Hidden Markov Model

Introduction:

There are different Japanese text parsers available, two of them are the Chasen morphological analyzer and the MeCab parser. The Chasen morphological analyzer is based on the Hidden Markov Model while the MeCab parser is based on the Conditional Random Field. In my project, I will need a parser that will separate the Japanese text/kanjis entered by the user. Japanese texts are different from English texts in that, there are no word boundaries in Japanese texts. Still, the parsing techniques for Japanese text are also dependent on the Hidden Markov Model. In this report, I am going to explain what HMM is, why it is used, working of the HMM and the Viterbi algorithm giving my example.

What HMM is?

- The Hidden Markov Model is based on the Markov model. The Markov Model is a finite automata model which is used to find out what will be the next state depending on the current state. Hence to predict the future, you do not have to look at the past history of states of the model.
  e.g. Consider the sentence "The monster swallowed _____". In this sentence, after the verb swallowed, it is less likely that the next word will be a verb again. Hence by looking at the current word which is say "swallowed", we can say that next word will be the most likely some article such as "a", followed by noun such as "boy" The purpose of Markov Model is to find the next word with the highest probability. The transition probability is assigned to each path. It is better to follow the path of traversing from one state to the other with the highest probability.
- The Markov Models are basically used to model sequence of events. These sequence of events could be fixed or not. The deterministic flow of events can be traffic lights turning from Green to Yellow to Red. The non-deterministic flow of events can be analyzing weather conditions.
- In the Markov Model, the output i.e., sequence of events/observations is simply the sequence of states visited.
- In the HMM, the states that the model is passed through is unknown to us. We only know the observation sequence, probabilities of transitioning from one state to other, the probability that for the current state what are the chances that given observation state will be generated.
- Hence, in the HMM, our purpose is to find the most likely sequence of states which could yield the observed sequence of symbols.

Why it is used?

- The HMM can be used in various applications such as speech recognition, part-of-speech tagging etc.
- I will explain POS (Part-Of-Speech) tagging with the HMM. In English, there are different types of POS tags such as DT(determiner), N(noun), V(verb) etc. Consider the sentence: The chocolate is sweet.

Here, the => DT,
chocolate => N
is => V
sweet => ADJ

We would like to find out what is more likely the tag sequence. We want to find out the best tag sequence. The HMM will be used to find the sequence of states that will lead to the highest probability of tag t for given word sequence w. Hence, as per the Markov assumption, the word w_i depends on tag ti and tag t_i depends on tag t_{i-1}

How the HMM works?:

- In the HMM, we know what the observable sequence is but we do not know what states it is been travelled to get this sequence.
- In the HMM, the states are not observable. We know some probabilities that the model will pass through.
- The HMM works as any other finite state machine having a set of hidden states, observable states, transition probabilities, initial state probabilities and output probabilities. The current state is not observable but each state produces an output with a certain probability.
- The HMM consists of a set of states as in any other finite automata model. Say states {q,r}, transition probabilities i.e., traveling from the state q to r for the given observation state, a set of observations such as string “aabb” and the probability of getting a symbol “b” for the current state say “r”
- The problem with the HMM is how to find out the state sequence that best explains the given set of observations? Because it is possible to have more than one state sequences that will lead to the same observation sequence. This problem could be solved with the use of the Viterbi algorithm.
- The working of the Viterbi algorithm is based on the dynamic programming approach. In the dynamic programming, the problem is divided into subproblems and the after solving each subproblem, the algorithm saves its answer in the table that could be used to solve the actual problem. The dynamic programming algorithm is used for solving problems having more than one possible solutions. The dynamic programming helps to find out the optimal solution for the given problem.
- Similarly in the Viterbi algorithm, the purpose is to find the best path sequence that will lead to the given observation states.
- My example for the Viterbi algorithm for given HMM is as follows:
  Sentence: Eye drops off shelf.
  As per the HMM,
  Set of states = {q, r}
  Set of observations = {eye, drops, off, shelf}

As the Viterbi algorithm follows the dynamic programming approach, it is required to store the best path and its probability for each possible state.

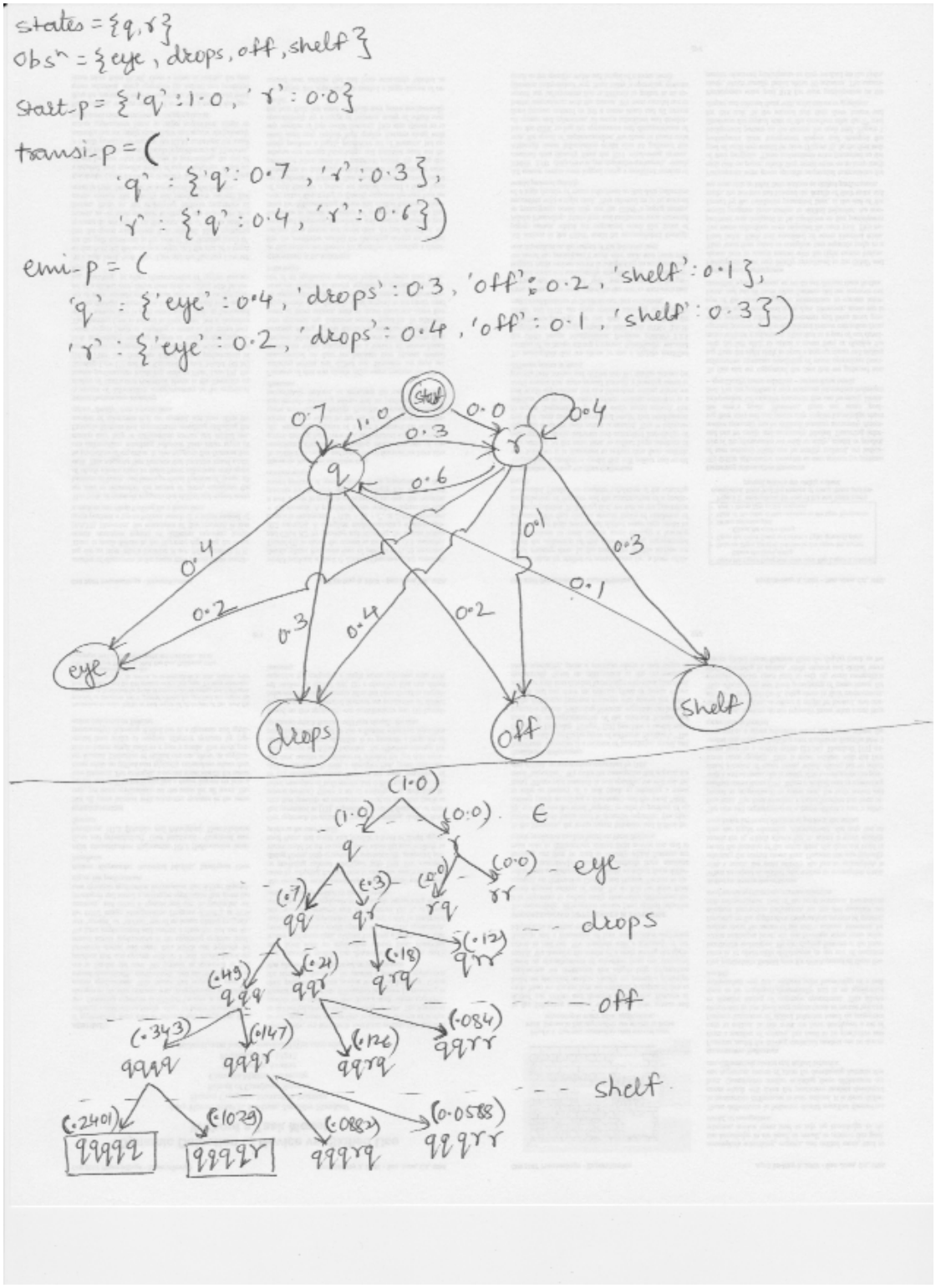

Figure 1: Example of HMM

HMM Training

We have seen that what the HMM is, why it is used and how it works. Basically there are three problems with the given Hidden Markov Model.
They are as follows:

1. Given the Hidden Markov Model and a sequence of observations, we would like to know what is the sequence of hidden states that leads to the given set of observations? This problem as described above can be solved using the Viterbi algorithm.
2. Given the Hidden Markov Model as well as emission and transition probabilities, what will be the probability of a given sequence of observations? This problem can be solved using the Forward algorithm. The Forward algorithm is similar to the Viterbi algorithm except the purpose of the Forward algorithm is find the sum of all possible ways to get to some end state. Thus, in the Forward algorithm, the probability of reaching to some end state is the total of product of all the probabilities of paths leading to that end state.
   In simple words,
   Forward Probability at timestep (t + 1) = Sum of (Forward Probability at timestep t * transition probability that will end up in that state)
   Consider the above example of the Viterbi algorithm.

Time ticks	1	2	3	4	5
input	ϵ	eye	eye drops	eye drops off	eye drops off shelf
Forward Probability of state q at timestamp t	1.0	0.28	0.0876	0.0149	0.0023
Forward Probability of state r at timestamp t	0.0	0.12	0.0444	0.007	0.0012
Total Probability	1.0	0.4	0.132	0.0219	**0.0035**

Table 1: Forward probabilities for "eye drops off shelf"

As explained earlier, given the observation "eye", what is the total probability i.e., sum of the probabilities of all paths that one will end up in state "q" or in state "r"
Hence, in above example,

forward probability of state q at time 2
= (forward probability of state q at time 1 * transition probability from q -> q * emission probability for observation eye from state q)
+ (forward probability of state r at time 1 * transition probability from r -> q * emission probability for observation eye from state r)

= (1.0 * 0.7 * 0.4) + (0.0 * 0.6 * 0.2)
= 0.28
Similarly, we can calculate the forward probability for other states and at different timestamps. As we are moving forward in the sequence, this is called as forward probabilities.
We can also use the Backward algorithm instead of the Forward algorithm. The difference between the Backward algorithm and the Forward algorithm is, in the Backward algorithm we calculate the probability of sequence by working backward.

Time ticks	1	2	3	4	5
Input	eye drops off shelf	drops off shelf	off shelf	shelf	ϵ
Backward probability of state q at timestamp t	**0.0035**	0.0083	0.032	0.1	1
Backward probability of state r at timestamp t		0.0105	0.018	0.3	1

Table 2: Backward probabilities for "eye drops off shelf"

Hence, in the above example,
the backward probability of state q at time 2
= (backward probability of state q at time 3 * transition probability from q -> q * emission probability for observation drops at state q)
+ (backward probability of state r at time 3 * transition probability from q -> r * emission probability for observation drops at state q)

= (0.032 * 0.7 * 0.3) + (0.018 * 0.3 * 0.3)
= 0.0083

3. Given an output sequence O, what HMM transition probabilities maximize the likelihood of the sequence? i.e We have the model and a set of all observations, how can we adjust the parameters to increase the probability of the observations given the model? This problem can be solved using the Forward-Backward algorithm, also known as the Baum-Welch algorithm. Using an initial parameter instantiation, the Forward-Backward algorithm iteratively re-estimates the parameters and improves the probability that given observations are generated by the new parameters. So, we need to re-estimate three parameters:
   - initial state distribution
   - transition probabilities
   - emission probabilities

How HMM training works?

- The purpose of using the HMM training is to find the HMM that maximizes the probability of the given observation sequence. We know the observation sequence, we have to find the three parameters as described above.
- To achieve this we will use the Forward-Backward/Baum-Welch algorithm. We follow the hill climbing approach. So we start with some random HMM model and iteratively make small changes to the solution, improving it a little each time. When there is no more improvement possible, the algorithm terminates.
- So we do not know what the model is. We can work out the probability of the observation sequence using some randomly chosen model. By looking at it, we can see which state transitions and emissions were probably used the most. By increasing the probability of those, we can choose a revised better model which gives a higher probability to the observation sequence. Thus, we are following the process of maximization which is referred to as training the model.
- Hence, we will guess randomly transition probabilities from state q to state r. But then how can we find the new re-estimated probability?
  New probability from state q to r = (forward probability from state q at timestamp t *
  backward probability from state r at timestamp t+1 *
  transition probability from state q to state r for the given observation)
- We will use this newly calculated probability for re-estimating start probabilities, transition probabilities and emission probabilities. Consider following diagram for guessed HMM:

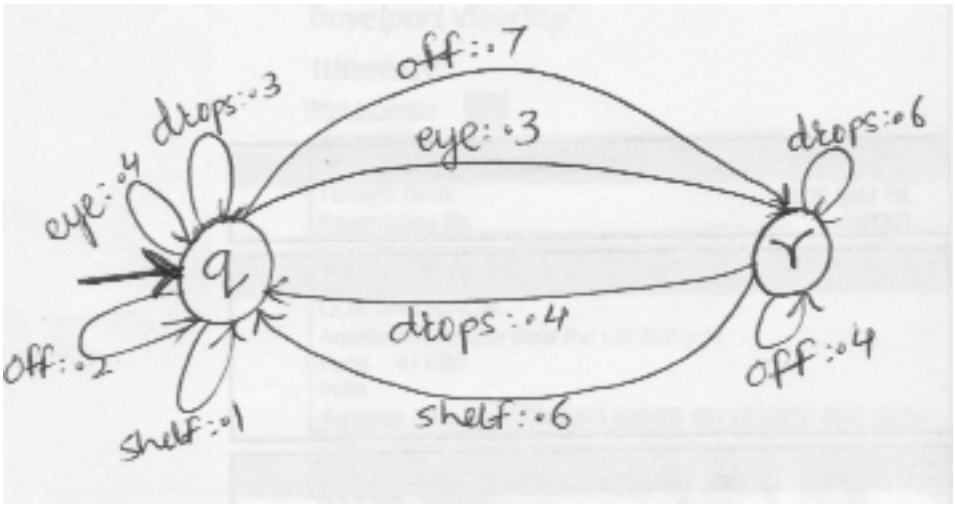

Figure 2: Guessed HMM

Time ticks	t1	t2	t3	t4		
Input	eye	drops	off	shelf	**Total P**	**New P**
q -> r for observation eye	0.003	0	0	0	0.003	0.23
r -> r for observation drops	0	0.001	0	0	0.001	0.08
r -> r for observation off	0	0	0.005	0	0.005	0.38
r -> q for observation shelf	0	0	0	0.004	0.004	0.3
					0.013	

Table 3: New probabilities for "eye drops off shelf"

Thus, in the above example,
Total probability from state q to state r for observation eye =

forward probability of state q at time t1 *
transition probability from state q to state r for observation ‘eye’ *
backward probability of state r at time t2

= 1.0 * 0.3 * 0.0105
= 0.00315
≈ 0.003
Similarly, we can calculate the new probability for other observations as well. After calculating all the total probabilities for remaining transitions, we calculate its total i.e. Summation(Total P). This is nothing but the expected number of transitions from one state to the other state.

New probability from state q to state r =
Total P from state q to state r / Summation (Total P)

= 0.003 / 0.013
= 0.23

Our purpose of calculating total probabilities is we would like to find out the new probability that will be greater than the previous probability. If we think that the new probability and old probability are somewhat similar, we will stop iterating through the HMM and that will be our new re-estimated parameters.