our two neighborhoods and two ways of doing model selection. Then we do a broader comparison, comparing both types of costs and their combination to our full set of baselines.

Observation Cost Functions In Table 2, we show results for observation cost functions. We note that the Trans1 neighborhood works much better than the Shuff10 neighborhood, but we find that using cost functions can close the gap in certain cases, particularly for Dutch and Slovene for which the Shuff10 MatLM scores approach or exceed the Trans1 scores without a cost.

Since the Shuff10 neighborhood exhibits more diversity than Trans1, we expect to see larger gains from using observation cost functions. We do in fact see larger gains in M-1, e.g., average improvements are 1.6-2.6 for Shuff10 and 0.4-1.3 for Trans1, though 1-1 gains are closer.

For Trans1, while Match does reach a slightly higher average M-1 than MatLM, the latter does much better in 1-1 (49.9 vs. 47.6 when using LL for model selection). For Shuff10, MatLM consistently does better than Match. Nonetheless, we suspect Match works as well as it does because it at least differentiates the observation (which is always part of the neighborhood) from the corruptions.

We find that the LL model selection criterion consistently works better than the CE criterion for model selection. When using LL model selection and fixing the neighborhood, all average scores are better than their CE baselines. For M-1, the average improvement is 1.0 to 2.6 points, and for 1-1 the average improvement ranges from 0.4 to 2.7.

We find the best overall performance when using MatLM with LL model selection with the Trans1 neighborhood, and we report this setting in our subsequent experiments.

Output Cost Function Table 3 shows results when using our Univ output cost function, as well as our full set of baselines. All (C)CE experiments used the Trans1 neighborhood.

We find that our contrastive estimation baseline (cost="none") has a higher average M-1 (61.8) than all results from the shared task, but its average 1-1 accuracy is lower than that reached by posterior regularization, the best system in the shared task according to 1-1. Using an observation cost function increases both M-1 and 1-1: MatLM yields an average 1-1 of 49.9, nearing the 50.1 of PR while exceeding it in M-1 by nearly 2 points.

When using the Univ cost function, we see some variation in performance across model selection criteria, but we find improvements in both M-1 and 1-1 accuracy under most settings. When doing model selection via Align voting, we roughly match the average 1-1 of PR, and when using the CE criterion, we beat it by 1 point on average (51.3 vs. 50.1).

Combined Costs When using the Univ cost, we find that model selection via CE works better than LL. So for the combined costs, we took the two best MatLM weights (α values) according to LL and the two best Univ weights (β values) according to CE and ran combined cost experiments (MatchLM+Univ) with the four pairs of hyperparameters. Then from among these four,

system	DA		NL		PT		SL		SV		avg	
	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1	M-1	1-1
HMM, EM	42.5	28.1	53.0	40.6	59.4	33.7	50.3	34.7	49.3	33.9	50.9	34.2
HMM, stepwise EM	51.7	38.2	61.6	45.2	66.5	46.7	53.6	35.7	55.3	39.6	57.7	41.1
Brown	47.1	39.2	57.3	43.1	67.6	51.6	58.3	42.3	57.6	51.3	57.6	45.5
`mkcls`	53.1	44.2	63.0	54.1	68.1	46.3	50.4	40.6	57.3	43.6	58.4	45.8
posterior regularization	53.8	45.6	57.6	45.4	74.4	56.1	60.0	**48.5**	58.8	54.9	60.9	50.1

contrastive estimation													
cost	model sel.												
none	N/A	58.5	42.7	62.5	49.5	70.7	43.8	58.6	46.1	58.7	53.8	61.8	47.2
Match	LL	58.8	42.8	66.3	53.3	70.6	43.3	60.3	43.7	59.8	54.9	63.1	47.6
MatLM	LL	58.7	42.8	**66.5**	60.4	70.5	43.6	59.1	47.7	59.2	54.8	62.8	49.9
Univ	CE	59.7	45.6	60.6	51.1	70.0	62.7	**60.9**	44.1	57.1	52.8	61.7	51.3
	LL	59.5	42.2	62.1	56.3	70.7	43.1	**60.9**	44.1	57.1	52.8	62.1	47.7
	Naïve	59.2	45.6	62.2	52.8	72.7	52.7	60.0	43.8	56.2	53.0	62.2	49.6
	Align	**61.6**	**47.3**	63.7	54.5	74.4	53.1	59.7	42.1	56.6	53.2	63.2	50.0
MatLM	CE	59.8	45.7	60.4	48.4	70.0	**62.8**	52.9	45.0	59.4	54.9	60.5	51.4
+	LL	59.3	42.5	61.9	56.2	70.8	43.1	59.3	41.9	**60.0**	**55.1**	62.3	47.8
	Naïve	58.5	44.4	64.9	60.3	65.4	52.1	55.5	45.9	59.0	54.4	60.6	51.4
Univ	Align	61.1	45.4	66.2	**60.9**	**75.8**	49.8	59.5	48.2	59.0	54.4	**64.3**	**51.7**

Table 3: Unsupervised POS tagging accuracies for five languages, showing results for three systems from the PASCAL shared task as well as three other baselines (EM, stepwise EM, and contrastive estimation). All (C)CE results use the Trans1 neighborhood. The best score in each column is bold.

we again chose results by CE, LL, and both voting schemes.

The results are shown in the lower part of Table 3. We find different trends in M-1 and 1-1 depending on whether we use CE or LL for model selection, which may be due to our limited hyperparameter search stemming from computational constraints. However, by comparing Naïve to Align, we see a consistent benefit from aligning tags before voting, leading to our highest average accuracies. In particular, using MatchLM+Univ and Align, we improve over CE by 2.5 in M-1 and 4.5 in 1-1, also improving over the best results from the shared task.

9 Conclusion

We have shown how to modify contrastive estimation to use additional sources of knowledge, both in terms of observation and output cost functions. We adapted a recently-proposed technique for estimating the log-likelihood of held-out data, finding it to be effective as a model selection criterion when using observation cost functions. We improved tagging accuracy by using weak supervision in the form of universal tag frequencies. We proposed a system combination method for POS induction systems that consistently performs better than naïve voting and circumvents hyperparameter selection. We reported results on par with or exceeding the best systems from the PASCAL 2012 shared task.

Contrastive estimation has been shown effective for numerous NLP tasks, including dependency grammar induction (Smith and Eisner, 2005b), bilingual part-of-speech induction (Chen et al., 2011), morphological segmentation (Poon et al., 2009), and machine translation (Xiao et al., 2011). The hope is that our contributions can benefit these and other applications of weakly-supervised learning.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and Waleed Ammar, Chris Dyer, David McAllester, Sasha Rush, Nathan Schneider, Noah Smith, and John Wieting for helpful discussions.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. Floresta sintá(c)tica: a treebank for Portuguese. In *Proc. of LREC*.

Y. Bengio, L. Yao, and K. Cho. 2013. Bounding the test log-likelihood of generative models. *arXiv preprint arXiv:1311.6184*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proc. of EMNLP*.

P. Blunsom and T. Cohn. 2011. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proc. of ACL*.

G. Bouma, G. Van Noord, and R. Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. *Language and Computers*, 37(1).

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based N-gram models of natural language. *Computational Linguistics*, 18(4).

M. Buch-Kromann, J. Wedekind, and J. Elming. 2007. The Copenhagen Danish-English dependency treebank v. 2.0. *code.google.com/p/copenhagen-dependency-treebank*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

D. Chen, C. Dyer, S. B. Cohen, and N. A. Smith. 2011. Unsupervised bilingual POS tagging with Markov random fields. In *Proc. of the First Workshop on Unsupervised Learning in NLP*.

S. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of NAACL*.

S. B. Cohen, D. Das, and N. A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*.

M. Creutz and K. Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. of ACL*.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4).

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

B. Dezs, A. Jüttner, and P. Kovács. 2011. LEMON - an open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.*, 264(5).

C. B. Do, Q. Le, C. H. Teo, O. Chapelle, and A. Smola. 2008. Tighter bounds for structured estimation. In *Advances in NIPS*.

C. Dyer, J. H. Clark, A. Lavie, and N. A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proc. of ACL*.

T. Erjavec, D. Fiser, S. Krek, and N. Ledinek. 2010. The JOS linguistically tagged corpus of Slovene. In *Proc. of LREC*.

K. Ganchev and D. Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proc. of EMNLP*.

K. Ganchev, J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proc. of ACL*.

K. Ganchev, J. V. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11.

D. Gelling, T. Cohn, P. Blunsom, and J. V. Graça. 2012. The PASCAL challenge on grammar induction. In *Proc. of NAACL-HLT Workshop on the Induction of Linguistic Structure*.

K. Gimpel and N. A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

K. Gimpel and N. A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proc. of NAACL*.

K. Gimpel. 2012. *Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

S. Goldwater and T. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*.

J. V. Graça, K. Ganchev, L. Coheur, F. Pereira, and B. Taskar. 2011. Controlling complexity in part-of-speech induction. *J. Artif. Int. Res.*, 41(2).

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of HLT-NAACL*.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. of EMNLP-CoNLL*.

J. Kaiser, B. Horvat, and Z. Kacic. 2000. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. of ICSLP*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Z. Li, Z. Wang, S. Khudanpur, and J. Eisner. 2010. Unsupervised discriminative language model training for machine translation using simulated confusion sets. In *Proc. of COLING*.

S. Li, J. V. Graça, and B. Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proc. of EMNLP*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *Proc. of NAACL*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. of HLT-NAACL*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2).

T. Naseem, B. Snyder, J. Eisenstein, and R. Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *JAIR*, 36.

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*.

J. Nivre, J. Nilsson, and J. Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proc. of LREC*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of CoNLL*.

F. J. Och. 1995. Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Bachelor's thesis (Studienarbeit), Friedrich-Alexander-Universität Erlangen-Nürnburg, Germany.

S. Petrov, D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

H. Poon, C. Cherry, and K. Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of HLT: NAACL*.

D. Povey and P. C. Woodland. 2002. Minimum phone error and I-smoothing for improved discrimative training. In *Proc. of ICASSP*.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. 2008. Boosted MMI for model and feature space discriminative training. In *Proc. of ICASSP*.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL*.

S. Riezler. 1999. *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen.

R. Rosenfeld. 1997. A whole sentence maximum entropy language model. In *Proc. of ASRU*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

N. A. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proc. of COLING-ACL*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*.

B. Snyder, T. Naseem, and R. Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proc. of ACL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proc. of CoNLL*.

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. 2010c. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proc. of ACL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proc. of EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proc. of CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proc. of EMNLP*.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proc. of ICSLP*.

O. Täckström, D. Das, S. Petrov, R. McDonald, and J. Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in NIPS 16*.

K. Toutanova and M. Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Advances in NIPS*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6.

J. Van Gael, A. Vlachos, and Z. Ghahramani. 2009. The infinite HMM for unsupervised POS tagging. In *Proc. of EMNLP*.

A. Vaswani, A. Pauls, and D. Chiang. 2010. Efficient optimization of an MDL-inspired objective function for unsupervised part-of-speech tagging. In *Proc. of ACL*.

M. Wang and C. D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2.

X. Xiao, Y. Liu, Q. Liu, and S. Lin. 2011. Fast generation of translation forest for large-scale SMT discriminative training. In *Proc. of EMNLP*.