Parsing and Machine learning

http://www-rohan.sdsu.edu/~gawron/aisem

Graph-based dependency parsing

Jean Mark Gawron

San Diego State University, Department of Linguistics

2010-08-19

Overview

Dependency parsing

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.
- Two basic types of dependency model

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.
- Two basic types of dependency model
  - Transition-based models: parameterized over parser transitions (in a sense very like that of an LR parser or a shift-reduce parser).

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.
- Two basic types of dependency model
  - Transition-based models: parameterized over parser transitions (in a sense very like that of an LR parser or a shift-reduce parser).
  - Graph-based models: parameterized over substructures of a dependency tree:

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.
- Two basic types of dependency model
  - Transition-based models: parameterized over parser transitions (in a sense very like that of an LR parser or a shift-reduce parser).
  - Graph-based models: parameterized over substructures of a dependency tree:
    1. Arc-factored: model limited to predicting individual arcs in graph

Dependency parsing

Given a string S in some language, find the best dependency graph between the words. (Kübler et al. 2009)

- Assumption: "Best" corresponds to a score assigned to each dependency trees by a dependency model.
- Two basic types of dependency model
  - Transition-based models: parameterized over parser transitions (in a sense very like that of an LR parser or a shift-reduce parser).
  - Graph-based models: parameterized over substructures of a dependency tree:
    1. Arc-factored: model limited to predicting individual arcs in graph
    2. Arc-factored +: model includes arity: How likely is a given word to have a fixed number of dependents?

Graph terminology

A labeled, directed graph $G = (V, A)$ consists of

1. V: a set of nodes (vertices). Never mind what they are.
2. A: a set of arcs

$$A \subseteq V \times R \times V$$

where R is the set of labels.

G is directed because:

$$(\text{news}, \text{NMOD}, \text{Economic}) \neq (\text{Economic}, \text{NMOD}, \text{news})$$

We say G is labeled digraph (labeled, directed graph).

Simple score functions

Score functions for $G = (V, A)$:

Parameters	Score(G)	Type
$\lambda_{(w_i,r,w_j)}$	$\sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)}$	Arc-factored
$\lambda_{(w_i,r,w_j)}$ α_{w_i}	$\sum_{(w_i,r,w_j)\in A} \lambda_{w_i,r,w_j})+$ $\sum_{w_i\in V} \alpha_{w_i}(\lvert \text{dpdnts}(w_i, A)\rvert)$	Arc-factored$^+$

where α_{w_i} assigns a score to the number of dependent w_i has.

Chart parser w/ graph-based model

Before adding a span s_0 to the left or right chart we check for the existence of a span s_1 with the same signature:

$$\begin{array}{ll} \textit{If} & \textit{score}(\textit{Graph}(s_0)) > \textit{score}(\textit{Graph}(s_1)) \\ \textit{Then} & \textit{Add-to-chart}(s_0) \end{array}$$

We will compute the Viterbi parse relative to the score function.

Restrictions on dependency graphs

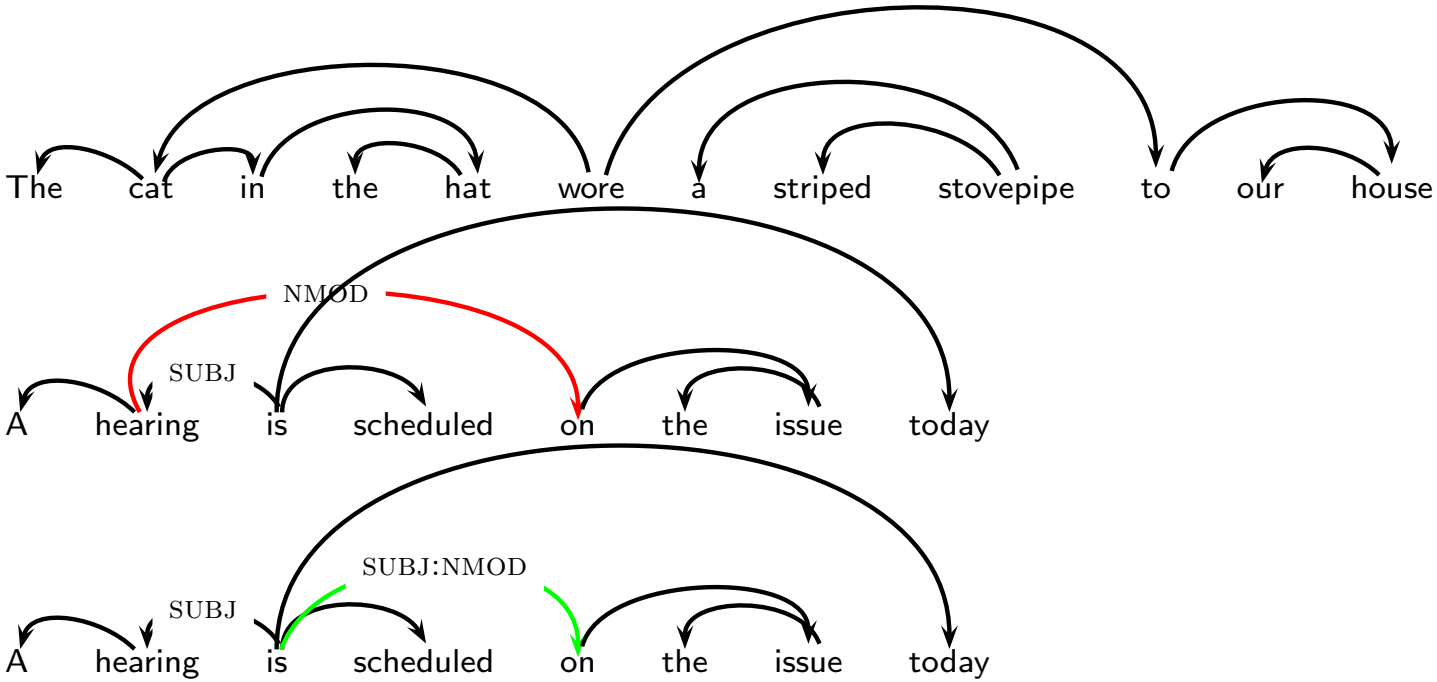
Let G = (A,V) be a dependency graph for sentence S = $w_i w_2, \ldots, w_n$.

1. G is a tree (no child has more than one parent).
2. If $(w_i, r, w_j) \in$ A, then $(w_i, r', w_j) \notin$ A, for all $r' \neq r$.
   We say: G is NOT a multi digraph.
3. Rooted: No word is the parent of w_0.
4. Connected: For every pair of words w_i and w_j, $w_i \leftrightarrow^* w_j$.
5. Acyclic: If $w_i \rightarrow^* w_j$, then it is not the case that $w_j \rightarrow^* w_i$,
6. Projectivity: All arcs are projective.

Where a projective arc (w_i, r, w_j) satisfies the following constraining:

For all words w_k in between w_i and w_j, $w_i \rightarrow^* w_k$.

Projectivity



Projectivization of non-projective dependency tree (Nivre and Nilsson 2005)

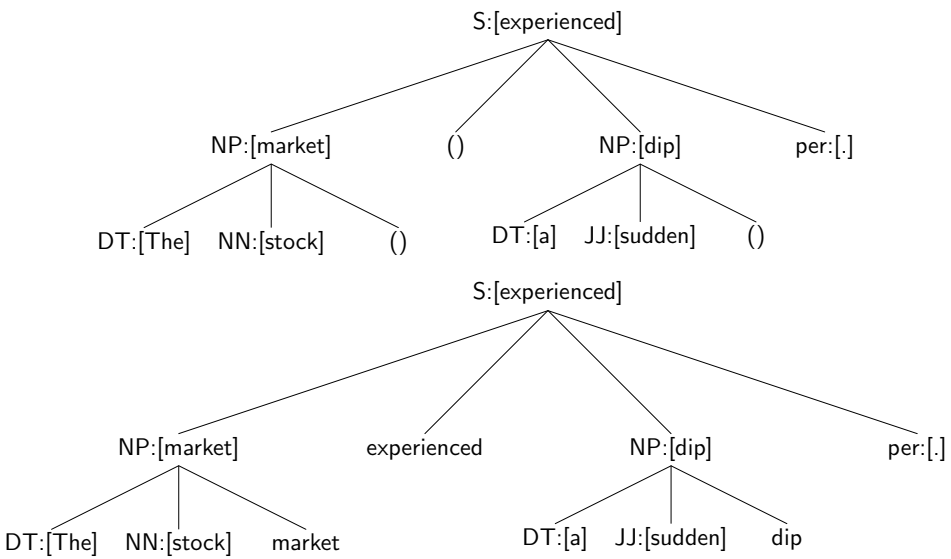
Nested property: For all words w_i the set of words $\{w_j \mid w_i \rightarrow^* w_j\}$ is a contiguous subsequence of S. (Equivalent to projectivity when assuming an endword Root).

Projectivity and word order

The non-projective analysis on the previous slide is

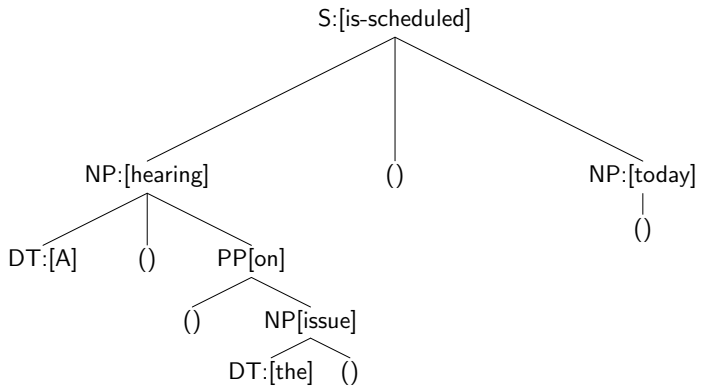
- Acyclic;
- We can still add ROOT on the left and have a connected graph;
- It is still a tree (every child has a unique parent).

Projective trees can be made into word-order respecting trees, which is why context-free parsing algorithms can construct them:

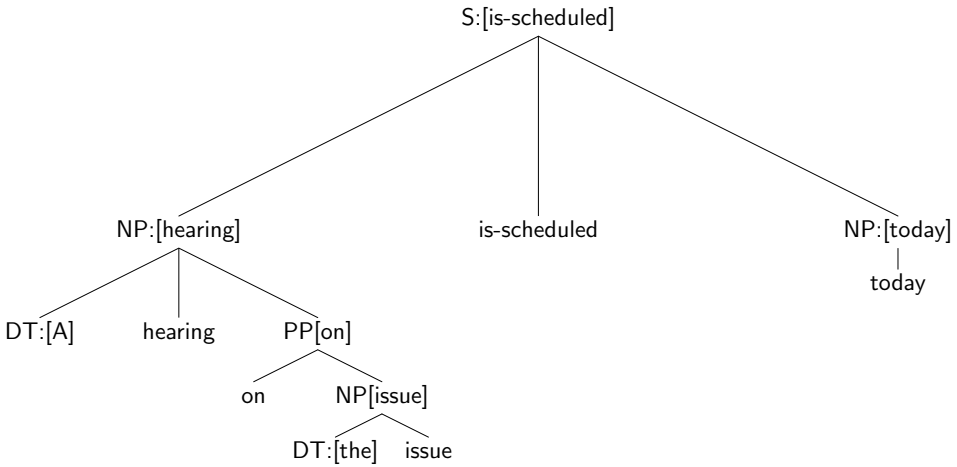


Non projective tree

A non projective dependency tree has dependency relations that are not CONSISTENT with the word order



Context-free parsing algorithms no longer apply.



On the other hand, while it is common cross-linguistically, non projectivity is constrained in many ways.

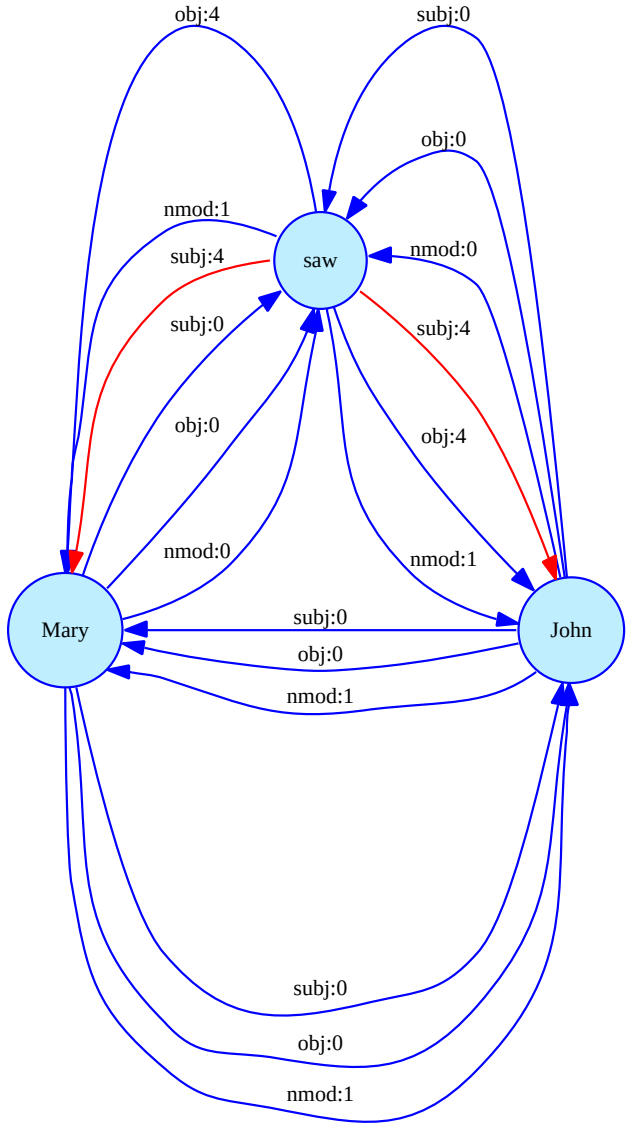
$\neq$ *A hearing is scheduled on the issue today.*

Maximum spanning tree (MST)

Consider the dependency hypothesis graph (DH) consisting of all arcs in R between all the words in S.

S = *John saw Mary*
R = {subj, obj, nmod}

The subgraph in red is a tree. It is spanning because it includes all the nodes in DH. It also has the highest score such a tree can have. It is a maximum spanning tree for DH. (MST)



Parsing as an MST problem

The following assumptions are needed

- No projectivity
- Arc factored parsing model (score is based exclusively on arc weights). Note arc-factored+ won't work.
- We reduce finding the MST for a labeled node to the problem of finding the MST for an unlabeled graph, by only allowing ONE directed arc between any pair of nodes (w_i, w_j):

$$\lambda_{w_i,w_j} = \max_r \lambda_{w_i,\, r,\, w_j}$$

- Note: The new unlabeled graph has order $|\mathrm{R}|\ n^2$ parameters.
- Finding the MST is also known as the maximum arborescence problem.

Chu-Liu-Edmonds

Chu and Liu (1965), Edmonds (1968), McDonald et al. (2005)b

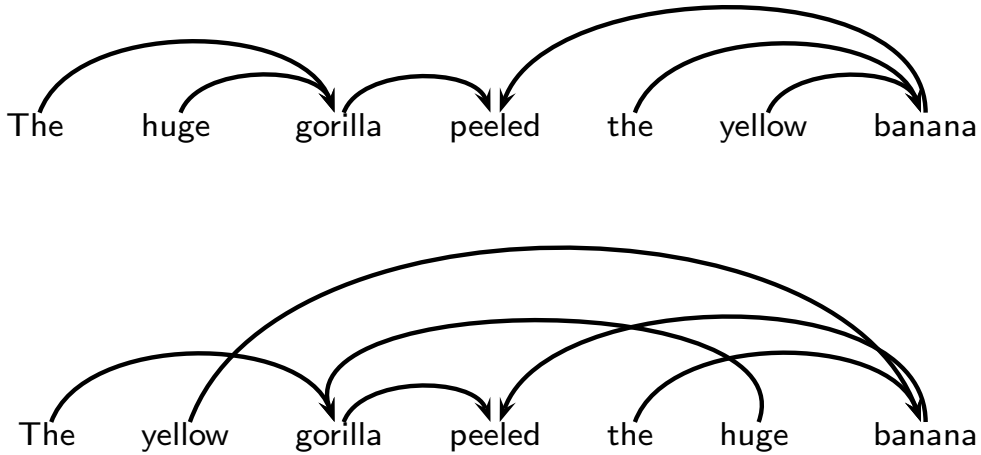
- The Chu-Liu-Edmonds algorithm finds the maximum spanning tree of a graph G.
- The algorithm is basically $O(n^3)$ but an improvement to $O(n^2)$ is possible for dense complete graphs like our Dependency Hypothesis graph (Tarjan 1977).
- This is better efficiency than projective parsing algorithms like Eisner's. (Non projective dependency parsing is easier than projective dependency parsing)
- McDonald et al. (2005)b implement a version of the algorithm as a dependency parser using an arc-factored model.

The potential downside

- Consider a pure word association model, for example, a probabilistic arc-factored model:

$$\mathrm{P}(w_j,\ \textsc{nmod},\ w_i) \approx \frac{\mathrm{count}((w_i,\ \textsc{nmod},\ w_j))}{\mathrm{count}(w_i)}$$

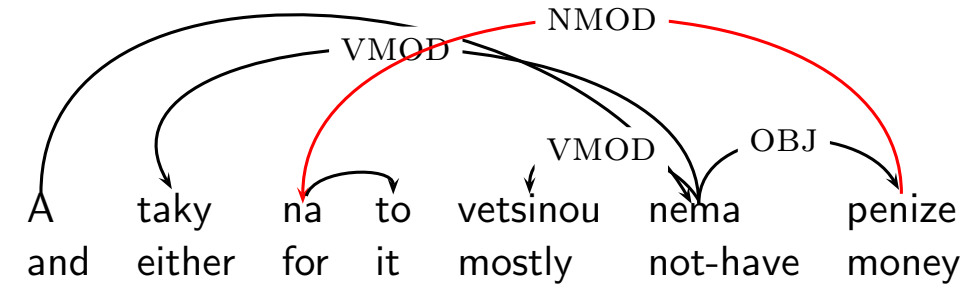
1. The yellow gorilla peeled the huge banana.
2. The yellow banana peeled the huge gorilla.
3. The huge banana peeled the yellow gorilla.



McDonald et al. (2005)b experiment

Parsing Czech, a free word order language with many non projective dependencies.

- On average 23% of training, development, and test sets had at least one non-projective dependency.
- Less than 2% of edge actually are non-projective.
- Non-projective edge-handling has relatively little impact on dependency accuracy. (More on feature set later).



McDonald et al. (2005)b results

	Czech	English	Parsing algorithm
Collins 1999	82.8		Projective
Nivre Nillson 2005	80.0		Pseudoprojective
McDonald 2005	83.3	90.9	Eisner
Single best MIRA	84.1	90.2	Chu Liu Edmonds
Factored MIRA	84.4	90.2	Chu Liu Edmonds

Success with Czech shows that learner sets feature weights so as to disfavor features over unlikely nonprojective edges.

Feature set

7×10^6 feats (English), 1.35×10^7 (Czech).

Uni-gram features
p-word, p-pos
p-word
p-pos
c-word, c-pos
c-word
c-pos

Bi-gram features
p-word, p-pos, c-word, c-pos
p-pos, c-word, c-pos
p-word, c-word, c-pos
p-word, c-pos, c-pos
p-word, p-pos, c-word
p-word, c-word
p-pos, c-pos

Between feats
p-word, b-pos, c-pos
Surround feats (big diff!)
p-pos, p-pos+1, c-pos-1, c-pos
p-pos-1, p-pos, c-pos-1, c-pos
p-pos, p-pos+1, c-pos, c-pos+1
p-pos-1, p-pos, c-pos, c-pos+1

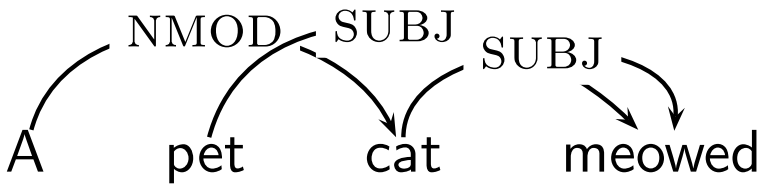
- All features conjoined with the direction of attachment and the distance between p and c.
- Arc-factored: For any 2 graphs G, G' $\in dt(\mathrm{S})$, and any words w_i, w_j in S:

$$\mathrm{Score}_{\mathrm{G}}(\mathrm{S}, (w_i, r, w_j)) = \mathrm{Score}_{\mathrm{G}'}(\mathrm{S}, (w_i, r, w_j))$$

- The score of a dependency arc in G depends only on the words in the sentence and parent and child words. In particular, it does not depend on the presence or absence of any other arcs in G.

Arc-factored+

The McDonald et al. (2005)a model will not accommodate scores with arity (Arc-factored +)



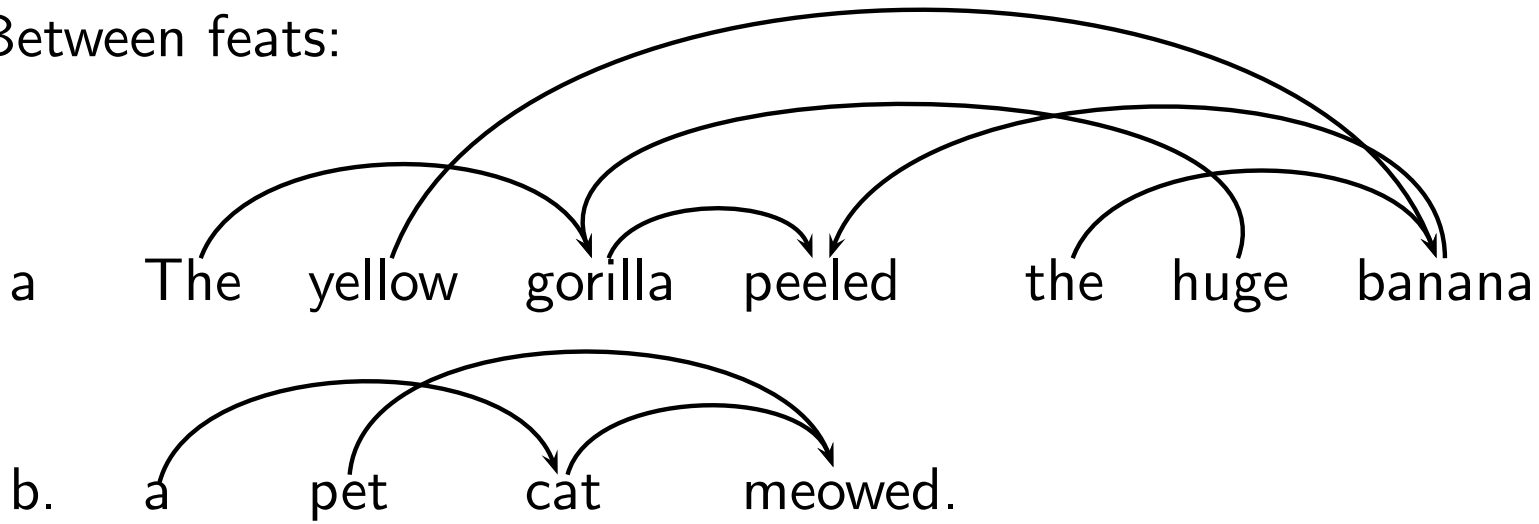
The model could be extended to use such features:

1. It would slow the Eisner parser by some G^2 factor.
2. it would no longer be possible to use Chu Liu Edmonds as a parser, since the score of an arc is no longer a fixed property determined by the two nodes it connects.

Does exploring non-projective trees hurt?

- Yes, for English: Eisner parser outperforms Chu-Liu-Edmonds parser (90.9 vs. 90.2)
- Nevertheless, the CLE parser does VERY well on English.
- A lot of the work of enforcing arity or projectivity can be done by the Surround feats and Between feats:

	English	Czech
Eisner	90.9	83.3
CLE	90.2	84.1



Linguistic interest

- A coherent model in which projectivity is an emergent property: Projectivity resides in the Eisner parser, not the grammar (the weights in w).
- Relaxing projectivity still does not result in chaos. (The CLE parser still does well on English).

Learning

Given a linear parsing model

- Γ *a set of constraints defining permissible dependency trees.*
- λ *Linear model parameters including a weight vector* **w** *and a feature function* **f**.
- h *a parsing algorithm that returns a permissible dependency tree of any sentence S.*

We want to find a set of parameters λ such that our parsing algorithm h chooses the highest scoring graph for S:

$$h(S, \Gamma, \lambda) = \arg\max_{G} \sum_{(w_j, r, w_i) \in G} \lambda_{(w_j, r, w_i)}$$

We will leave out Γ below.

Perceptron

N is the number of rounds of training, $|\mathcal{D}|$ is the size of the training set:

$\textsc{Perceptron}(\mathcal{D}, N)$

1 $\mathbf{w} = 0$
2 **for** $n \leftarrow 1$ **to** N
3 **do for** $d \leftarrow 1$ **to** $|\mathcal{D}|$
4 **do** Let G' $= h(S_d, \lambda = (\mathbf{w}, f))$
5 **if** G' $\neq G_d$
6 **then** $\mathbf{w} = \mathbf{w} + \sum_{(w_i, r, w_j) \in G_d} \mathbf{f}(w_i, r, w_j) - \sum_{(w_i, r, w_j) \in G'} \mathbf{f}(w_i, r, w_j)$
7 **return w**

Parse score with Perceptron model

The perceptron yields a linear model:

- A model parameter $\lambda_{(w_i,r,w_j)}$:

$$\lambda_{(w_i,r,w_j)} = \mathbf{w} \cdot f(w_i, r, w_j)$$

- Our parser will choose parses based on the sum of all parameter values in candidate graphs G:

$$h(S, \lambda) = \arg\max_{G} \sum_{(w_i,r,w_j) \in G} \mathbf{w} \cdot f(w_i, r, w_j)$$

Perceptron properties

- If the corpus is linearly separable, the perceptron learning algorithm will find a linear model that separates it.
- Which model? Results are highly variable. The model found can depend on the order in which events are presented.
- What if the date aren't linearly separable?
  Not so good. In that case, the perceptron is not guaranteed to terminate. It will eventually begin to cycle. However, there is no general bound on the number of steps in a cycle, so the cycles can be quite hard to detect.

General "online" training

Perceptron is an instance of online training: Parameters of model are updated with each example.

ONLINE-LEARNER($\mathcal{D}, N$)

1 **w** $= 0, i = 0$
2 **for** $n \leftarrow 1$ **to** N
3 **do for** $d \leftarrow 1$ **to** $|\mathcal{D}|$
4 **do** $w^{i+1} =$ update w^i acording to instance (S_d, G_d)
5 $i = i + 1$
6 **return w**

Averaged perceptron

- In online training, parameters updated $|\mathcal{D}|$ times for each of N rounds;
- Each update can be looked at as a separate hypothesis about the correct values of parameters **w**; Call the hypothesis for round i and training example j $w_{i,j}$. The last such hypotheses is $w_{|\mathcal{D}|,\mathrm{N}}$
- Collins (2002), following Freund and Schapire (1999), finds that averaging the results of these hypothesis helps prevent overfitting. Instead of returning $w_{|\mathcal{D}|,\mathrm{N}}$, we return:

$$\frac{\sum_{i,j}^{|\mathcal{D}|,\mathrm{N}} w_{i,j}}{|\mathcal{D}| \times \mathrm{N}}$$

Online training w/ averaging

PERCEPTRON($\mathcal{D}, N$)

1 $\mathbf{w} = 0, v = \mathbf{0}, i = 0$
2 **for** $n \leftarrow 1$ **to** N
3 **do for** $d \leftarrow 1$ **to** $\mid \mathcal{D} \mid$
4 **do** $w^{i+1} =$ update w^i acording to instance $(\mathrm{S}_d, \mathrm{G}_d)$
5 $v = v + w^{i+1}$
6 $i = i + 1$
7 $w = v/(\mathrm{N}+ \mid \mathcal{D} \mid)$
8 **return w**

Multi-class classification

Crammer and Singer (2002)

MULTICLASS OPTIMIZATION($\mathcal{D}$)

1 min $\| w \|$

2 s.t. $s(x, y) - s(x, y') \geq \mathrm{L}(y, y')$

3 $\forall (x, y) \in \mathcal{D}, y' \in \mathrm{cl}(x)$

$cl(x)$ is the set of possible classes for training instance x, y is the correct class for x, and $\mathrm{L}(y, y')$ is a real-valued loss function specific to the classification problem. We call:

$$s(x, y) - s(x, y')$$

the margin between y and y'.

Minimization of $\| w \|$ is subject to a set of constraints: For each training example, the margin between the right class y and any other class y' must be at least as great as the loss of y' ($\mathrm{L}(y, y')$).

Margin in dependency parsing

McDonald et al. (2005)a use structured SVMs, in which the candidate parsetrees for a sentence S (dt(S) are the "classes":

DEPENDENCY PARSING OPTIMIZATION($\mathcal{D}$)

1 min $\| w \|$
2 s.t. $s(\mathrm{S}, \mathrm{G}) - s(\mathrm{S}, \mathrm{G}') \geq \mathrm{L}(\mathrm{G}, \mathrm{G}')$
3 $\forall(\mathrm{S}, \mathrm{G}) \in \mathcal{D}, \mathrm{G}' \in \mathrm{dt}(\mathrm{S})$

$$\mathrm{L}(\mathrm{G}, \mathrm{G}') = \sum_{(w', r, w) \in \mathrm{G}} [(w'', r, w) \in \mathrm{G}' \text{ and } w'' \neq w']$$

The loss for G' is the number of words with an incorrect parent.

Constraints: the margin between each correct tree G and each incorrect tree G' must be at least as large as the loss.

MIRA

The Margin Infused Relaxed Algorithm (MIRA) Crammer and Singer propose for solving the multiclass optimization problem is an online learning algorithm (Crammer and Singer 2002):

ONLINE MULTICLASS UPDATE($\mathcal{D}$)

1 $\min \parallel w^{i+1} - w^{i} \parallel$
2 s.t. $s(\mathrm{S}_d, \mathrm{G}_d) - s(\mathrm{S}_d, \mathrm{G'}) \geq \mathrm{L}(\mathrm{G}_d, \mathrm{G'})$
3 $\forall \mathrm{G'} \in \mathrm{dt}(\mathrm{S}_d)$

This update is substituted into line 4 of the online update algorithm (with averaging). Scores are calculated with respect to w^{i+1}:

$$s(\mathrm{S}_d, \mathrm{G}_d) = w^{i+1} \cdot f(\mathrm{S}_d, \mathrm{G}_d)$$

Problem: How do we use this practically in dependency parsing, when the number of parse trees grows exponentially with the length of the sentence?

The k-best assumption

McDonald et al. (2005)a: the constraints that matter for margin optimization are those from the incorrect trees with the k-best scores.

ONLINE K-BEST UPDATE($\mathcal{D}$)

1 min $\| w^{i+1} - w^i \|$
2 s.t. $s(S_d, G_d) - s(S_d, G') \geq L(G_d, G')$
3 $\forall G' \in$ best $_k(S_d)$

	k =1	k =2	k =5	k=10	k =20
Accuracy	90.73	90.82	90.88	90.92	90.91
Train time	183m	235m	627m	1372m	2491m

Modifying Eisner (2000) to k-best (Huang and Chiang 2005)

What to take away: I

Dependency parsers can be broadly divided into two classes.

1. Transition-based
2. Graph-based

This choice directly affects two things:

1. the set of parsing algorithms available.
   - transition-based: some kind of shift-reduce type architecture
   - transition-based: deterministic parsing not required
2. the feature model
   - transition-based: features of parser configurations
   - graph-based
     1. arc-based: McDonald et al. (2005)a shows us how rich this can be.
     2. arc-based + : arity

What to take away: II

The choice of learning algorithm is largely orthogonal to the choice between transition-based and graph-based approaches.

Max margin methods (SVMs) generalize to multiclass problems and to classification with structured classes (parsing, HMMs, protein sequencing)

1. The key idea is the introduction of a real-valued loss function which measures how far away an incorrect (structured) class is from the correct answer.
2. By using the loss to bound the score differences between correct and incorrect trees a QP optimization problem with solutions that can be found efficiently can be formulated.
3. Like Perceptrons and SVMs such optimization problems can be solved in a dual form that updates w example by example: online training.

Bibliography

Chu, Y.J., and T.H. Liu.

1965.

On the shortest arborescence of a directed graph.

Science Sinica 14:1396–1400.

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*, 1–8. Association for Computational Linguistics.

Crammer, K., and Y. Singer.

2002.

On the algorithmic implementation of multiclass kernel-based vector machines.

The Journal of Machine Learning Research 2:265–292.

Edmonds, J.

1968.

Optimum Branchings1.

Mathematics of the decision sciences 346.

Eisner, Jason. 2000.

Bilexical grammars and their cubic-time parsing algorithms.

In H. Bunt and A. Nijholt (Eds.), *Advances in Probabilistic and Other Parsing Technologies*, 29–62. Kluwer Academic Publishers, October.

Freund, Y., and R.E. Schapire.

1999.

Large margin classification using the perceptron algorithm.

Machine learning 37(3):277–296.

Huang, L., and D. Chiang. 2005.

Better k-best parsing.

In *Proceedings of the Ninth International Workshop on Parsing Technology*, 53–64. Association for Computational Linguistics.

Kübler, S., R. McDonald, and J. Nivre. 2009.

Dependency parsing.

Morgan & Claypool Publishers.

Synthesis Lectures on Human Language Technologies.

McDonald, R., K. Crammer, and F. Pereira. 2005a.

Online large-margin training of dependency parsers.

In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 91–98. Association for Computational Linguistics.

McDonald, R., F. Pereira, K. Ribarov, and J. Hajič. 2005b.

Non-projective dependency parsing using spanning tree algorithms.

In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 523–530. Association for Computational Linguistics.

Nivre, J., and J. Nilsson. 2005.

Pseudo-projective dependency parsing.

In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 99–106. Association for Computational Linguistics.

Tarjan, R.E.

1977.

Finding optimum branchings.

Networks 7(1):25–35.