Assignment 3: Parsing Reranker
11–711 recitation

Maria Ryskina
Parse Reranking

Training data:
- sentence $x$
- k–best list $\mathcal{L} = \{y_1, y_2, \ldots, y_k\}$
- gold parse tree $y^*$

Test input:
- sentence $x$
- k–best list $\mathcal{L}$

Goal: find the best tree on the list

$$\hat{y}^* = \arg\max_{y \in \mathcal{L}} F_1(y, y^*)$$

In prediction, approximating with a learned scoring function:

$$\hat{y} = \arg\max_{y \in \mathcal{L}} s(y, x)$$
Perceptron

• Scoring function: \( s(y) = \mathbf{w}^\top \mathbf{f}(y) \)

• Training procedure:
  - make a guess: \( \hat{y} = \arg\max_{y \in \mathcal{L}} \mathbf{w}^\top \mathbf{f}(y) \)
  - adjust weights:
    \[
    \mathbf{w} \leftarrow \mathbf{w} - \mathbf{f}(\hat{y}) \\
    \mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(y^*)
    \]

• Tips:
  - use best tree on the list \( \hat{y}^* \) instead of gold \( y^* \)
  - use averaging
Loss-augmented SVM

- Scoring function: \( s(y) = w^\top f(y) \)

- Training:
  - optimization problem:
    \[
    \min_w k\|w\|^2 - \sum_i \left( w^\top f(y_i^*) - \max_{y \in \mathcal{L}_i} (w^\top f(y) + l(y)) \right)
    \]
  - loss-augmented guess: \( \hat{y} = \arg\max_{y \in \mathcal{L}} (w^\top f(y) + l(y)) \)
  - loss is what we approximate: \( l(y) = 1 - F_1(y, y^*) \)

- At test time, predict under scoring function: \( \arg\max_{y \in \mathcal{L}} w^\top f(y) \)
Primal SVM: Interface

LossAugmentedLinearModel.getLossAugmentedUpdateBundle(D datum, IntCounter weights)

- Takes datum (your representation of k best trees) and feature weights
- Returns an UpdateBundle object:
  - features of the gold tree: \( f(y^*) \)
  - features of the loss-augmented guess tree: \( f(\hat{y}) \)
  - loss of guess tree: \( l(\hat{y}) \)

PrimalSubgradientSVMLearner.train(IntCounter initWeights, final LossAugmentedLinearModel<D> model, List<D> data, int iters)

- Takes initial feature weights, your model, data and number of iterations
- Returns learned feature weights in an IntCounter
- Uses a provided AdaGrad implementation
MaxEnt

• Scoring function: \( s(y) = P(y|x; \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(y))}{\sum_{y' \in \mathcal{L}} \exp(\mathbf{w}^\top \mathbf{f}(y'))} \)

• Training:
  – log conditional likelihood:

\[
L(\mathbf{w}) = \sum_i \log P(y_i^*|x_i; \mathbf{w})
\]

  – optimization problem:

\[
\max_{\mathbf{w}} \quad L(\mathbf{w}) - \lambda \| \mathbf{w} \|^2
\]
MaxEnt: Interface

edu.berkeley.nlp.math.DifferentiableFunction():

- `double valueAt(double[] x)` – computes value
- `double[] derivativeAt(double[] x)` – computes gradient

LBFGSMinimizer.minimize(DifferentiableFunction function, double[] initial, double tolerance)

- Takes log conditional likelihood function, initial weights, and tolerance hyperparameter
- Returns learned feature weights in an array
- Uses a provided L–BFGS implementation
- Be careful: use DifferentiableFunction in `edu.berkeley.nlp.math`, not in `edu.berkeley.nlp.assignments.rerank`
Features

- Features can be products of multiple components:
  - position
  - rule
  - parent label
  - span features
  - ...

[Hall et al. ’14. Less Grammar, More Features]
Span features (Hall et al. ’14)

- First / last word of the span
- Span length
- Span shape:

```
PRN
( CEO of Enron )
(XxX)
```

```
VP
said , “ Too bad , ”
x,“Xx,”
```

- Span context:

```
VP → no VBP NNS
```

```
VP
VBP NNS
no read messages in his inbox
```

- Split point:

```
NP → (NP ... impact) PP)
```

```
NP
NP PP
has an impact on the market
```
Head Features (Charniak & Johnson ’05)

• Syntactic (functional) heads
  – can be obtained with SurfaceHeadFinder

• Semantic (lexical) heads
Head Features (Charniak & Johnson ’05)

- Heads: head-to-head dependencies between constituents
- HeadTree: head and its projections (+ siblings)
Sibling n-gram (Charniak & Johnson ’05)

- NGram: n-grams of adjacent children + relative position to head

[Images: Mark Johnson]
Word Features (Charniak & Johnson ’05)

- Word: word + n ancestors
Tree n-gram (Charniak & Johnson ’05)

• NGramTree: subtree spanning over a lexical n-gram

[Images: Mark Johnson]
Neighbours (Charniak & Johnson ’05)

- Neighbours: label + span length + context POS tags

> 5 words

[Images: Mark Johnson]
Heavyness (Charniak & Johnson ’05)

- Heavyness: label + span length + closeness to end of sentence
Right branch (Charniak & Johnson ’05)

- **RightBranch**: number of nodes in rightmost branch
Coordination (Charniak & Johnson ’05)

- **CoPar**: depth of parallelism of adjacent conjuncts
- **CoLenPar**: difference in length in adjacent conjuncts

[Images: Mark Johnson]
Tips and resources

• Store feature vectors instead of trees
• Compare using gold tree vs. best from k–best list
• Do ablation study
• Watch out for correlation between features

• Useful classes:
  – edu.berkeley.nlp.parser.EnglishPennTreebankParseEvaluator
  – edu.berkeley.nlp.ling.AnchoredTree
  – edu.berkeley.nlp.assignments.rerank.SurfaceHeadFinder

• Slides for Charniak & Jonhson ’05 features:  