Day 5

Unsupervised Learning

In this class we will address the problem of *unsupervised* learning of linguistic structures, namely *parts-of-speech*. In this setting we are not given any labeled data. Instead, all we get to see is a set of natural language sentences. The underlying question is:

> Can we learn something from raw text?

This task is particularly challenging since the process by which linguistic structures are generated is not always clear and even when it is, it is normally too complex to be formally expressed. Nevertheless, unsupervised learning has been applied to a wide range of natural language processing tasks, such as: Part-of-Speech Induction (Schütze, 1995; Merialdo, 1994; Clark, 2003), Dependency Grammar Induction (Klein and Manning, 2004; Smith and Eisner, 2006), Constituency Grammar Induction (Klein and Manning, 2004), Statistical Word Alignments (Brown et al., 1993) and Anaphora Resolution (Charniak and Elsner, 2009), just to name a few.

Different motivations have pushed research in this area. From both a linguistic and cognitive point of view, unsupervised learning is useful as a tool to study language acquisition. From a machine learning point of view, unsupervised learning is a fertile ground for testing new learning methods, where significant improvements can yet be made. From a more pragmatic perspective, unsupervised learning is required since annotated corpora is a scarce resource for different reasons. Independently of the reason, unsupervised learning is an increasing active field of research.

A first problem with unsupervised learning, since we don't observe any labeled data (i.e., the training set is now $\mathcal{D} = \{x_1, \ldots, x_M\}$), is that most of the methods studied so far (Perceptron, Mira, SVMs) cannot be used since we cannot compare the true output with the predicted output. Note also that a direct minimization of the *complete negative log-likelihood* of the data, $\log P_\theta(\mathcal{D})$, is very challenging, since it would require marginalizing out (*i.e.*, summing over) all possible hidden variables:

$$\log P_\theta(\mathcal{D}) = \sum_{m=1}^{M} \log \sum_{y \in \mathcal{Y}} P_\theta(x_m, y). \tag{5.1}$$

Note also that the objective above is *non-convex* even for a linear model: hence, it may have local minima, which makes optimization much more difficult.

Another observation is that normally we are restricted to generative models, with some remarkable exceptions (Smith and Eisner, 2005a), since the objective of discriminative models when no labels are observed are meaningless ($\sum_{y_m} P(y^m|x^m) = 1$); this rules out, for instance, Maximum Entropy classifiers.

The most common optimization method in the presence of hidden (latent) variables is the Expectation Maximization (EM) algorithm. Note that this algorithm is a generic optimization routine that does not depend on a particular model. The next section will explain the EM algorithm. On Section 5.2 we will apply the EM algorithm to the task of part-of-speech induction, where one is given raw text and a number of clusters and the task is to cluster words that behave similarly in a grammatical sense.

5.1 Expectation Maximization Algorithm

Given a particular model $p_\theta(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and a training corpus $\mathcal{X}$ of D sentences $\bar{\mathbf{x}}^1 \ldots \bar{\mathbf{x}}^D$, training seeks model parameters θ that minimize the negative log-likelihood of the corpus:

$$\textbf{Negative Log Likelihood}: \quad \mathcal{L}(\theta) = \widehat{\mathbf{E}}[-\log p_\theta(\bar{\mathbf{x}})] = \widehat{\mathbf{E}}[-\log \sum_{\bar{\mathbf{y}}} p_\theta(\bar{\mathbf{x}}, \bar{\mathbf{y}})], \tag{5.2}$$

where $\widehat{\mathbf{E}}[f(\bar{\mathbf{x}})] = \frac{1}{D}\sum_{i=1}^{D} f(\bar{\mathbf{x}}^i)$ denotes the empirical average of a function f over the training corpus.

Because of the hidden variables $\bar{\mathbf{y}}$, the likelihood term contains a sum over all possible hidden structures inside of a logarithm, which makes this quantity hard to compute.

The most common minimization algorithm to fit the model parameters in the presence of hidden variables is the Expectation Maximization (EM) algorithm.

The EM procedure can be thought of intuitively in the following way. If we observe the hidden variables' values for all sentences in the corpus, then we could easily compute the maximum likelihood value of the parameters as described in Section 2.2. On the other hand, if we had the model parameters we could label data using the model, and collect the sufficient statistics described in Section 2.2. Since we are working in an unsupervised setting, we never get to observe the hidden state sequence. Instead, given a training set $\mathcal{X} = \{\bar{\mathbf{x}}^1 \dots \bar{\mathbf{x}}^D\}$, we will need to collect sufficient statistics, or expected counts that represent the expected number of times that each hidden variable is expected to be used with the current parameters setting. These sufficient statistics will then be used during learning as fake observations of the hidden variables. Using the node and edge posterior distributions described in Equations 2.17 and 2.18, the sufficient statistics can be computed by the following formulas:

$$\textbf{Initial Counts}: \quad ic(y_l) = \sum_{d=1}^{D} \gamma_1(y_l); \tag{5.3}$$

$$\textbf{Final Counts}: \quad fc(y_N, y_{N-1}) = \sum_{d=1}^{D} \xi_{N-1}(y_l, y_m); \tag{5.4}$$

$$\textbf{Transition Counts}: \quad tc(y_l, y_m) = \sum_{d=1}^{D} \sum_{i=1}^{N-1} \xi_i(y_l, y_m); \tag{5.5}$$

$$\textbf{State Counts}: \quad sc(v_q, y_m) = \sum_{d=1}^{D} \sum_{i=1, \mathbf{x}_i = v_q}^{N} \gamma_i(y_m). \tag{5.6}$$

Compare the previous Equations with the ones described in Section 2.2 for the same quantities. The main difference is that while in the presence of supervised data you sum the observed events, when you have no label data you sum the posterior probabilities of each event. If these probabilities were such that the probability mass was around single events then both Equations will produce the same result.

The EM procedure starts with an initial guess for the parameters θ^0 at time $t = 0$. The algorithm iterates for T iterations until it converges to a local minima of the negative log likelihood, and each iteration is divided into two steps:

The first step - "E Step" (Expectation) - computes the posteriors for the hidden variables $p_\theta(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})$, given the current parameter values θ^t and the observed variables. In the case of the HMM this requires only to run the FB algorithm.

The second step - "M step" (Maximization) - uses $p_\theta(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})$ to "softly fill in" the values of the hidden variables $\bar{\mathbf{y}}$, and collects the sufficient statistics, initial counts (Eq: 5.3), transition counts (Eq: 5.5) and state counts (Eq: 5.6) and uses those counts to estimate maximum likelihood parameters θ^{t+1} as described in Section 2.2.

The EM algorithm is guaranteed to converge to a local minimum of $\mathcal{L}(\theta)$ under mild conditions. Note that we are not committing to the best assignment of the hidden variables, but summing the occurrences of each parameter weighed by the posterior probability of all possible assignments. This modular split into two intuitive and straightforward steps accounts for the vast popularity of EM.

More formally, EM minimizes $\mathcal{L}(\theta)$ via block-coordinate descent on an upper bound $F(q,\theta)$ using an auxiliary distribution over the latent variables $q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})$:

$$\begin{aligned}
\mathcal{L}(\theta) &= \widehat{\mathbf{E}}\left[-\log \sum_{\bar{\mathbf{y}}} p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})\right] && (5.7)\\
&= \widehat{\mathbf{E}}\left[-\log \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) * \frac{p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})}{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}\right] \le \widehat{\mathbf{E}}\left[-\sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log \frac{p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})}{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}\right] && (5.8)\\
&= \widehat{\mathbf{E}}\left[\sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log \frac{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}{p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})}\right] = F(q,\theta), && (5.9)
\end{aligned}$$

where we have multiplied and divided the $p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})$ by the same quantity $q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})$, and the lower bound comes from applying Jensen Inequality (Equation 5.8). $F(q,\theta)$ is normally referred to as the energy function, which comes from the physics field and refers to the energy of a given system that we want to minimize.

$$\textbf{EM Upper Bound}: \quad \mathcal{L}(\theta) \le F(q,\theta) = \widehat{\mathbf{E}}\left[\sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log \frac{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}{p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})}\right]. \quad (5.10)$$

The alternating E and M steps at iteration $t+1$ can be seen as minimizing the energy function first with respect to $q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})$ and then with respect to θ:

$$\mathbf{E}: \quad q^{t+1}(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) = \arg\min_{q(\bar{\mathbf{y}}|\bar{\mathbf{x}})} F(q,\theta^t) = \arg\min_{q(\bar{\mathbf{y}}|\bar{\mathbf{x}})} \mathrm{KL}(q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \,||\, p_{\theta^t}(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})) = p_{\theta^t}(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}); \quad (5.11)$$

$$\mathbf{M}: \quad \theta^{t+1} = \arg\min_{\theta} F(q^{t+1},\theta) = \arg\max_{\theta} \widehat{\mathbf{E}}\left[\sum_{\bar{\mathbf{y}}} q^{t+1}(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log p_\theta(\bar{\mathbf{x}},\bar{\mathbf{y}})\right]; \quad (5.12)$$

where $\mathrm{KL}(q||p) = \mathbf{E}_q[\log \frac{q(\cdot)}{p(\cdot)}]$ is the Kullback-Leibler divergence. The KL term in the E-Step results from dropping all terms from the energy function that are constant for a set θ, in this case the likelihood of the observation sequence $p_\theta(\bar{\mathbf{x}})$:

$$\sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log \frac{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}{p_\theta(\bar{\mathbf{x}}, \bar{\mathbf{y}})} = \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) - \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log p_\theta(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \tag{5.13}$$

$$= \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) - \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log p_\theta(\bar{\mathbf{x}}) p_\theta(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \tag{5.14}$$

$$= \sum_{\bar{\mathbf{y}}} q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) \log \frac{q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})}{p_\theta(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})} - \log p_\theta(\bar{\mathbf{x}}) \tag{5.15}$$

$$= \mathrm{KL}(q(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}) || p_\theta(\bar{\mathbf{y}} \mid \bar{\mathbf{x}})) - \log p_\theta(\bar{\mathbf{x}}). \tag{5.16}$$

Algorithm 13 presents the pseudo code for the EM algorithm. Note that this algorithm is agnostic of a particular model, it only requires the model to implement a common interface.

Algorithm 13 EM algorithm.

```
1:  input: dataset $\mathcal{D}$, an initialized model
2:  for $t = 1$ to $T$ do
3:      model.clear_counts()
4:      for $seq \in \mathcal{D}$ do
5:          E-Step:
6:          posteriors,likelihood =model.compute_posteriors($seq$)
7:          model.update_counts($seq$,posteriors)
8:      end for
9:      M-Step:
10:     model.update_params(counts)
11: end for
```

One important thing to note in Algorithm 13 is that for the HMM model we already have all the model pieces we require. In fact the only method we don't have yet implemented from previous classes is the method to update_counts(posteriors).

Exercise 5.1 *Implement the method update_counts(seq,posteriors).*

```
def update_counts(self,seq,posteriors):
```

Use the method you defined previously to check the count tables to check if this method is correct. Use a corpus with only one sentence to make the test simpler.

```
In []: run readers/pos_corpus.py
In []: posc = PostagCorpus("en",max_sent_len=15,train_sents=1,
    dev_sents=0,test_sents=0)
```

```
In []: run sequences/hmm.py
In []: hmm = HMM(posc)
In []: hmm.train_supervised(posc.train,smoothing=0.1)
In []: hmm.clear_counts()
In []: posteriors,likelihood = hmm.get_posteriors(posc.train.
    seq_list[0])
In []: hmm.update_counts(posc.train.seq_list[0],posteriors)
In []: hmm.sanity_check_counts(posc.train)
```

If you pass this test, then you have all the pieces to implement the EM algorithm. Look at the code for EM algorithm in file sequences/em.py *and check it for yourself.*

```
    def train(self,seq_list,nr_iter=10,smoothing=0,evaluate=
        True):
        if(evaluate):
            ### Evaluate accuracy at initial iteration
            pred = self.model.viterbi_decode_corpus(seq_list.
                seq_list)
            acc = self.model.evaluate_corpus(seq_list.seq_list,
                pred)
        for t in xrange(1,nr_iter):
            #E-Step
            total_likelihood = 0
            self.model.clear_counts(smoothing)
            for seq in seq_list.seq_list:
                posteriors,likelihood = self.model.
                    get_posteriors(seq)
                self.model.update_counts(seq,posteriors)
                total_likelihood += likelihood
            print "Iter: %i - Log Likelihood %f"%(t,-1*math.log
                (total_likelihood))
            #M-Step
            self.model.update_params()

            if(evaluate):
                 ### Evaluate accuracy at this iteration
                pred = self.model.viterbi_decode_corpus(
                    seq_list.seq_list)
                acc = self.model.evaluate_corpus(seq_list.
                    seq_list,pred)
                print "Iter: %i acc %f"%(t,acc)
```

5.2 Part of Speech Induction

In this section we present the Part-of-Speech induction task. Part-of-Speech tags are pre-requisite for many text applications. The task of Part-of-Speech tagging where one is given a labeled training set of words and respective tags is a well studied task with several methods achieving high prediction quality, as we saw in Chapters 2 and 3.

On the other hand the task of Part-of-Speech induction where one does not have access to a labeled corpus is a much harder task with a huge space for improvement. In this case, we are given only the raw text along with sentence boundaries and a predefined number of clusters we can use. This problem can be seen as a clustering problem. We want to cluster words that behave grammatically in the same way on the same cluster. This is a much harder problem.

Formally, the problem setting is the following: we are given a training set $\mathcal{X} = \bar{\mathbf{x}}^1 \ldots \bar{\mathbf{x}}^D$ of D training examples, where each example $\bar{\mathbf{x}} = \mathbf{x}_1 \ldots \mathbf{x}_N$ is a sentence of N words, whose values v are taken from a vocabulary $\mathcal{V}$ of possible word types. We are also given the set of clusters $\mathbf{Y}$ that we are allowed to use. The hidden structure $\bar{\mathbf{y}} = \mathbf{y}_1 \ldots \mathbf{y}_N$ corresponds to a sequence of cluster assignments for each individual word, such that $\mathbf{y}_n = y_l$ with $y_l \in \mathbf{Y}$.

Depending on the task at hand we can pick an arbitrary number of clusters. If the goal is to test how well our method can recover the true pos tags then we should use the same number of clusters as pos tags. On the other hand, if the task is to extract features to be used by other methods we can use a much bigger number of clusters (e.g. 200) to capture correlations not captured by pos tags, like lexical affinity.

Note, however that nothing is said about the identity of each cluster. The model has no preference in assigning cluster 1 to nouns vs cluster 2 to nouns. Given this non-identifiability several metrics have been proposed for evaluation (Reichart and Rappoport, 2009; Haghighi and Klein, 2006; Meilă, 2007; Rosenberg and Hirschberg, 2007). In this class we will use a common and simple metric called **1-Many**, which maps each cluster to majority pos tag that it contains (see Figure 5.1 for an example).

Exercise 5.2 *Run the EM algorithm for part of speech induction:*

```
In []: run readers/pos_corpus.py
In []: posc = PostagCorpus("en",max_sent_len=15,train_sents=
    1000,dev_sents=0,test_sents=0)
In []: run sequences/hmm.py
In []: hmm = HMM(posc)
In []: hmm.initialize_radom()
In []: run sequences/em.py
In []: em = EM(posc,hmm)
In []: em.train(posc.train,nr_iter=20)
```

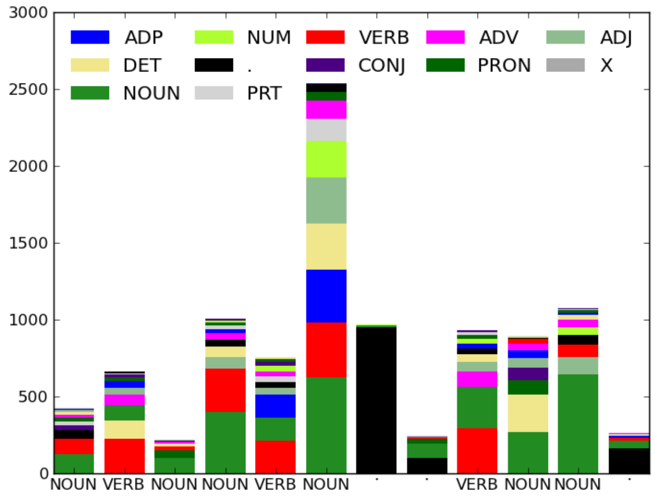

Figure 5.1: Confusion Matrix example. Each cluster is a column. The best tag in each column is represented under the column (1-many) mapping. Each color represents a true Pos Tag.

```
Out []: Init acc 0.335505
Out []: Iter: 1 - Log Likelihood 16.071708
Out []: Iter: 1 acc 0.361960
Out []: Iter: 2 - Log Likelihood 11.212829
Out []: Iter: 2 acc 0.381000
Out []: Iter: 3 - Log Likelihood 11.091918
Out []: Iter: 3 acc 0.387013
Out []: Iter: 4 - Log Likelihood 10.751445
Out []: Iter: 4 acc 0.391222
Out []: Iter: 5 - Log Likelihood 10.046576
Out []: Iter: 5 acc 0.390420
Out []: Iter: 6 - Log Likelihood 9.055178
Out []: Iter: 6 acc 0.391723
Out []: Iter: 7 - Log Likelihood 8.109925
Out []: Iter: 7 acc 0.390420
Out []: Iter: 8 - Log Likelihood 7.497388
Out []: Iter: 8 acc 0.390520
Out []: Iter: 9 - Log Likelihood 7.225907
Out []: Iter: 9 acc 0.393827
Out []: Iter: 10 - Log Likelihood 7.127711
Out []: Iter: 10 acc 0.398236
Out []: Iter: 11 - Log Likelihood 7.105954
Out []: Iter: 11 acc 0.404449
Out []: Iter: 12 - Log Likelihood 7.111193
Out []: Iter: 12 acc 0.406654
Out []: Iter: 13 - Log Likelihood 7.041794
Out []: Iter: 13 acc 0.411264
Out []: Iter: 14 - Log Likelihood 6.958736
Out []: Iter: 14 acc 0.408558
Out []: Iter: 15 - Log Likelihood 6.828692
Out []: Iter: 15 acc 0.407656
Out []: Iter: 16 - Log Likelihood 6.693052
Out []: Iter: 16 acc 0.403848
Out []: Iter: 17 - Log Likelihood 6.670297
Out []: Iter: 17 acc 0.405451
Out []: Iter: 18 - Log Likelihood 6.684892
Out []: Iter: 18 acc 0.408658
Out []: Iter: 19 - Log Likelihood 6.706640
Out []: Iter: 19 acc 0.412166
```

Note: your results may not be the same as in this example since we are using a random start, but the trend should be the same. Also note that in some iterations the likelihood does not go down because of some rounding errors, however the general trend is that likelihood decreases over iterations.

In the previous exercise we used an HMM to do Part-of-Speech induction using 12 clusters (by omission the HMM uses as number of hidden states the one provided by the corpus). A first observation is that the log-likelihood is always increasing as expected. Another observation is that the accuracy goes up from 33% to 41%. Note that normally you will run this algorithm for 200 iterations, we stopped earlier for time constraints. Another observations is that the accuracy is not monotonic increasing, this is because the likelihood is not a perfect proxy for the accuracy. In fact all that likelihood is measuring are co-occurrences of words in the corpus; it has no idea of pos tags. The fact we are improving derives from the fact that language is not random but follows some specific hidden patterns. In fact this patterns are what true pos-tags try to capture. A final observation is that the performance is really bad compared to the supervised scenario, so there is a lot of space for improvement. The actual state of the art is around 71% for fully unsupervised (Graça, 2010; Berg-Kirkpatrick et al., 2010) and 80% (Das and Petrov, 2011) using parallel data and information from labels in the other language.

Looking at Figure 5.1 shows the confusion matrix for this particular example. A first observation is that most clusters are mapped to nouns, verbs or punctuation. This is a none fact since there are many more nouns and verbs than any other tags. Since maximum likelihood prefers probabilities to be uniform (Imagine two parameters. In one setting both have value 0.5 so the likelihood will be 0.5*0.5 = 0.25, while in the other case one as 0.1 and 0.9 so the maximum likelihood is 0.09). Several approaches have been proposed to address this problem under moving towards a Bayesian setting or using Posterior

Regularization (Johnson, 2007; Graça et al., 2009) more about this later today. Part-of-Speech induction is a very active field of research, in fact in the last two ACL conferences (Association for Computational Linguistics) the short paper award (2010) and the best paper award (2011) were about this topic (Lamar et al., 2010; Das and Petrov, 2011).

Bibliography

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

Bertsekas, D., Homer, M., Logan, D., and Patek, S. (1995). *Nonlinear programming*. Athena Scientific.

Bishop, C. (2006). *Pattern recognition and machine learning*, volume 4. Springer New York.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 440.

Bottou, L. (1991). *Une Approche Theorique de l'Apprentissage Connexionniste: Applications a la Reconnaissance de la Parole*. PhD thesis.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge Univ Pr.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

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

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL*.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the National Conference on Artificial Intelligence*, pages 598–603. Citeseer.

Charniak, E. and Elsner, M. (2009). EM works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 148–156. Association for Computational Linguistics.

Charniak, E., Johnson, M., Elsner, M., Austerweil, J., Ellis, D., Haxton, I., Hill, C., Shrivaths, R., Moore, J., Pozar, M., et al. (2006). Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 168–175. Association for Computational Linguistics.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*, volume 119. The MIT press.

Chu, Y. J. and Liu, T. H. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Clark, A. (2003). Combining distributional and morphological information for part of speech induction. In *Proc. EACL*.

Cohen, S., Gimpel, K., and Smith, N. (2008). Logistic normal priors for unsupervised probabilistic grammar induction. In *In NIPS*. Citeseer.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Cover, T., Thomas, J., Wiley, J., et al. (1991). *Elements of information theory*, volume 6. Wiley Online Library.

Covington, M. (1990). Parsing discontinuous constituents in dependency grammar. *Computational Linguistics*, 16(4):234–236.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online Passive-Aggressive Algorithms. *JMLR*, 7:551–585.

Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292.

Das, D. and Petrov, S. (2011). Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*, volume 2. Wiley New York.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Eisner, J. and Satta, G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*.

Finkel, J., Kleeman, A., and Manning, C. (2008). Efficient, feature-based, conditional random field parsing. *Proceedings of ACL-08: HLT*, pages 959–967.

Graça, J. (2010). *Posterior Regularization Framework: Learning Tractable Models with Intractable Constraints*. PhD thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico.

Graça, J., Ganchev, K., Pereira, F., and Taskar, B. (2009). Parameter vs. posterior sparisty in latent variable models. In *Proc. NIPS*.

Haghighi, A. and Klein, D. (2006). Prototype-driven learning for sequence models. In *Proc. HTL-NAACL*. ACL.

Henderson, J. (2003). Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 24–31. Association for Computational Linguistics.

Hopcroft, J., Motwani, R., and Ullman, J. (1979). *Introduction to automata theory, languages, and computation*, volume 3. Addison-wesley Reading, MA.

Huang, L. and Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proc. of ACL*, pages 1077–1086.

Hudson, R. (1984). *Word grammar*. Blackwell Oxford.

Jaynes, E. (1982). On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers.

Johnson, M. (1998). Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

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

Klein, D. and Manning, C. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Klein, D. and Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

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

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proc. of ACL*, pages 1–11.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Proc. EMNLP*.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *EMNLP*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Procs. of ICML*, pages 282–289.

Lamar, M., Maron, Y., Johnson, M., and Bienenstock, E. (2010). SVD and clustering for unsupervised POS tagging. In *Proceedings of the ACL 2010 Conference: Short Papers*, pages 215–219, Uppsala, Sweden. Association for Computational Linguistics.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, UK.

Manning, C. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 59. MIT Press.

Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Martins, A. F. T., Smith, N. A., and Xing, E. P. (2009). Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598. Citeseer.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of CoNLL*.

McDonald, R. and Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

McDonald, R. T., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Meilă, M. (2007). Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895.

Melčuk, I. (1988). *Dependency syntax: theory and practice*. State University of New York Press.

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational linguistics*, 20(2):155–171.

Mitchell, T. (1997). *Machine learning*.

Nivre, J. (2009). Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 351–359. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006). Labeled pseudo-projective dependency parsing with support vector machines. In *Procs. of CoNLL*.

Nocedal, J. and Wright, S. (1999). *Numerical optimization*. Springer verlag.

Pérez, F. and Granger, B. E. (2007). IPython: a System for Interactive Scientific Computing. *Comput. Sci. Eng.*, 9(3):21–29.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2008a). Discriminative log-linear grammars with latent variables. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1153–1160, Cambridge, MA. MIT Press.

Petrov, S. and Klein, D. (2008b). Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 867–876, Honolulu, Hawaii. Association for Computational Linguistics.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *In Proc. IEEE*, 77(2):257–286.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1):151–175.

Reichart, R. and Rappoport, A. (2009). The NVI clustering evaluation measure. In *Proc. CONLL*.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, pages 410–420.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA.

Schütze, H. (1995). Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148. Morgan Kaufmann Publishers Inc.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*.

Shannon, C. (1948). A mathematical theory of communication. *Bell Syst. Tech. Journ.*, 27(379):623.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. CUP.

Smith, D. A. and Eisner, J. (2008). Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Smith, D. A. and Smith, N. A. (2007). Probabilistic models of nonprojective dependency trees. In *Proc. EMNLP-CoNLL*.

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

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

Smith, N. A. and Eisner, J. (2006). Annealing structural bias in multilingual weighted grammar induction. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 569–576, Morristown, NJ, USA. Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proc. of CoNLL*.

Tarjan, R. (1977). Finding optimum branchings. *Networks*, 7(1):25–36.

Taskar, B., Klein, D., Collins, M., Koller, D., and Manning, C. (2004). Max-margin parsing. In *Proc. EMNLP*, pages 1–8.

Tesnière, L. (1959). *Eléments de syntaxe structurale*. Libraire C. Klincksieck.

Tutte, W. (1984). *Graph Theory*. Addison-Wesley, Reading, MA.

Vapnik, N. V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.