CSCI 544 – Applied Natural Language Processing, Spring 2020

Written Homework 2: Hidden Markov Models

Out: February 4, 2019

Total: 13 pages.

General instructions

1. This is not a graded assignment. Do not turn it in.
2. The assignment is meant as preparation for the in-class exams. You should aim to answer all the questions on your own, without help.
3. Space is provided as it would be on the exams. Answers should be concise and fit in the space provided; in the exams it will not be possible to add space, and long and rambling answers will be penalized.
4. After solving the problems (or giving them your best try), you are encouraged to discuss and compare solutions with your classmates.
5. You are welcome to discuss the problems with us. We encourage open discussion on Piazza, so that the entire class can benefit from the discussion.
6. ~~Answers to select problems will be distributed at a later time.~~
   Answers are indicated in blue.

For further background on hidden Markov models, you may consult the following **optional** readings from Jurafsky and Martin, *Speech and Language Processing* (3rd edition draft):

Chapter 8: Part-of-Speech Tagging

Chapter A: Hidden Markov Models

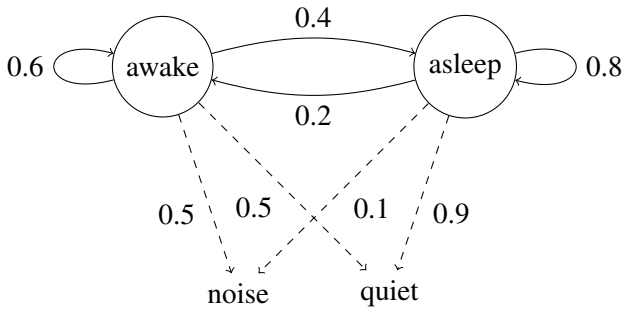
For the purpose of this assignment, please use the definitions given in the assignment, not the ones in the above chapters or any other source.

Markov chains. A Markov chain is a state machine which consists of the following:

1. A set of states $Q = \{q_1, \ldots, q_n\}$.
2. A transition probability matrix A, where each a_{ij} represents the probability of transitioning from state q_i to state q_j, such that for each i, $\sum_{j=1}^{n} a_{ij} = 1$.
$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$
3. A set of possible observations $V = \{v_1, \ldots, v_m\}$.
4. An emission probability matrix B, where each b_{ij} represents the probability of state q_i emitting the observation v_j, such that for each i, $\sum_{j=1}^{m} b_{ij} = 1$.
$$B = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nm} \end{bmatrix}$$
5. A special start state q_0 which is not associated with observations, along with transition probabilities $a_{01} \cdots a_{0n}$ from the start state to the other states. The start state may be identical to one of the other states.

A Markov chain starts at the start state q_0, and at each time point $t_1, t_2 \ldots$ performs a transition and emits an observation. The *Markov property* states that the probability of being in a particular state at time t_i depends only on the previous state (that is, the state at t_{i-1}), and the probability of an observation at time t_i depends only on the current state (that is, the state at t_i).

Problem 1. Consider a Markov chain that represents the probability that a child left alone in her room will be awake or asleep. There are two states $\{\text{awake}, \text{asleep}\}$, and two possible observations coming from the room $\{\text{noise}, \text{quiet}\}$. The transition and emission probabilities are noted in the following diagram: transitions are shown with solid arrows, and emissions with dashed arrows. (Note that the diagram is identical to the one discussed in class, but the probabilities are different!)



The child starts by being awake, and remains in the room for 4 time points, $t_1 \ldots t_4$ (4 iterations of the Markov chain).

a. What is the most probable sequence of states for $t_1 \ldots t_4$?

asleep, asleep, asleep, asleep

b. What is the probability of the above sequence of states?

$$0.4 \times 0.8^3 = 0.2048$$

c. What is the most probable sequence of states and observations?

(asleep,quiet), (asleep,quiet), (asleep,quiet), (asleep,quiet)

d. What is the probability of the above sequence of states and observations?

$$(0.4 \times 0.9) \times (0.8 \times 0.9)^3 = 0.13436928$$

e. What is the least probable sequence of states?

asleep, awake, asleep, awake

f. What is the probability of the above sequence of states?

$$(0.4 \times 0.2)^2 = 0.0064$$

g. What is the least probable sequence of states and observations?

(asleep,noise), (awake, $\left\{ \begin{matrix} noise \\ quiet \end{matrix} \right\}$ *), (asleep,noise), (asleep,noise)*

h. What is the probability of the above sequence of states and observations?

$$(0.4 \times 0.1) \times (0.2 \times 0.5) \times (0.4 \times 0.1) \times (0.8 \times 0.1) = 0.0000128$$

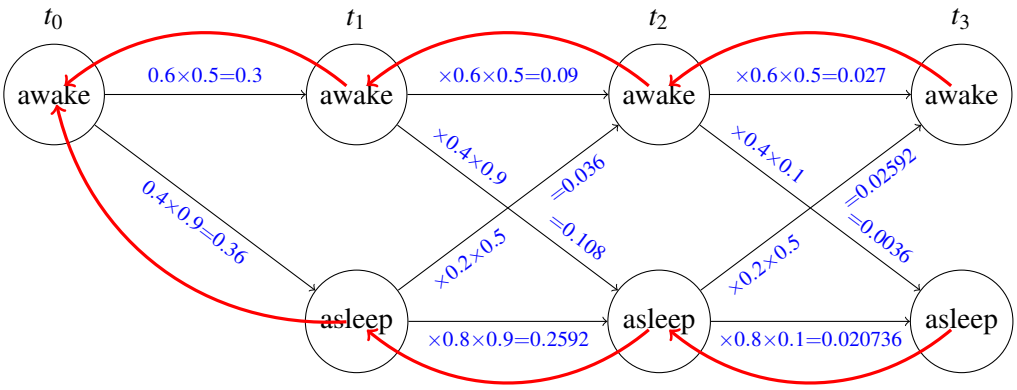
The Viterbi algorithm. A Hidden Markov Model (HMM) is a Markov chain where we cannot observe the states directly, but we can observe the emissions. The Viterbi algorithm is used for decoding the sequence of states, that is finding the most likely sequence of states that could give rise to a sequence of observations. Given a set of states Q and a sequence of time points $1 \dots T$, the algorithm builds two matrices of size $Q \times (1 \dots T)$: a probability matrix representing the probability of each state at each time point, and a backpointer matrix which points from each state at each time point to the most likely previous state. At the final time point T, the algorithm selects the state with the highest probability, and returns the path of backpointers from that state, representing the most likely sequence of states to give rise to the observations. The following is pseudocode for the algorithm: the notation $a(q', q)$ represents the transition probability between states q' and q, and $b(q, o_t)$ represents the emission probability by state q of the observation noted at time t.

 # Initialization step at $t = 1$
for q **in** Q :
    $\text{probability}(q, 1) = a(q_0, q) * b(q, o_1)$
    $\text{backpointer}(q, 1) = q_0$
 # Recursion step for the remaining time points
for t **from** 2 **to** T :
    for q **in** Q :
        $\text{probability}(q, t) = \max_{q' \in Q} \text{probability}(q', t-1) * a(q', q) * b(q, o_t)$
        $\text{backpointer}(q, t) = \arg\max_{q' \in Q} \text{probability}(q', t-1) * a(q', q)$
 # Termination step
$\text{most\_probable\_state}(T) = \arg\max_{q' \in Q} \text{probability}(q', T)$
return the backtrace path by following the backpointers from the most probable state

Problem 2. Consider the same Markov chain from problem 1, this time as a hidden Markov model. The child starts by being awake, and remains in the room for 3 time points, $t_1 \dots t_3$ (3 iterations of the Markov chain). The observations are: quiet, quiet, noise.

a. Using the Viterbi algorithm, identify the most likely sequence of states that would give rise to these observations. Show your work.

Most likely sequence: awake, awake, awake



b. After the first iteration (at t_1), the sum of the probabilities is less than 1. Why? Where is the rest of the probability mass?

The algorithm only considers the probabilities that are consistent with the observation (in this case, o_1 = quiet). The rest of the probability mass is reserved for the other possible observations (that is, o_1 = noise).

c. Suppose we are not interested in decoding the sequence of states (that is, whether the child was awake or asleep at each point), but only in the overall most likely state at the end (that is, whether the child is awake or asleep at the end). Obviously we can remove the backpointer lines from the Viterbi algorithm; however, this would still give us the probability of only the most likely path to each end state. What additional change can we make to the algorithm so that instead of giving us the probability of the most likely path to each state at each time, it will give the overall probability of being at each state at each time? Explain why.

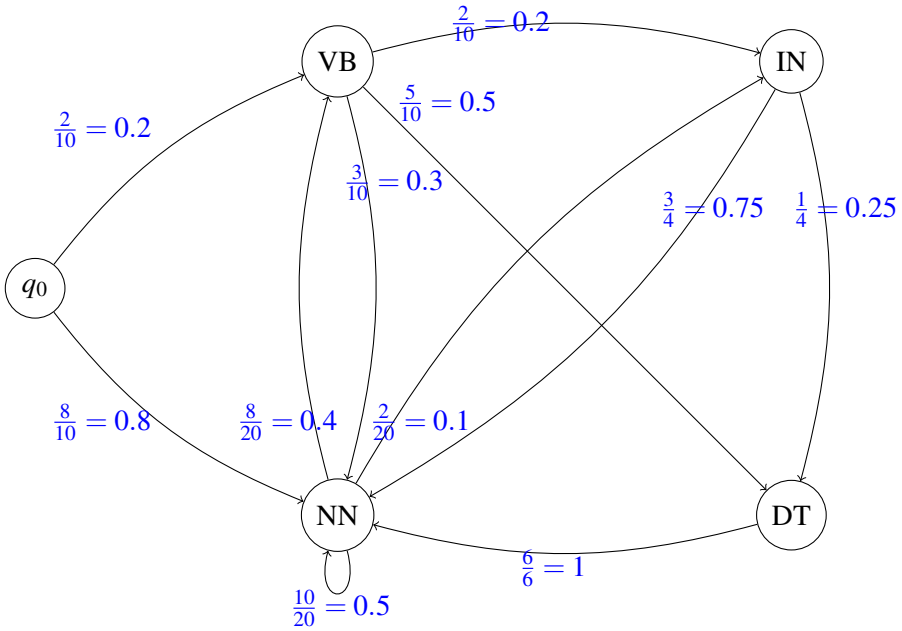
*Single change: replace the **max** function with **sum**.*

Since the paths are mutually exclusive, the sum of the probabilities of individual paths represents the probability of disjunction, that is taking at least one of the paths. By summing up the probabilities of all the paths that converge at each node, we get the probability of arriving at that node regardless of path.

Problem 3. In part-of-speech tagging, we learn a hidden Markov model from a tagged corpus: each state is a part-of-speech tag, transition probabilities are the conditional probabilities of tags given the previous tag, observations are words, and emission probabilities are the conditional probabilities of words given tags. The start state is the beginning of a sentence, which is not a part-of-speech tag. In this problem we will look at some data that will help us tag the sentence *Time flies like an arrow.* We will use the following sentences as a corpus of training data (the notation word/TAG means word tagged with a specific part-of-speech tag).

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

a. Based on the corpus, fill in the transition probabilities in the state chart below.



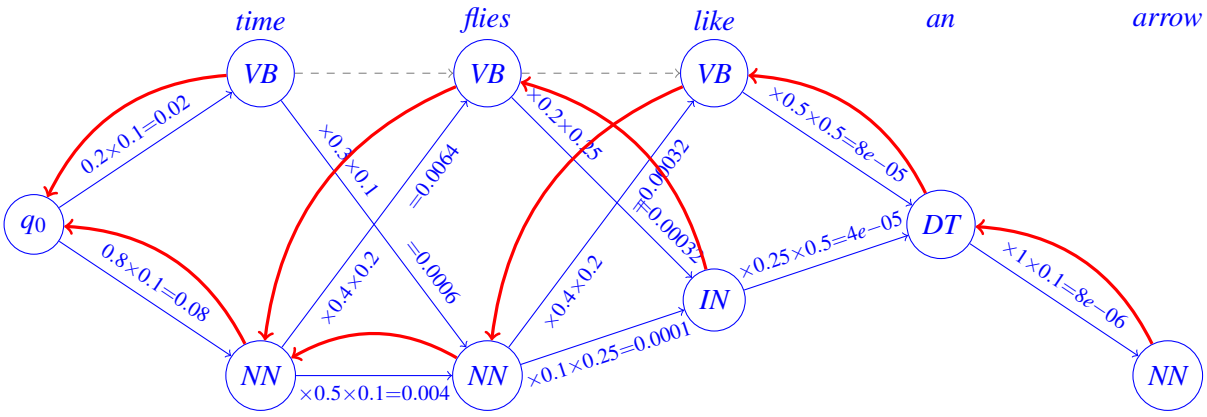
b. Fill in the emission probabilities for the 4 states; only write down the probabilities for the words *time flies like an arrow*.

	time	flies	like	an	arrow
VB	$\frac{1}{10} = 0.1$	$\frac{2}{10} = 0.2$	$\frac{2}{10} = 0.2$	*0*	*0*
NN	$\frac{3}{30} = 0.1$	$\frac{3}{30} = 0.1$	*0*	*0*	$\frac{3}{30} = 0.1$
IN	*0*	*0*	$\frac{1}{4} = 0.25$	*0*	*0*
DT	*0*	*0*	*0*	$\frac{3}{6} = 0.5$	*0*

c. Now use the Viterbi algorithm with the above model to tag the sentence *Time flies like an arrow*. What is the most likely tag sequence, based on the training data? Show your work.

Most likely sequence: NN, NN, VB, DT, NN

Gray dashed lines denote zero probability transitions



Problem 4. You are building a system to tag utterances in English and Spanish. You collect the following sample of 20 short utterances, tagged with part-of-speech labels.

English			Spanish
hay/NN fever/NN	bread/NN pudding/NN	field/NN ***llamas***/NN	***llamas***/VB médico/NN
barn/NN ***hay***/NN	goat/NN meat/NN	desert/NN ***hay***/NN	***hay***/VB queso/NN
hay/NN burns/VB	fish/NN swim/VB	dogs/NN bark/VB	vemos/VB ***llamas***/NN
llamas/NN spit/VB	people/NN run/VB	birds/NN fly/VB	comemos/VB bebemos/VB
ride/VB donkeys/NN	make/VB ***hay***/NN	eat/VB drink/VB	alimentos/NN bebidas/NN

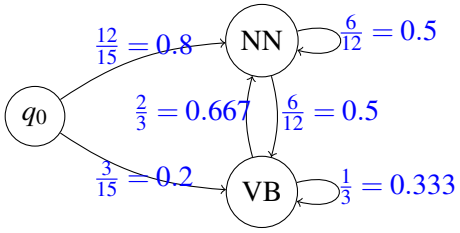
Of course, the above is a very biased sample of general English and Spanish (and some of the utterances are barely grammatical), but let's assume that they are a good sample of the corpus we have at hand. We will tag and identify the language of the unseen utterance *hay llamas*.

a. Based on the training data, what is the prior probability that an utterance selected at random will be English? Spanish?

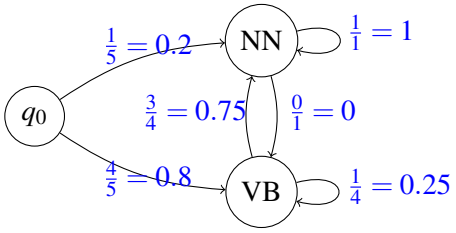
English: $\frac{15}{20} = 0.75$ Spanish: $\frac{5}{20} = 0.25$

b. Assume the tags are generated by a separate hidden Markov model for each language. Based on the training data, estimate the *transition probabilities* for each language.

English



Spanish

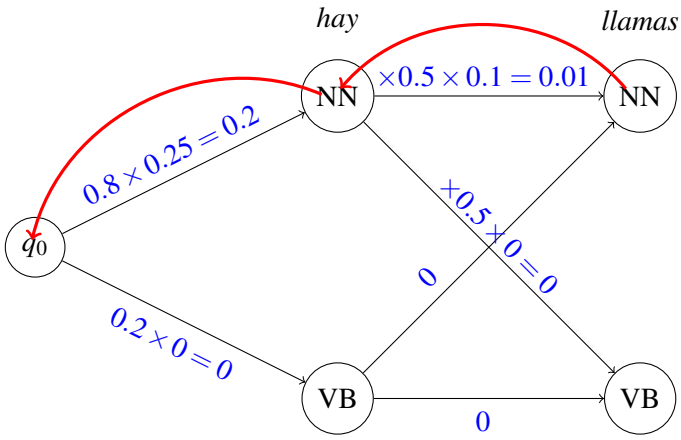


c. Now estimate the *emission probabilities*, again separately for each language. Only write down the probabilities for the words *hay* and *llamas*.

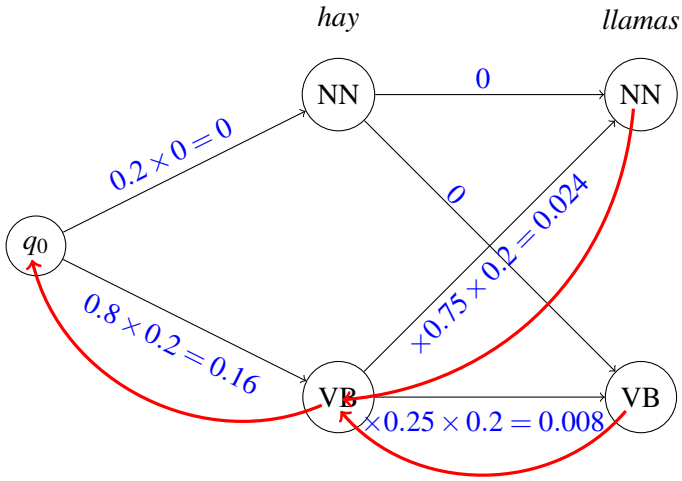
	English		Spanish	
	hay	llamas	hay	llamas
VB	*0*	*0*	$\frac{1}{5} = 0.2$	$\frac{1}{5} = 0.2$
NN	$\frac{5}{20} = 0.25$	$\frac{2}{20} = 0.1$	*0*	$\frac{1}{5} = 0.2$

d. For each language, use the Viterbi algorithm to decode the most likely tag sequence.

English



Spanish



e. What is the most likely language and tag sequence for the utterance *hay llamas*? Why? (Don't forget the prior probabilities!)

The most likely language and tag sequence is English hay/NN llamas/NN: $0.75 \times 0.01 = 0.0075$
Compare to the most likely Spanish tag sequence hay/VB llamas/NN: $0.25 \times 0.024 = 0.006$

f. What is the most likely language (under any tag sequence) for the utterance *hay llamas*? Why? (Don't forget the prior probabilities!)

The most likely language is Spanish: $0.25 \times 0.032 = 0.008$
Compare to the probability of English: $0.75 \times 0.01 = 0.0075$

Problem 5. Suppose we want to use a hidden Markov model to tag a text with the most likely tag for each word, regardless of context. (Yes, this is overkill, as there are much simpler ways to do this, but for this exercise assume we want to use the mechanism of a hidden Markov model.) How would we set the transition probabilities? How would we set the emission probabilities? Why?

The transition probabilities are calculated irrespective of the originating state: that is, the probability of transition from state q_i to q_j is the overall probability of q_j, independent of q_i. In symbols: for each $i, j \leq n$, $a_{ij} = P(q_j)$. This satisfies the constraint on the transition matrix: for each i, $\sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} P(q_j) = 1$.

The emission probabilities are the same as in a standard HMM: for each $i \leq n, j \leq m$, b_{ij} represents the probability of state q_i emitting the observation v_j, that is $P(v_j|q_i)$.

Under this setup, the joint probability at time t of transitioning to q and observing o_t will be $P(q) \cdot P(o_t|q) \propto P(q|o_t)$ (Bayes' theorem). The most probable tag at time t will be $\arg\max_q P(q|o_t)$, that is the most likely tag for the word o_t. The Viterbi algorithm will always choose backpointers to the previous most probable state, since the multipliers from each incoming state are identical.

Two common errors:

1. *A uniform transition probability; emission arcs b_{ij} labeled directly with $P(q_i|v_j)$. **This is incorrect:** while the decoding algorithm will work with such numbers, this is not a HMM because the emission arcs do not constitute a probability distribution, that is they fail to satisfy $\sum_{j=1}^{m} b_{ij} = 1$.*
2. *A uniform transition probability; emission arcs b_{ij} as in a standard HMM: $P(v_j|q_i)$. **This is incorrect:** while the transitions and emissions constitute probability distributions (and therefore the model is a HMM), it does not yield the most likely tag for each word, because it ignores the prior probabilities of the tags.*

Detailed explanation of problem 5.
Two common (and wrong) answers to problem 5 are:

1. Tranistion probabilities: uniform; emission probabilities: $P(\text{tag}|\text{word})$.
2. Tranistion probabilities: uniform; emission probabilities: $P(\text{word}|\text{tag})$.

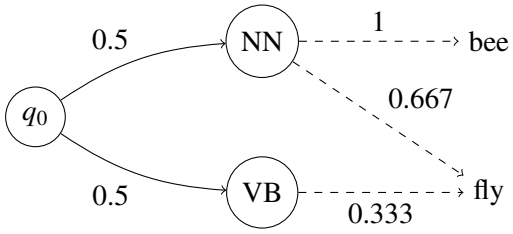
To illustrate the problem with these answers, we will consider a very small dataset with just 10 sentences, each sentence consisting of just one word.

1. fly/NN	2. fly/NN	3. fly/NN	4. fly/NN
5. fly/VB	6. fly/VB		
7. bee/NN	8. bee/NN	9. bee/NN	10. bee/NN

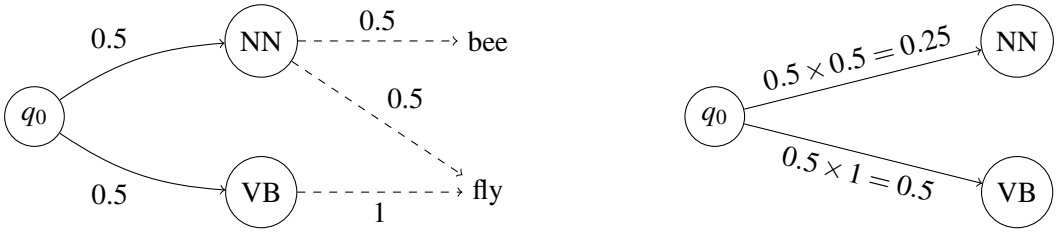
We calculate some probabilities from this dataset.

$P(\text{NN}) = 0.8$	$P(\text{NN}\|\text{fly}) = 0.667$	$P(\text{NN}\|\text{bee}) = 1$	$P(\text{fly}\|\text{NN}) = 0.5$	$P(\text{bee}\|\text{NN}) = 0.5$
$P(\text{VB}) = 0.2$	$P(\text{VB}\|\text{fly}) = 0.333$	$P(\text{VB}\|\text{bee}) = 0$	$P(\text{fly}\|\text{VB}) = 1$	$P(\text{bee}\|\text{VB}) = 0$

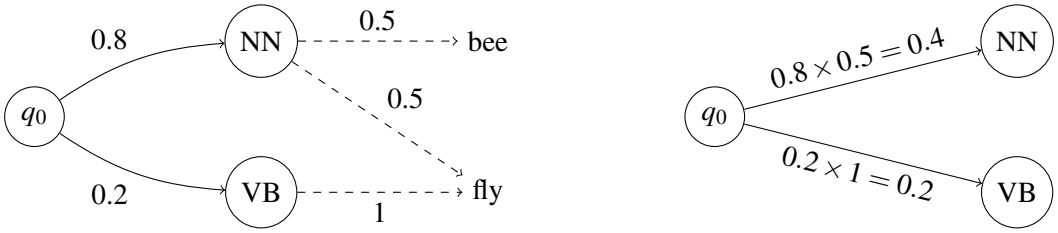
The problem asks for a model that would tag each word with the most likely tag, regardless of context, so it should tag *fly* as NN, and also tag *bee* as NN. The answers above don't quite work. Answer 1 would construct the following state machine.



It is easy to see that this state machine is not a Markov chain at all: the emissions from NN sum up to 1.667, while the emissions from VB sum up to 0.333. Neither of these is a probability distribution. Answer 2, on the other hand, would construct the following state machine.



The state machine above is a Markov chain; however, as the Viterbi decoding on the right shows, it does not produce the desired results – it tags the word *fly* as VB, whereas the most likely tag is NN. What does work, however, is a standard HMM.



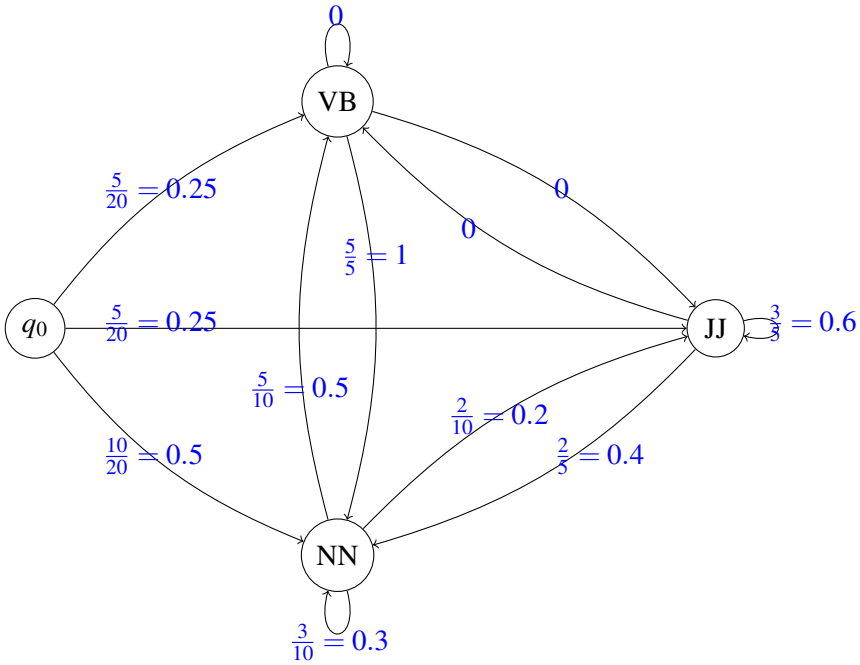
Here the emissions and transitions are both probability distributions, and the most likely tag is guaranteed to be chosen because of Bayes' theorem: $P(\text{tag}|\text{word}) \propto P(\text{word}|\text{tag}) \times P(\text{tag})$. Because the sentences are one word long, the probability of a transition from q_0 to state q_j is equal to the overall probability of tag q_j. With longer sentences, we just need to set the probabilities of transitions from the any state q_i to q_j to be equal to the overall probability of tag q_j. Bayes' theorem will then guarantee that the most likely tag for each word will be chosen, regardless of context.

Problem 6. You are building a system to tag utterances in English. You collect the following sample of 20 short utterances, tagged with part-of-speech labels.

mister/NN jones/NN	**mister**/NN smith/NN	color/NN **gray**/NN	**squirrels**/NN eat/VB
money/NN talks/VB	**gray**/NN wins/VB	**squirrels**/NN drink/VB	**mister**/NN healthy/JJ
mister/NN runs/VB	princess/NN royal/JJ	**squirrels**/VB food/NN	earn/VB **money**/NN
drink/VB tea/NN	cook/VB potatoes/NN	fix/VB things/NN	**gray**/JJ **squirrels**/NN
blue/JJ **gray**/JJ	happy/JJ **squirrels**/NN	yellow/JJ red/JJ	mellow/JJ green/JJ

Using the above data, we will tag the unseen sentence *mister gray squirrels money*.

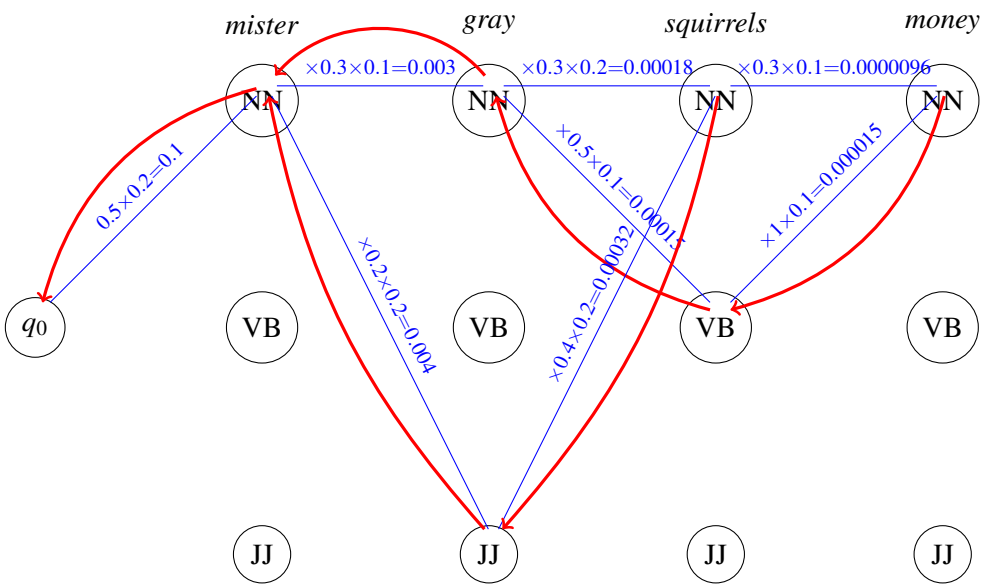
a. Assume the tags are generated by a hidden Markov model. Based on the training data, estimate the *transition probabilities* of the model.



b. Now estimate the *emission probabilities*. Only write down the probabilities for the words *mister*, *gray*, *squirrels*, and *money*.

	mister	gray	squirrels	money
NN	$\frac{4}{20} = 0.2$	$\frac{2}{20} = 0.1$	$\frac{4}{20} = 0.2$	$\frac{2}{20} = 0.1$
VB	0	0	$\frac{1}{10} = 0.1$	0
JJ	0	$\frac{2}{10} = 0.2$	0	0

c. Now use the Viterbi algorithm to decode the utterance *mister gray squirrels money*. Remember the backpointers. Do not draw arrows for zero probabilities. Show your work.



d. What is the most likely tag sequence for the utterance *mister gray squirrels*?

NN JJ NN

e. What is the most likely tag sequence for *mister gray squirrels money*?

NN NN VB NN

f. Did any of the tags change from part (d) to part (e)? Why or why not?

Yes, the tags changed. The reason is the transition probabilities: the transition VB → NN has a substantially higher probability than NN → NN. Even though VB was not the most likely tag for *squirrels* at that point, the following transition to *money* (which is unambiguously NN) makes the posterior probability of VB highest for *squirrels*.

Problem 7. A program intended to be a hidden Markov model part-of-speech tagger had a bug which caused it to ignore the previous state when estimating transition probabilities. That is, instead of estimating transition probabilities $a(q_i, q_j) \leftarrow P(state = q_j | prev\_state = q_i)$, it estimated the transition matrix as $a(q_i, q_j) \leftarrow P(state = q_j)$ for every state q_i. The emission matrix was not affected by the bug, so it was estimated properly: $b(q_i, v_j) \leftarrow P(observation = v_j | state = q_i)$.

a. Is the buggy model still a Markov chain? Explain why or why not.

Yes, this is a Markov chain, because the transitions from each state constitute a probability distribution: for each state i, $\sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} P(q_j) = 1$. (The emissions from each state also form a probability distribution since they're the same as in a standard HMM.)

b. If the buggy model is used for decoding, what tag will it give to each word? Explain why.

The buggy model will assign each word the most probable tag in the corpus for that word. This is because the joint probability at time t of transitioning to q and observing o_t will be $P(q) \cdot P(o_t|q) \propto P(q|o_t)$ (Bayes' theorem); therefore the most probable tag at time t will be $\arg\max_q P(q|o_t)$, that is the most likely tag for the word o_t. The Viterbi algorithm will always choose backpointers to the previous most probable state, since the multipliers from each incoming state are identical.