Assignment1 - HMM tagging

Adrián Pauer

1 Brills tagger

For the Brill's tagger task, we followed the assignment's instructions and created five different data splits for evaluation. We experimented with various parameter configurations for the tagger. As the baseline, we employed both a unigram tagger and a default tagger that assigns the most frequent tag observed in the training data. We also varied the number of transformation rules used by the Brill tagger, testing sets of 50, 100, and 250 rules. The rule template used in all configurations was brill.fntbl37().

The data was split according to the following schemes:

- $S, H, T = \text{data}[-40000:], \text{data}[-60000:-40000], \text{data}[:-60000]$
- $S_{\text{cx}}, H_{\text{cx}}, T_{\text{cx}} = \text{data}[:40000], \text{data}[-20000:], \text{data}[40000:-20000]$
- $S_1, H_1, T_1 = \text{data}[20000:60000], \text{data}[60000:80000], \text{data}[:20000] + \text{data}[80000:]$
- $S_2, H_2, T_2 = \text{data}[10000:50000], \text{data}[50000:70000], \text{data}[:10000] + \text{data}[70000:]$
- $S_3, H_3, T_3 = \text{data}[-60000:-20000], \text{data}[-20000:], \text{data}[:-60000]$

The tagging accuracies on the English test sets corresponding to these splits are reported in Table 1 and mean accuracies and standard deviations in Table 2.

init_tagger	rules_num	split	accuracy
default	50	0	0.556488
		1	0.553392
		2	0.565038
		3	0.556711
		4	0.547499
	100	0	0.604933
		1	0.601976
		2	0.617330
		3	0.604598
		4	0.598084
	250	0	0.669465
		1	0.661402
		2	0.674756
		3	0.663147
		4	0.659290
unigram	50	0	0.849686
		1	0.837986
		2	0.849304
		3	0.839733
		4	0.843821
	100	0	0.855135
		1	0.843981
		2	0.856186
		3	0.847293
		4	0.849496
	250	0	0.863818
		1	0.854380
		2	0.867108
		3	0.855895
		4	0.859936

Table 1: Accuracy for English dataset for different parameters of brills tagger

init_tagger	rules_num	mean	std
default	50	0.555826	0.006351
	100	0.605384	0.007219
	250	0.665612	0.006368
unigram	50	0.844106	0.005358
	100	0.850418	0.005186
	250	0.860227	0.005322

Table 2: Mean accuracy and standard deviation for English dataset

From both tables, it is evident that using the `unigram` tagger as the baseline yields significantly better performance compared to the default tagger. The best tagging accuracies on the test sets range approximately from 84% to 86%. Furthermore, increasing the number of transformation rules generally leads to a slight improvement in accuracy, indicating that a larger rule set allows the tagger to better refine its predictions.

For the Czech dataset, we used only the `unigram` tagger as the baseline, as the `default` tagger consistently caused the system to run out of memory during training. The resulting accuracies are presented in Table 3, with corresponding average accuracies and standard deviations summarized in Table 4.

init_tagger	rules_num	split	accuracy
unigram	50	0	0.584064
		1	0.601467
		2	0.617752
		3	0.629359
		4	0.586176
	100	0	0.591025
		1	0.608986
		2	0.623805
		3	0.632178
		4	0.593321
	250	0	0.605107
		1	0.625379
		2	0.638275
		3	0.645291
		4	0.607002

Table 3: Accuracy on Czech dataset for different parameters of brills taggert

init_tagger	rules_num	mean	std
unigram	50	0.603763	0.019707
	100	0.609863	0.018177
	250	0.624211	0.018060

Table 4: Mean accuracy and standard deviation for Czech dataset

Tables 3 and 4 indicate that the accuracy on the test data for the Czech dataset consistently hovers around 60%. Compared to the English dataset, the results also exhibit higher standard deviations, suggesting greater variability in performance across the different data splits.

2 HMM tagging

We constructed a trigram Hidden Markov Model (HMM) in which the states are represented by pairs of tags. The transition probabilities for tag trigrams, bigrams, and unigrams were estimated from the training data. After applying smoothing using the held-out set H, we obtained the final transition and emission probability distributions used for further processing.

$$P((t_1, t_2)|(t_0, t_1)) = \alpha_3 P_3((t_1, t_2)|(t_0, t_1)) + \alpha_2 P_2((t_0, t_1)) + \alpha_1 P_1((t_2)) + \alpha_0 \frac{1}{|tags|}$$

$$P(w|t_1, t_2) = \alpha_2 P_2((w|t_1, t_2)) + \alpha_1 P_1(w) + \alpha_3 \frac{1}{|V|}$$

We evaluated the performance of the trigram HMM model using the original data split and the Viterbi algorithm for decoding. To prevent numerical underflow during computation, we applied the logarithmic trick in the calculation of the Viterbi scores (alphas). Additionally, for efficiency, we employed pruning by retaining only the top 10 alphas at each step.

For the English dataset, using the original data split, we achieved an accuracy 0.8572992986207076, which is comparable to the performance of the Brill tagger. The results were similar for the Czech dataset, where we obtained an accuracy 0.6149370317232584.

2.1 Supervised HMM and baum welch

For the Baum-Welch algorithm, we used the first 10,000 words of the training data for supervised initialization of the model parameters. The remaining portion of the data was used for unsupervised training via the Baum-Welch procedure. Our implementation followed the standard approach presented in the lecture slides: computing the forward probabilities (alphas), then the backward probabilities (betas), and finally updating the transition and emission probabilities using expected counts.

However, the algorithm did not converge within a reasonable timeframe for either the English or Czech datasets. Processing a single epoch (one pass through the dataset) took approximately 20 hours. This strongly suggests a possible bug or inefficiency in the implementation that significantly impacts performance.