11-711 Project 3

Hengruo Zhang
Electrical & Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
hengruoz@andrew.cmu.edu

1 Introduction

In this project, we implement a feature extractor and two training procedures to learn weights for extracted features. While 1-best baseline gets 83.66 F1, our AWESOME reranker gets 86.18 F1 and our BASIC reranker gets 83.75. We use Maximum Entropy for the AWESOME reranker and Primal SVM sub-gradient descent for the BASIC reranker.

The structure of this project is as follows:

- Section 2 introduces the implementation of submitted version.
- Section 3 introduces our experiments, which also explain why we adopted the implementation in Section 2.
- Section 4 provides error analysis.

2 Implementation

2.1 Overview

Each training datum is a pair of k-best list $\mathcal{L} = \{y_1, y_2, \ldots, y_k\}$ and the gold parse tree y^*. Each test datum is a k-best list $\mathcal{L} = \{y_1, y_2, \ldots, y_k\}$. Our goal is training a model to find the best tree on the list:

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

In prediction phase, we need to approximate with a learned scoring function $s(y)$:

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

Generally, the scoring function $s(y)$ is in the form of $s(y) = g(\mathbf{w}^T \mathbf{f}(y))$. Therefore, our goal is to find a way to extract feature vector $\mathbf{f}(y)$ from a parse tree y and to learn an appropriate weight $\mathbf{w}$. To learn an appropriate weight $\mathbf{w}$, we need to minimize training loss

$$\min_{\mathbf{w}} \sum_i l_i(\arg\max_y g(\mathbf{w}^T \mathbf{f}(y)))$$

where $l_i(y) = l(y, y_i^*)$.

2.2 Feature Extraction

We select the following features to compose feature vectors $\mathbf{f}(y)$:

- the position in the k-best list
- CFG rules in this parse tree
- the bucketed span of each sub-parse-tree
- span shape
- the first words of the sentence
- the last words of the sentence
- the POS tag of the first words of the sentence
- the POS tag of the last words of the sentence
- number of nodes in rightmost branch

There are totally 2119075 unique features.

2.3 Reranker

Base Reranker

We use a base reranker class to finish some common operations like initialization weights, calculating the best candidate tree index, and getting the best parse tree. We initialize the weights with uniform random numbers between 0.0 and 0.0001.

AWESOME Reranker: MaxEnt

In Maximum Entropy model, the scoring function is $s(y) = P(y|x; \mathbf{w}) = \frac{\exp(\mathbf{w}^T \mathbf{f}(y))}{\sum_{y' \in \mathcal{L}} \exp(\mathbf{w}^T \mathbf{f}(y'))}$. With this equation and log-loss, we can get the optimization problem

$$\max_{\mathbf{w}} \sum_i \log P(y_i^* \mid x_i; \mathbf{w}) - \lambda \|\mathbf{w}\|^2$$

In our implementation, we choose 1e-3 as the tolerance and 1e-6 as λ.

BASIC Reranker: Primal SVM

In Primal SVM model, the scoring function is $s(y) = \mathbf{w}^T \mathbf{f}(y)$. We use $1 - F_1(y, y^*)$ as the augmented loss. With this equation and hinge-loss, we can get the optimization problem

$$\min_{\mathbf{w}} k\|\mathbf{w}\|^2 - T(\mathbf{w})$$

where

$$T(\mathbf{w}) = \sum_i \left(\mathbf{w}^T \mathbf{f}(y_i^*) - \max_{y \in \mathcal{L}_i} (\mathbf{w}^T \mathbf{f}(y) + l_i(y)) \right)$$

In our implementation, we choose step size 0.1 and L2-multiplier 0.001.

3 Performance

3.1 Weight Initialization

We compare different distributions to initialize weights as listed in Table 1. Different initialization distributions don't influence the results obviously.

Dist	ME F1	SVM F1
Uni on $[0, 1e-5]$	85.2	83.65
Uni on $[0, 1e-4]$	85.5	83.75
Uni on $[0, 1e-3]$	85.42	83.77
Uni on $[0, 1e-2]$	85.23	83.72
Uni on $[-1, 1]$	85.13	83.77
Norm $\mu = 0, \sigma = 1e-5$	85.18	83.82
Norm $\mu = 0, \sigma = 1e-4$	85.08	83.92
Norm $\mu = 0, \sigma = 1e-3$	84.88	83.79
Norm $\mu = 0, \sigma = 1e-2$	84.99	83.69

Table 1: Ways of Weight Initialization

3.2 Feature Selection

Because there are too many optional features, it is unrealistic to compare all combinations. We also tried adding n-grams of adjacent children, relative positions to head, words with their n ancestors, context POS tags, and so on. The current combination is the optimal combination for AWESOME reranker and is relatively faster.

3.3 Tolerance of MaxEnt

We compare different tolerance values, used to judge stopping training, for Maximum Entropy. The results are listed in Table 2. When tolerance value is larger than $1e-3$, F1 scores won't increase.

Tolerance	F1	Time (s)
1e-1	83.84	63.3
1e-2	85.49	113.2
1e-3	85.5	127.4
1e-4	85.5	128.3
1e-5	85.5	124.7
1e-6	85.5	126.7
1e-7	85.5	129.5

Table 2: Tolerance

3.4 λ of MaxEnt

We compare different λ values.By the results in Table 3, we see F1 gets its maximum value when λ is 1e-6.

Lambda	F1	Time (s)
1e-1	83.71	85.3
1e-2	84.15	93.2
1e-3	85.5	127.4
1e-4	86.13	130.3
1e-5	86.17	129.1
1e-6	86.18	129.1
1e-7	86.17	129.1
1e-8	86.17	129.1

Table 3: Tolerance

3.5 Step Size and L2 Multiplier

For Primal SVM model, we tried different step sizes and L2 multipliers. The records are in Table 4.

3.6 Maximum Iteration

By the records in Table 5, more iterations can increase both F1 and time. After 30 iterations, the model gets converged.

step size	**l2**	F1	**Time (s)**
1e-1	1e-1	83.75	729.4
1e-1	1e-2	83.75	703.1
1e-1	1e-3	83.79	693.5
5e-2	1e-1	83.69	713.4
5e-2	1e-2	83.69	684.2
5e-2	1e-3	83.68	739.8
1e-2	1e-1	83.54	729.1
1e-2	1e-2	83.52	693.0
1e-2	1e-3	83.58	687.5

Table 4: Step Size and L2 Multiplier

Max Iter	F1	**Time (s)**
15	83.15	361.2
20	83.51	473.7
25	83.67	587.5
30	83.75	703.1
35	83.76	853.4
40	83.76	998.6

Table 5: Maximum Iteration

4 Error Analysis

4.1 Maximum Length of Test Sentences

By Table 6, it's obvious that F1 scores decrease first and increase later with the maximum length of test sentences. That's because a pretty long sentence has more features provided to reranker and a pretty short sentence is very easy to extract all features it has to match.

Length	**F1**	**Length**	**F1**
5	88.99	25	82.77
10	85.62	30	84.7
15	84.4	35	85.6
20	83.3	40	86.18

Table 6: F1 for different max lengths