

# Q1:

## (a)

i:

The new prob is 0.491

After changing,  $p(\text{Day 1} = H | \text{Observations for Day 1-3})$  is roughly 0.5

Because in the default model, the emission probabilities favor larger numbers of ice creams on hot days, this means that the only two plausible weather sequences are HHH and CHH, other paths (e.g. HCH, CHH ...) are have much smaller probability.

So,  $p(\text{Day 1} = H | \text{Observations}) \approx \frac{p(HHH, w)}{p(HHH, w) + p(CHH, w)}$

After replacing Day 1's observation with 1 ice cream, the term penalizes the HHH path and boosts the CHH path. As a result, the two paths contribute nearly equal total probability:

$$p(HHH, w) \approx p(CHH, w)$$

So, the new prob is roughly 0.5.

ii

After changing,  $p(\text{Day 1} = H)$  has reduced from roughly 0.9 to about 0.5 as in question (a)i. the uncertainty on Day 1 propagates forward through  $p(H|C)$  and  $p(H|H)$ . The forward probability for Day 2 being Hot becomes a weighted combination of these two possible starting points:

$$p(t_2 = H) \propto p(t_1 = H)p(H|H) + p(t_1 = C)p(H|C)$$

Since  $p(t_1 = H)$  dropped,  $p(\text{Day 1} = H)$  has also slightly dropped rom about 0.977 to roughly 0.918 in the initial reconstruction.

iii:

After 10 iterations of re-estimation:

Day 1:

$p(H)$  rises from nearly 0 to about 0.557.

The model has partially re-interpreted the "1 ice cream" observation as sometimes compatible with hot days.

Day 2:

$p(H)$  remains very high (0.993), since its “3 ice cream” observation still overwhelmingly indicates a hot day.

Thus the EM process smooths the probabilities, slightly recovering the Day 1 hot probability while keeping Day 2 confidently hot.

## **(b)**

### **i**

From the two graphs, the most noticeable difference appears between Days 11–14.

In the original graph (before changing  $p(1|H)$ ), the model still considers these days to be mostly hot, with  $p(H)$  staying around 0.7–0.9 even though ice-cream consumption temporarily drops to 1 or 2.

After setting  $p(1|H)=0$  and restricting  $p(2|H) = 0.3$ , the model can no longer explain low ice-cream counts under hot weather.

As a result, during Days 11–14 the pink  $p(H)$  line collapses to nearly 0, marking these as cold days.

This sharp dip contrasts with the earlier smooth curve and demonstrates how the strong prior (“never eat 1 ice cream on a hot day”) forces the reconstruction to switch abruptly from Hot to Cold in that mid-period.

After Day 14, when the ice-cream counts rise again,  $p(H)$  recovers quickly to values near 1.

### **ii:**

After setting  $p(1|H) = 0$ , any “1-ice-cream” observation can no longer be generated by the Hot state.

Therefore, even after 10 iterations, the probabilities for those days of “1-ice-cream” remain  $p(H) = 0$ .

The model has reached a fixed point — re-estimation cannot change structural zeros.

### **iii**

After 10 iterations,  $p(1|H)$  remains 0. it never increases.

Expectation step (E-step):

During the forward-backward computation, every path in the trellis that would have required a Hot to (1 ice cream) emission now has probability 0.

Therefore, none of the  $2^{33}$  possible state sequences that include such emissions contribute any probability mass.

As a result, the expected (fractional) count for the pair (H, 1) is 0.

Maximization step (M-step):

The M-step re-estimates each emission probability as:

$$p(w|t) = \frac{\text{expectedcount}(t, w)}{\text{expectedcount}(t)}$$

Since the expected count for (H, 1) = 0, the update yields  $p(1 | H) = 0$  again.

This repeats on every iteration: each E-step sees zero paths using (H, 1), each M-step keeps it 0.

## (c)

### i

Intuitively,  $\beta_{bos}(0)$  represents “the probability of generating the entire sentence starting from the beginning-of-sentence tag.”

Just as the forward algorithm ends with  $\alpha_{eos}(n + 1)$ , the backward algorithm begins with  $\beta_{bos}(0)$ .

Both values equal the overall sentence probability  $p(w)$ .

$$p(w_1, .., w_n) = \alpha_{eos}(n + 1) = \beta_{bos}(0)$$

### ii

Meaning of an H constituent:

It represents the part of the sentence that is generated while the model is in the Hot state.

Probability of H to 1 C:

$p_B(1|H) * p_A(C|H)$ : emit "1" while in Hot, then transition to Cold.

Probability of H to  $\epsilon$

$p_A(eos|H)$ : terminate the Hot sequence by going to the end-of-sentence symbol.

Why prefer the more complex grammar (with EH/EC):

It separates emission and transition into different rules, making the structure clearer and the inside-style probability computations easier to organize.

## Q2:

### (a)

Because  $\alpha_{BOS}(0)$  and  $\beta_{EOS}(n + 1)$  are the boundary, and setting them as 1 represents that the valid tag sentence must start from BOS state and end in EOS state. These boundary conditions are supposed to be true not just because they are meant to be in this way theoretically but also as base cases for forward and backward algorithm to propagate with initials and have a neutral multiplicative factor.

### (b)

This happens because the raw file is unlabeled, and the model can freely marginalize over the tags, matching the data distribution it was trained on. While the dev file contains fixed labels that may differ in distribution and constrain the model more strictly. Therefore, the perplexity (surprisal) on the dev file is higher.

The dev perplexity is more important, since it reflects the model's generalization ability to unseen labeled data rather than how well it explains its own training distribution.

### (c)

Because using words from dev to construct  $V$  may introduce data leakage. This information leakage might give model unfair prior knowledge of test data, which makes model cheat.

### (d)

The iterations hurt the overall tagging accuracy. The tagging accuracy overall decreased from 88.63% to 87.035%; the accuracy on known words decreased from 93.06% to 91.39%; the accuracy on seen words increased from 44.11% to 45.79%; and the accuracy on novel words decreased from 42.73% to 40.29%.

### (e)

It sometimes might help because model can learn new words from the context. Through semi-supervised learning, model witnessed some new words in the untagged sentences, and can infer the tag of such new words by the surrounding words that are learned during the supervised learning. And such inference might be aligned with the correct tagging.

The fixed words are as follows, which are correctly tagged by the semi-supervised model but mistakenly tagged by the supervised model.

[FIXED] strongly: ('strongly', 'D') → ('strongly', 'R') (gold=R)  
[FIXED] OOV: ('OOV', 'N') → ('OOV', 'V') (gold=V)  
[FIXED] cafeteria: ('cafeteria', 'I') → ('cafeteria', 'N') (gold=N)  
[FIXED] exclusive: ('exclusive', 'D') → ('exclusive', 'J') (gold=J)  
[FIXED] OOV: ('OOV', 'N') → ('OOV', 'V') (gold=V)  
[FIXED] up: ('up', 'I') → ('up', 'R') (gold=R)  
[FIXED] previously: ('previously', 'D') → ('previously', 'R') (gold=R)  
[FIXED] assumed: ('assumed', 'N') → ('assumed', 'V') (gold=V)  
[FIXED] offices: ('offices', 'V') → ('offices', 'N') (gold=N)  
[FIXED] packaging: ('packaging', 'D') → ('packaging', 'N') (gold=N)  
[FIXED] checks: ('checks', ',') → ('checks', 'N') (gold=N)  
[FIXED] checks: ('checks', 'P') → ('checks', 'N') (gold=N)  
[FIXED] 65: ('65', 'N') → ('65', 'C') (gold=C)  
[FIXED] OOV: ('OOV', 'V') → ('OOV', 'R') (gold=R)  
[FIXED] reckons: ('reckons', 'D') → ('reckons', 'V') (gold=V)  
[FIXED] out: ('out', 'I') → ('out', 'R') (gold=R)  
[FIXED] Latin: ('Latin', 'N') → ('Latin', 'J') (gold=J)  
[FIXED] OOV: ('OOV', 'N') → ('OOV', 'J') (gold=J)  
[FIXED] tells: ('tells', 'N') → ('tells', 'V') (gold=V)  
[FIXED] serious: ('serious', 'V') → ('serious', 'J') (gold=J)  
[FIXED] about: ('about', 'R') → ('about', 'I') (gold=I)  
[FIXED] 3.1: ('3.1', 'D') → ('3.1', 'C') (gold=C)  
[FIXED] back: ('back', 'R') → ('back', 'J') (gold=J)  
[FIXED] seat: ('seat', 'V') → ('seat', 'N') (gold=N)  
[FIXED] whose: ('whose', 'I') → ('whose', 'W') (gold=W)  
[FIXED] match: ('match', 'N') → ('match', 'V') (gold=V)  
[FIXED] vary: ('vary', 'N') → ('vary', 'V') (gold=V)  
[FIXED] advantages: ('advantages', 'V') → ('advantages', 'N') (gold=N)  
[FIXED] buys: ('buys', 'J') → ('buys', 'V') (gold=V)  
[FIXED] buys: ('buys', 'J') → ('buys', 'V') (gold=V)  
[FIXED] comfortable: ('comfortable', 'N') → ('comfortable', 'J') (gold=J)  
[FIXED] procedures: ('procedures', 'V') → ('procedures', 'N') (gold=N)  
[FIXED] Amex: ('Amex', 'C') → ('Amex', 'N') (gold=N)

[FIXED] founded: ('founded', 'N') → ('founded', 'V') (gold=V)  
[FIXED] nothing: ('nothing', 'I') → ('nothing', 'N') (gold=N)  
[FIXED] Unfortunately: ('Unfortunately', 'N') → ('Unfortunately', 'R') (gold=R)  
[FIXED] Later: ('Later', 'N') → ('Later', 'R') (gold=R)  
[FIXED] Vietnam: ('Vietnam', 'V') → ('Vietnam', 'N') (gold=N)  
[FIXED] OOV: ('OOV', 'V') → ('OOV', 'R') (gold=R)  
[FIXED] sixth: ('sixth', 'N') → ('sixth', 'J') (gold=J)  
[FIXED] OOV: ('OOV', 'V') → ('OOV', 'N') (gold=N)  
[FIXED] newly: ('newly', 'N') → ('newly', 'R') (gold=R)  
[FIXED] OOV: ('OOV', 'N') → ('OOV', 'J') (gold=J)  
[FIXED] OOV: ('OOV', 'V') → ('OOV', 'N') (gold=N)  
[FIXED] province: ('province', ',') → ('province', 'N') (gold=N)  
[FIXED] entry: ('entry', 'V') → ('entry', 'N') (gold=N)  
[FIXED] Those: ('Those', 'N') → ('Those', 'D') (gold=D)  
[FIXED] Third: ('Third', 'N') → ('Third', 'J') (gold=J)  
[FIXED] Hunt: ('Hunt', 'J') → ('Hunt', 'N') (gold=N)  
[FIXED] silver: ('silver', 'J') → ('silver', 'N') (gold=N)  
[FIXED] Hunt: ('Hunt', 'W') → ('Hunt', 'N') (gold=N)  
[FIXED] asking: ('asking', 'I') → ('asking', 'V') (gold=V)  
[FIXED] money-market: ('money-market', 'N') → ('money-market', 'J') (gold=J)  
[FIXED] Malaysia: ('Malaysia', 'D') → ('Malaysia', 'N') (gold=N)  
[FIXED] abortion: ('abortion', 'P') → ('abortion', 'N') (gold=N)  
[FIXED] targets: ('targets', 'I') → ('targets', 'V') (gold=V)  
[FIXED] league: ('league', 'J') → ('league', 'N') (gold=N)  
[FIXED] more: ('more', 'J') → ('more', 'R') (gold=R)  
[FIXED] widens: ('widens', ',') → ('widens', 'V') (gold=V)  
[FIXED] Columbus: ('Columbus', 'V') → ('Columbus', 'N') (gold=N)  
[FIXED] Day: ('Day', 'T') → ('Day', 'N') (gold=N)  
[FIXED] oddly: ('oddly', 'V') → ('oddly', 'R') (gold=R)  
[FIXED] 52: ('52', 'N') → ('52', 'C') (gold=C)  
[FIXED] match: ('match', 'N') → ('match', 'V') (gold=V)  
[FIXED] unlikely: ('unlikely', 'N') → ('unlikely', 'J') (gold=J)  
[FIXED] behind: ('behind', 'N') → ('behind', 'R') (gold=R)  
[FIXED] 4.4: ('4.4', 'D') → ('4.4', 'C') (gold=C)  
[FIXED] fostered: ('fostered', 'I') → ('fostered', 'V') (gold=V)

[FIXED] Fed: ('Fed', 'D') → ('Fed', 'N') (gold=N)  
[FIXED] centers: ('centers', ',') → ('centers', 'N') (gold=N)  
[FIXED] D'Arcy: ("D'Arcy", 'D') → ("D'Arcy", 'N') (gold=N)  
[FIXED] primarily: ('primarily', 'N') → ('primarily', 'R') (gold=R)  
[FIXED] sit: ('sit', 'N') → ('sit', 'V') (gold=V)  
[FIXED] Park: ('Park', 'V') → ('Park', 'N') (gold=N)  
[FIXED] fledgling: ('fledgling', 'D') → ('fledgling', 'N') (gold=N)  
[FIXED] water: ('water', 'D') → ('water', 'N') (gold=N)  
[FIXED] hit: ('hit', 'D') → ('hit', 'V') (gold=V)  
[FIXED] harder: ('harder', 'N') → ('harder', 'R') (gold=R)  
[FIXED] poor: ('poor', 'N') → ('poor', 'J') (gold=J)  
[FIXED] bid: ('bid', 'V') → ('bid', 'N') (gold=N)  
[FIXED] mark: ('mark', 'I') → ('mark', 'V') (gold=V)  
[FIXED] steel: ('steel', 'D') → ('steel', 'N') (gold=N)  
[FIXED] shipped: ('shipped', 'N') → ('shipped', 'V') (gold=V)  
[FIXED] steel: ('steel', 'D') → ('steel', 'N') (gold=N)  
[FIXED] structural: ('structural', 'C') → ('structural', 'J') (gold=J)  
[FIXED] steel: ('steel', 'D') → ('steel', 'N') (gold=N)  
[FIXED] steel: ('steel', 'D') → ('steel', 'N') (gold=N)  
[FIXED] damp: ('damp', '\$') → ('damp', 'V') (gold=V)  
[FIXED] specialists: ('specialists', '') → ('specialists', 'N') (gold=N)  
[FIXED] 13th: ('13th', 'N') → ('13th', 'J') (gold=J)  
[FIXED] that: ('that', 'I') → ('that', 'D') (gold=D)  
[FIXED] fear: ('fear', 'V') → ('fear', 'N') (gold=N)  
[FIXED] increases: ('increases', 'J') → ('increases', 'V') (gold=V)  
[FIXED] field: ('field', 'D') → ('field', 'N') (gold=N)  
[FIXED] smart: ('smart', 'V') → ('smart', 'J') (gold=J)  
[FIXED] leave: ('leave', 'N') → ('leave', 'V') (gold=V)  
[FIXED] OOV: ('OOV', 'N') → ('OOV', 'J') (gold=J)  
[FIXED] OOV: ('OOV', 'V') → ('OOV', 'N') (gold=N)  
[FIXED] 's: ('"s"', 'P') → ('"s"', 'V') (gold=V)  
[FIXED] Beach: ('Beach', ',') → ('Beach', 'N') (gold=N)  
[FIXED] Fed: ('Fed', 'D') → ('Fed', 'N') (gold=N)  
[FIXED] saw: ('saw', 'V') → ('saw', 'N') (gold=N)  
[FIXED] Unfortunately: ('Unfortunately', 'N') → ('Unfortunately', 'R') (gold=R)

[FIXED] stock-market: ('stock-market', 'C')  $\rightarrow$  ('stock-market', 'N') (gold=N)  
 [FIXED] drugs: ('drugs', 'C')  $\rightarrow$  ('drugs', 'N') (gold=N)  
 [FIXED] junk-bond: ('junk-bond', 'N')  $\rightarrow$  ('junk-bond', 'J') (gold=J)  
 [FIXED] high-yield: ('high-yield', 'N')  $\rightarrow$  ('high-yield', 'J') (gold=J)  
 [FIXED] junk-bond: ('junk-bond', 'N')  $\rightarrow$  ('junk-bond', 'J') (gold=J)  
 [FIXED] junk-bond: ('junk-bond', 'N')  $\rightarrow$  ('junk-bond', 'J') (gold=J)  
 [FIXED] portfolios: ('portfolios', ',')  $\rightarrow$  ('portfolios', 'N') (gold=N)  
 [FIXED] Partnership: ('Partnership', 'I')  $\rightarrow$  ('Partnership', 'N') (gold=N)  
 [FIXED] 98: ('98', 'N')  $\rightarrow$  ('98', 'C') (gold=C)  
 [FIXED] out: ('out', 'I')  $\rightarrow$  ('out', 'R') (gold=R)  
 [FIXED] structured: ('structured', ',')  $\rightarrow$  ('structured', 'V') (gold=V)  
 [FIXED] 111: ('111', 'N')  $\rightarrow$  ('111', 'C') (gold=C)

To explain this more deeper, I implemented functions in the backward function to save the fractional emission and transition counts for each token. Specifically, For each position  $j$  in every raw sentence, the emission responsibilities  $p(t_j = t | \text{sentence})$  were stored as tuples (word, tag, p). For every adjacent pair of positions, the fractional transition responsibilities  $p(t_j = s, t_{j+1} = t | \text{sentence})$  were recorded as (prev\_word, next\_word, prev\_tag, next\_tag, p). These logs capture how corpus contributed to the parameter updates of A and B. After training, we can trace the reason that fix the tagging. For example, the tag of 'tells' is fixed from 'N' to 'V' with the context 'violinist tells', 'investor tells', and 'expert tells' that contributed 3, 0.447, and 0.035 fractional counts in respective.

Contextual contributors for word 'tells' (V):

(,  $\rightarrow$  V) in ', tells' contrib=3.614  
 (N  $\rightarrow$  V) in 'violinist tells' contrib=3.000  
 (:  $\rightarrow$  V) in '-- tells' contrib=3.000  
 (N  $\rightarrow$  V) in 'investor tells' contrib=0.447  
 (N  $\rightarrow$  V) in 'expert tells' contrib=0.177  
 (T  $\rightarrow$  V) in 'expert tells' contrib=0.035  
 (,  $\rightarrow$  V) in 'expert tells' contrib=0.031  
 (P  $\rightarrow$  V) in 'expert tells' contrib=0.027  
 (V  $\rightarrow$  V) in 'expert tells' contrib=0.022  
 (R  $\rightarrow$  V) in 'expert tells' contrib=0.020

Another example is that the tag of 'founded' fixed from 'N' to 'V' thanks to 'son founded' with

contribution of 3 fractional counts.

## (f)

Semi-supervised learning does not always help because (1) as iteration goes on, the tagging scheme of model tends to favor the incomplete-data log-likelihood object, which only cares which tag being used can best explain the data regardless of the correctness. In such circumstance, the model might tag the words with wrong tags but higher explanation, and those tags are reinforced by iterations since they are kept and might cause further wrong inference on the other word's tagging, degrading emission and transition estimates. As a result, the log-likelihood on the raw data increases but the tagging accuracy decreases. (2) There might exist mismatch between the supervised data and raw data, training on raw data may shift the parameter distribution away from the one trained on the supervised data. (3) The ratio of supervised data amount to raw data amount also impacts the model's accuracy on tagging. If the supervised data is very little compared to the raw data amount, then the parameter might be drastically shifted and the learned knowledge cannot resist the change from the raw data and be covered ruthlessly.

## (g)

The performances on endev of bigram and unigram HMM with different combination of training sets are as shown in the table.

Model	Tagging acc. (all)	CE
Bigram HMM - ensup	88.663%	7.5995
Unigram HMM - ensup	85.544%	8.7939
Bigram HMM - ensup+enraw	87.110%%	7.4819
Unigram HMM - ensup+enraw	85.340%	8.7730

When only doing supervised learning on ensup, bigram HMM has higher tagging accuracy and lower cross entropy than unigram HMM. The result that Bigram HMM performs better than Unigram in terms of accuracy and cross entropy does not change after using enraw. However, both performances of two models become worse. However unigram HMM has smaller cross entropy compared to only using ensup.

## (h)

One stage:

1. only enraw: all: 9.412%, known: 10.000%, seen: 4.377%, novel: 2.906% / Cross-entropy: 10.8643 nats (= perplexity 52278.880)
2. ensup+enraw: all: 87.110%, known: 91.452%, seen: 43.771%, novel: 41.480% / Cross-entropy: 7.4819 nats (= perplexity 1775.550)
3. ensup+ensup+ensup+enraw: all: 90.208%, known: 94.913%, seen: 42.929%, novel: 40.885% / 7.1453 nats (= perplexity 1268.093)

Two stages:

4. stage1: enraw stage2: ensup: all: 88.663%, known: 93.059%, seen: 44.108%, novel: 42.734% / Cross-entropy: 7.4505 nats (= perplexity 1720.761)
5. stage1: ensup stage2: ensup: all: 87.035%, known: 91.397%, seen: 45.791%, novel: 40.291% / Cross-entropy: 7.3486 nats (= perplexity 1553.994) (Original one)

Based on the result, ensup+enraw, ensup+ensup+ensup+enraw, and enraw->ensup seem to work.

1. ensup+enraw: during the training, the accuracy was improved at first and then dropped, which indicating that the incomplete-data log-likelihood dominated while iteration increases. However, it is still slightly better than our original approach. This might be because that the model trained simultaneously with both data is less likely to be pulled by the log-likelihood of raw data.
2. ensup+ensup+ensup+enraw: the accuracy pattern is similar to ensup+enraw, but the accuracy is much higher and the cross-entropy is much lower. This indicates that by weighting supervised data more heavily, we can suppress the drift from correct tag to incomplete-data log-likelihood and force the model parameter more explainable of the gold label rather than the raw data distribution.
3. enraw->ensup: the accuracy is also higher than the original approach, and this is because that doing supervised learning after unsupervised learning makes model learn the distribution of the raw data and then provide supervision signals that correctify the tagging, which is more stable since we are leading the model towards right direction after it learns the basic knowledge of data rather than reinforcing the incomplete-data log-likelihood as we did in the original approach.

## Q3

In this part, I implemented posterior decoding as an alternative to the standard Viterbi decoding for the HMM tagger.

Unlike Viterbi decoding, which searches for the single most likely tag sequence  $\operatorname{argmax}_y P(y|x)$ , posterior decoding independently chooses for each word the tag with the highest marginal posterior probability:

$$y_t = \operatorname{argmax}_y P(y_t | x)$$

This allows each position to be tagged based on the overall marginal distribution rather than being constrained by one globally optimal sequence.

And I also implement Hard lexical constraints. It prevent implausible tags for closed-class and highly regular tokens (e.g., the as D, punctuation, auxiliaries). This substantially reduces systematic errors, yielding a further +3.46% over Viterbi and +3.46–3.46 vs no-hard posterior. The experiments results as below:

System	Tagging acc. (all)
HMM (Viterbi)	<b>87.035%</b>
HMM (Posterior, no hard)	<b>88.225%</b>
HMM (Posterior + Hard constraints)	<b>91.682%</b>

## Q4

(a)

When trained on the same supervised corpus (ensup) with different parameters, the CRF achieved much lower cross-entropy (0.42 nats) and higher tagging accuracy (the best is 89%) compared to the HMM (cross-entropy: 7.45 nats, accuracy: 87%).

This demonstrates that discriminative training of  $p(t|w)$  allows the CRF to focus directly on predicting tags rather than modeling the word sequence, giving better generalization on the dev set.

(b)

Adding unlabeled data (enraw) slightly improved the HMM's log-likelihood (cross-entropy drop from 7.45 to 7.35) and accuracy ( $\approx +0.5\%$ ).

This shows that the HMM can exploit unlabeled sentences through EM by marginalizing over hidden tag sequences.

The CRF, however, cannot use enraw at all, since its conditional objective  $\log p(t \mid w)$  requires gold tags  $t$ .

Thus, its results remain unchanged.

The iterations of semi-supervised training can initially help but often end up hurting overall tagging accuracy after a few rounds since the model will more rely on log-likelihood of the raw data, which helps model to explain words regardless of the actual tags. For known words, the accuracy stays the same since their emission probabilities are well learnt from the supervision data. For seen words, the accuracy might be slightly improved at first due to the extra contexts, then drop if the log-likelihood is reinforced. For the novel words, they might be slightly improved at first since they could be referenced by the nearby known words, but they might be even worse as the iteration goes on than seen words since there is little evidence for them, and log-likelihood might easily make the tag to explain them in other direction far away from the correct tags.

## Q5

I once ate 5 ice creams in one day during a summer holiday.

It was very hot, and they kept handing out free samples.

I didn't get sick, but I didn't want to see ice cream again for a week!