An Empirical Analysis of Optimization for Max-Margin NLP

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick and **Dan Klein**
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{jkk,tberg,klein}@cs.berkeley.edu

Abstract

Despite the convexity of structured max-margin objectives (Taskar et al., 2004; Tsochantaridis et al., 2004), the many ways to optimize them are not equally effective in practice. We compare a range of online optimization methods over a variety of structured NLP tasks (coreference, summarization, parsing, etc) and find several broad trends. First, margin methods do tend to outperform both likelihood and the perceptron. Second, for max-margin objectives, primal optimization methods are often more robust and progress faster than dual methods. This advantage is most pronounced for tasks with dense or continuous-valued features. Overall, we argue for a particularly simple online primal subgradient descent method that, despite being rarely mentioned in the literature, is surprisingly effective in relation to its alternatives.

1 Introduction

Structured discriminative models have proven effective across a range of tasks in NLP including tagging (Lafferty et al., 2001; Collins, 2002), reranking parses (Charniak and Johnson, 2005), and many more (Taskar, 2004; Smith, 2011). Common approaches to training such models include margin methods, likelihood methods, and mistake-driven procedures like the averaged perceptron algorithm. In this paper, we primarily consider the relative empirical behavior of several online optimization methods for margin-based objectives, with secondary attention to other approaches for calibration.

It is increasingly common to train structured models using a max-margin objective that incorporates a loss function that decomposes in the same way as the dynamic program used for inference (Taskar, 2004). Fortunately, most structured margin objectives are convex, so a range of optimization methods with similar theoretical properties are available – in short, any of these methods will work in the end. However, in practice, how fast each method converges varies across tasks. Moreover, some of the most popular methods more loosely associated with the margin objective, such as the MIRA algorithm (Crammer and Singer, 2003) or even the averaged perceptron (Freund and Schapire, 1999) are not global optimizations and can have different properties.

We analyze a range of methods empirically, to understand on which tasks and with which feature types, they are most effective. We modified six existing, high-performance, systems to enable loss-augmented decoding, and trained these models with six different methods. We have released our learning code as a Java library.[1] Our results provide support for the conventional wisdom that margin-based optimization is broadly effective, frequently outperforming likelihood optimization and the perceptron algorithm. We also found that directly optimizing the primal structured margin objective based on subgradients calculated from single training instances is surprisingly effective, performing consistently well across all tasks.

2 Learning Algorithms

We implemented a range of optimization methods that are widely used in NLP; below we categorize them into margin, likelihood, and perceptron-like methods. In each case, we used a structured loss function, modified to suit each task. In general, we focus on online methods because of their substantial speed advantages, rather than algorithms such as LBFGS (Liu and Nocedal, 1989) or batch Exponentiated Gradient (Collins et al., 2008).

[1] http://nlp.cs.berkeley.edu/software.shtml

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 273–279,
Lisbon, Portugal, 17-21 September 2015.

Algorithm 1 The Online Primal Subgradient Algorithm with ℓ_1 or ℓ_2 regularization, and sparse updates

Parameters:

- iters — Number of iterations
- C — Regularization constant (10^{-1} to 10^{-8})
- η — Learning rate (10^0 to 10^{-4})
- δ — Initializer for q (10^{-6})

```
w = 0        Weight vector
q = δ        Cumulative squared gradient
u = 0        Time of last update for each weight
n = 0        Number of updates so far
for iter ∈ [1, iters] do
    for batch ∈ data do
        Sum gradients from loss-aug. decodes
        g = 0
        for (x_i, y_i) ∈ batch do
            y = argmax_{y'∈Y(x_i)} [SCORE(y') + L(y', y_i)]
            g += (f(y) − f(y_i))
        Update the active features
        q += g^2        Element-wise square
        n += 1
        for f ∈ nonzero features in g do
            w_f = UPDATE-ACTIVE(w_f, g_f, q_f)
            u_f = n
```

The AdaGrad update

function UPDATE-ACTIVE(w, g, q)

$$\textbf{return } \frac{w\sqrt{q}-\eta g}{\eta \mathrm{C}+\sqrt{q}} \qquad [\ell_2]$$

$$\mathrm{d} = |w - \frac{\eta}{\sqrt{q}}g| - \frac{\eta}{\sqrt{q}}\mathrm{C} \qquad [\ell_1]$$

$$\textbf{return } \mathrm{sign}(w - \frac{\eta}{\sqrt{q}}g) \cdot \max(0, \mathrm{d}) \qquad [\ell_1]$$

Functions only needed for sparse updates

A single update equivalent to a series of AdaGrad updates where the weight's subgradient was zero

function UPDATE-CATCHUP(w, q, t)

$$\textbf{return } w\left(\frac{\sqrt{q}}{\eta \mathrm{C}+\sqrt{q}}\right)^t \qquad [\ell_2]$$

$$\textbf{return } \mathrm{sign}(w) \cdot \max(0, |w| - \frac{\eta \mathrm{C}}{\sqrt{q}}t) \qquad [\ell_1]$$

Compute $\mathbf{w}^\top \mathbf{f}(y')$, but for each weight, apply an update to catch up on the steps in which the gradient for that weight was zero

```
function SCORE(y')
    s = 0
    for f ∈ f(y') do
        w_f = UPDATE-CATCHUP(w_f, q_f, n − u_f)
        u_f = n
        s += w_f
    return s
```

Note: To implement without the sparse update, use SCORE $= \mathbf{w}^\top \mathbf{f}(y')$, and run the update loop on the left over all features. Also, for comparison, to implement perceptron, remove the sparse update and use UPDATE-ACTIVE $=$ **return** $w + g$.

2.1 Margin

Cutting Plane (Tsochantaridis et al., 2004) Solves a sequence of quadratic programs (QP), each of which is an approximation to the dual formulation of the margin-based learning problem. At each iteration, the current QP is refined by adding additional active constraints. We solve each approximate QP using Sequential Minimal Optimization (Platt, 1999; Taskar et al., 2004).

Online Cutting Plane (Chang and Yih, 2013) A modified form of cutting plane that only partially solves the QP on each iteration, operating in the dual space and optimizing a single dual variable on each iteration. We use a variant of Chang and Yih (2013) for the L_1 loss margin objective.

Online Primal Subgradient (Ratliff et al., 2007) Computes the subgradient of the margin objective on each instance by performing a loss-augmented decode, then uses these instance-wise subgradients to optimize the global objective using AdaGrad (Duchi et al., 2011) with either L_1 or L_2 regularization. The simplest implementation of AdaGrad touches every weight when doing the update for a batch. To save time, we distinguish between two different types of update. When the subgradient is nonzero, we apply the usual update. When the subgradient is zero, we apply a numerically equivalent update later, at the next time the weight is queried. This saves time, as we only touch the weights corresponding to the (usually sparse) nonzero directions in the current batch's subgradient. Algorithm 1 gives pseudocode for our implementation, which was based on Dyer (2013).

2.2 Likelihood

Stochastic Gradient Descent The built-in training method for many of the systems was softmax-margin likelihood optimization (Gimpel and Smith, 2010) via subgradient descent with either AdaGrad or AdaDelta (Duchi et al., 2011; Zeiler, 2012). We include results with each system's default settings as a point of comparison.

2.3 Mistake Driven

Averaged Perceptron (Freund and Schapire, 1999; Collins, 2002) On a mistake, weights for features on the system output are decremented and weights for features on the gold output are incremented. Weights are averaged over the course of training, and decoding is not loss-augmented.

Margin Infused Relaxed Algorithm (Crammer and Singer, 2003) A modified form of the perceptron that uses loss-augmented decoding and makes the smallest update necessary to give a margin at least as large as the loss of each solution. MIRA is generally presented as being related to the perceptron because it does not explicitly optimize a global objective, but it also has connections to margin methods, as explored by Chiang (2012). We consider one-best decoding, where the quadratic program for determining the magnitude of the update has a closed form.

3 Tasks and Systems

We considered tasks covering a range of structured output spaces, from sequences to non-projective trees. Most of the corresponding systems use models designed for likelihood-based structured prediction. Some use sparse indicator features, while others use dense continuous-valued features.

Named Entity Recognition This task provides a case of sequence prediction. We used the NER component of Durrett and Klein (2014)'s entity stack, training it independently of the other components. We define the loss as the number of incorrectly labelled words, and train on the CoNLL 2012 division of OntoNotes (Pradhan et al., 2007).

Coreference Resolution This gives an example of training when there are multiple gold outputs for each instance. The system we consider uses latent links between mentions in the same cluster, marginalizing over the possibilities during learning (Durrett and Klein, 2013). Since the model decomposes across mentions, we train by treating them as independent predictions with multiple gold outputs, comparing the inferred link with the gold link that is scored highest under the current model. We use the system's weighted loss function, and the same data as for NER.

Constituency Parsing We considered two different systems. The first uses only sparse indicator features (Hall et al., 2014), while the second is parameterized via a neural network and adds dense features derived from word vectors (Durrett and Klein, 2015).[2] We define the loss as the number of incorrect rule productions, and use the standard Penn Treebank division (Marcus et al., 1993).

Dependency Parsing We used the first-order MST parser in two modes, Eisner's algorithm for projective trees (Eisner, 1996; McDonald et al., 2005b), and the Chu-Liu-Edmonds algorithm for non-projective trees (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005a). The loss function was the number of arcs with an incorrect parent or label, and we used the standard division of the English Universal Dependencies (Agić et al., 2015). The built-in training method for MST parser is averaged, 1-best MIRA, which we include for comparison purposes.

Summarization With this task, we explore a case in which there is relatively little training data and the model uses a small number of dense features. The system uses a linear model with features considering counts of bigrams in the input document collection. The system forms the output summary by selecting a subset of the sentences in the input collection that does not exceed a fixed word-length limit (Berg-Kirkpatrick et al., 2011). Inference involves solving an integer linear program, the loss function is bigram recall, and the data is from the TAC shared tasks (Dang and Owczarzak, 2008; Dang and Owczarzak, 2009).

3.1 Tuning

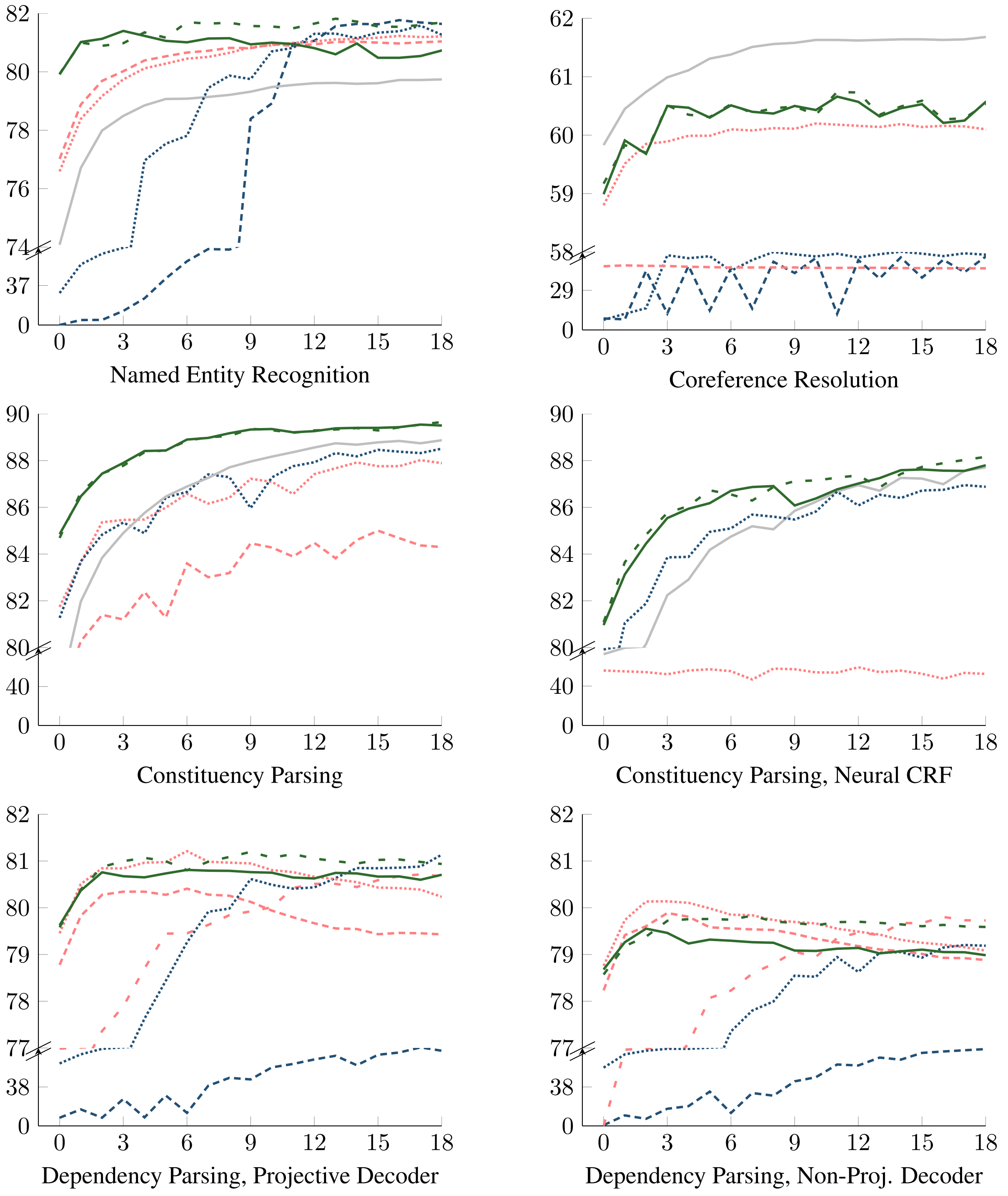
For each method we tuned hyperparameters by considering a grid of values and measuring dev set performance over five training iterations, except for constituency parsing, where we took five measurements, 4k instances apart. For the cutting plane methods we cached constraints in memory to save time, but the memory cost was too great to run batch cutting plane on constituency parsing (over 60 Gb), and so is not included in the results.

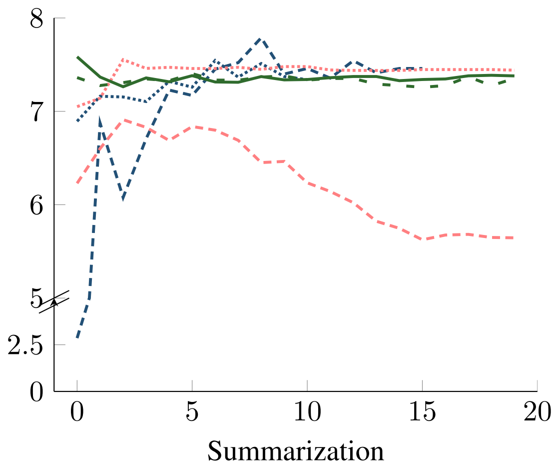
4 Observations

From the results in Figure 1 and during tuning, we can make several observations about these optimization methods' performance on these tasks.

Observation 1: Margin methods generally perform best As expected given prior work, margin methods equal or surpass the performance of likelihood and perceptron methods across almost all of these tasks. Coreference resolution is an exception, but that model has latent variables that likelihood may treat more effectively,

[2] Our results are slightly lower as we save time by only using the dense features and a reduced n-gram context.





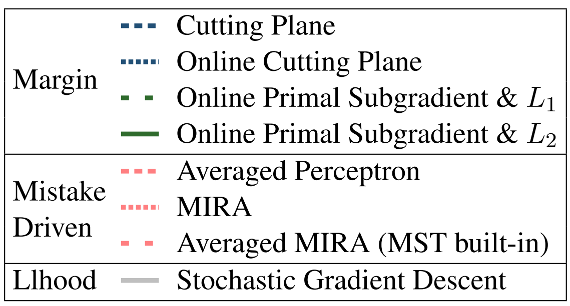


Figure 1: Variation in dev set performance (y) across training iterations (x). To show all variation, the scale of the y-axis changes partway, as indicated. Lines that stop early had converged.

	Time per iteration relative to averaged perceptron						
Method	NER	Coref	Span Parser	Neural Parser	MST Proj.	MST Non-Proj.	Summ.
AP	1.0	1.0	1.0	-	1.0	1.0	1.0
MIRA	1.9	1.0	1.0	1.0	1.0	1.0	1.0
CP	60.8	2.7	-	-	6.8	8.4	0.6
OCP	2.7	1.7	0.9	0.9	1.5	1.6	1.1
OPS	3.9	1.3	1.1	1.0	1.8	2.0	0.9
Decoding	0.6	0.2	0.9	0.7	0.7	0.6	0.7

Table 1: Comparison of time per iteration relative to the perceptron (or MIRA for the Neural Parser). Decoding shows the time spent on inference. Times were averaged across the entire run. OPS uses batch size 10 for NER to save time, but performs just as well as with batch size 1 in Figure 1.

and has a weighted loss function tuned for likelihood (softmax-margin).

Observation 2: Dual cutting plane methods appear to learn more slowly Both cutting plane methods took more iterations to reach peak performance than the other methods. In addition, for batch cutting plane, accuracy varied so drastically that we extended tuning to ten iterations, and even then choosing the best parameters was sometimes difficult. Table 1 shows that the online cutting plane method did take slightly less time per iteration than OPS, but not enough to compensate for the slower learning rate.

Observation 3: Learning with real-valued features is difficult for perceptron methods Learning models for tasks such as NER, which are driven by sparse indicator features, often roughly amounts to tallying the features that are contrastively present in correct hypotheses. In such cases, most learning methods work fairly well. However, when models use real-valued features, learning may involve determining a more delicate balance between features. In the models we consider that have real-valued features, summarization and parsing with a neural model, we can see that perceptron methods indeed have difficulty.[3]

Observation 4: Online Primal Subgradient is robust and effective All of the margin based methods, and gradient descent on likelihood, require tuning of a regularization constant and a step size (or convergence requirements for SMO). The dual methods were particularly sensitive to these hyperparameters, performing poorly if they were not chosen carefully. In contrast, performance for the primal methods remained high over a broad range of values.

Our implementation of sparse updates for AdaGrad was crucial for high-speed performance, decreasing time by an order of magnitude on tasks with many sparse features, such as NER and dependency parsing.

Observation 5: Other minor properties We found that varying the batch size did not substantially impact performance after a given number of decodes, but did enable a speed improvement as decoding of multiple instances can occur in parallel. Increasing batch sizes leads to a further improvement to OPS, as overall there are fewer updates per iteration. For some tasks, re-tuning the step size was necessary when changing batch size.

5 Conclusion

The effectiveness of max-margin optimization methods is widely known, but the default choice of learning algorithm in NLP is often a form of the perceptron (or likelihood) instead. Our results illustrate some of the pitfalls of perceptron methods and suggest that online optimization of the max-margin objective via primal subgradients is a simple, well-behaved alternative.

6 Acknowledgments

We would like to thank Greg Durrett for assistance running his code, Adam Pauls for advice on dual methods, and the anonymous reviewers for their helpful suggestions. This work was supported by National Science Foundation grant CNS-1237265, Office of Naval Research MURI grant N000140911081, and a General Sir John Monash Fellowship to the first author. Opinions, findings, conclusions and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of sponsors.

[3]For the neural parser, the perceptron took a gradient step for each mistake, but this had dismal performance.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA, June.

Ming-Wei Chang and Wen-Tau Yih. 2013. Dual coordinate descent algorithms for efficient large margin structured prediction. *Transactions of the Association for Computational Linguistics*, 1:207–218.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, June.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187, April.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, pages 1396–1400.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, June.

Michael Collins. 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.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, March.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Text Analysis Conference*.

Hoa Trang Dang and Karolina Owczarzak. 2009. Overview of the TAC 2009 summarization track. In *Text Analysis Conference*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA, October.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. volume 2, pages 477–490.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China, July.

Chris Dyer. 2013. Notes on AdaGrad. Technical report, Carnegie Mellon University, June.

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

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, pages 340–345.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, December.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland, USA, June.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, December.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, USA, June.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October.

John C. Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press.

Sameer Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in OntoNotes. In *Proceedings of the International Conference on Semantic Computing*, pages 446–453, September.

Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. 2007. (Online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain, July.

Ben Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 104–112, July.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.