Computational Linguistics

Sequence Modeling: HMM, MEMM, and CRF

Parameswari Krishnamurthy

Language Technologies Research Centre
IIIT-Hyderabad

param.krishna@iiit.ac.in

Sequence modeling

- **Sequence modeling** involves predicting or labeling sequences of data, such as words in a sentence.

Sequence modeling

- **Sequence modeling** involves predicting or labeling sequences of data, such as words in a sentence.
- Common tasks: **Part-of-Speech (POS) tagging** and **Named Entity Recognition (NER)**.

Sequence modeling

- **Sequence modeling** involves predicting or labeling sequences of data, such as words in a sentence.
- Common tasks: **Part-of-Speech (POS) tagging** and **Named Entity Recognition (NER)**.
- Models used:
  - **Hidden Markov Model (HMM)**
  - **Maximum Entropy Markov Model (MEMM)**
  - **Conditional Random Field (CRF)**

1. Hidden Markov Models (HMM)

- A Hidden Markov Model (HMM) is used to model systems where we observe outputs but not the underlying states.
- In the context of POS tagging:
  - **Hidden States:** The sequence of POS tags (NN, VB, etc.) that we aim to predict.
  - **Observable Symbols:** The actual words in the sentence.

HMM in POS Tagging

Example Sentence: *"Secretariat is expected to race tomorrow"*

- The goal is to assign the correct POS tag to each word.
- Words are observable symbols; POS tags are hidden states.
- HMM helps us determine the most likely sequence of POS tags given the sentence.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.
  - *Example:* $P(\text{VB} \mid \text{NN})$, i.e., the probability of a Verb following a Noun.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.
  - *Example:* $P(\text{VB} \mid \text{NN})$, i.e., the probability of a Verb following a Noun.
- **Emission Probabilities:** Probability of an observation being generated from a state.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.
  - *Example:* $P(\text{VB} \mid \text{NN})$, i.e., the probability of a Verb following a Noun.
- **Emission Probabilities:** Probability of an observation being generated from a state.
  - *Example:* $P(\text{"race"} \mid \text{VB})$, i.e., the probability of the word “race” being generated given that the state is Verb.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.
  - *Example:* $P(\text{VB} \mid \text{NN})$, i.e., the probability of a Verb following a Noun.
- **Emission Probabilities:** Probability of an observation being generated from a state.
  - *Example:* $P(\text{"race"} \mid \text{VB})$, i.e., the probability of the word “race” being generated given that the state is Verb.
- **Initial State Distribution:** Probability distribution over the initial states.

Components of HMM

- **States:** Hidden variables that influence the observed outcomes.
  - *Example:* POS tags like Noun (NN), Verb (VB), etc.
- **Observations:** The visible outcomes influenced by the hidden states.
  - *Example:* Words in a sentence like “dog”, “runs”, etc.
- **Transition Probabilities:** Probability of transitioning from one state to another.
  - *Example:* $P(\text{VB} \mid \text{NN})$, i.e., the probability of a Verb following a Noun.
- **Emission Probabilities:** Probability of an observation being generated from a state.
  - *Example:* $P(\text{"race"} \mid \text{VB})$, i.e., the probability of the word “race” being generated given that the state is Verb.
- **Initial State Distribution:** Probability distribution over the initial states.
  - *Example:* $P(\text{NN})$, i.e., the probability that the first word in a sentence is a Noun.

Why is it “Hidden”?

- The term “hidden” refers to the fact that the POS tags (states) are not directly observable.

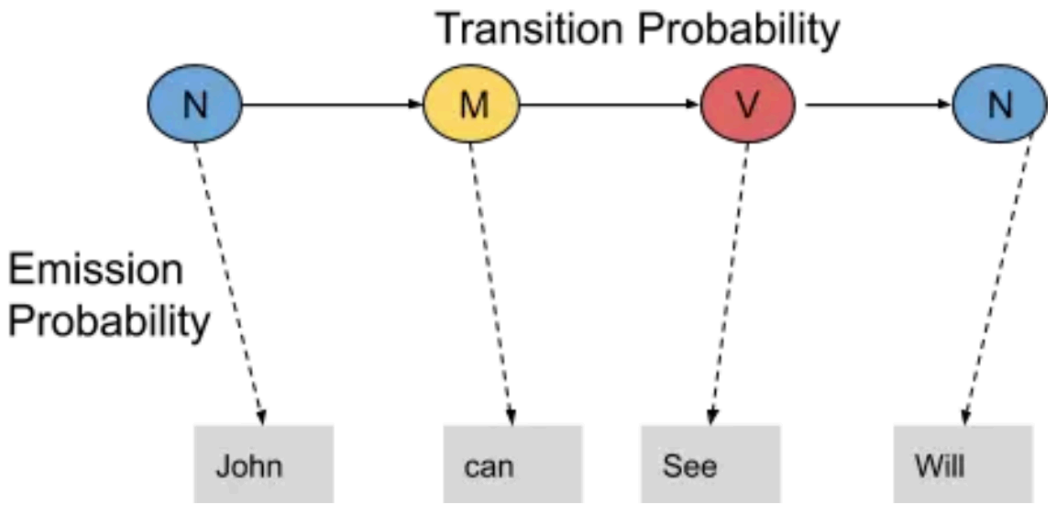
Why is it “Hidden”?

- The term “hidden” refers to the fact that the POS tags (states) are not directly observable.
- We only see the words (observable symbols) and must infer the POS tags using HMM.

Why is it “Hidden”?

- The term “hidden” refers to the fact that the POS tags (states) are not directly observable.
- We only see the words (observable symbols) and must infer the POS tags using HMM.
- HMM finds the most likely sequence of POS tags that could produce the observed sequence of words.

HMM in Pos tagging



POS Tagging Example

- `John can see Will`
- In this example, we consider three POS tags: Noun, Model, and Verb.
- Let the sentence be tagged as **Noun, Model, Verb, Noun**.
- To calculate the probability associated with this particular sequence of tags, we need:

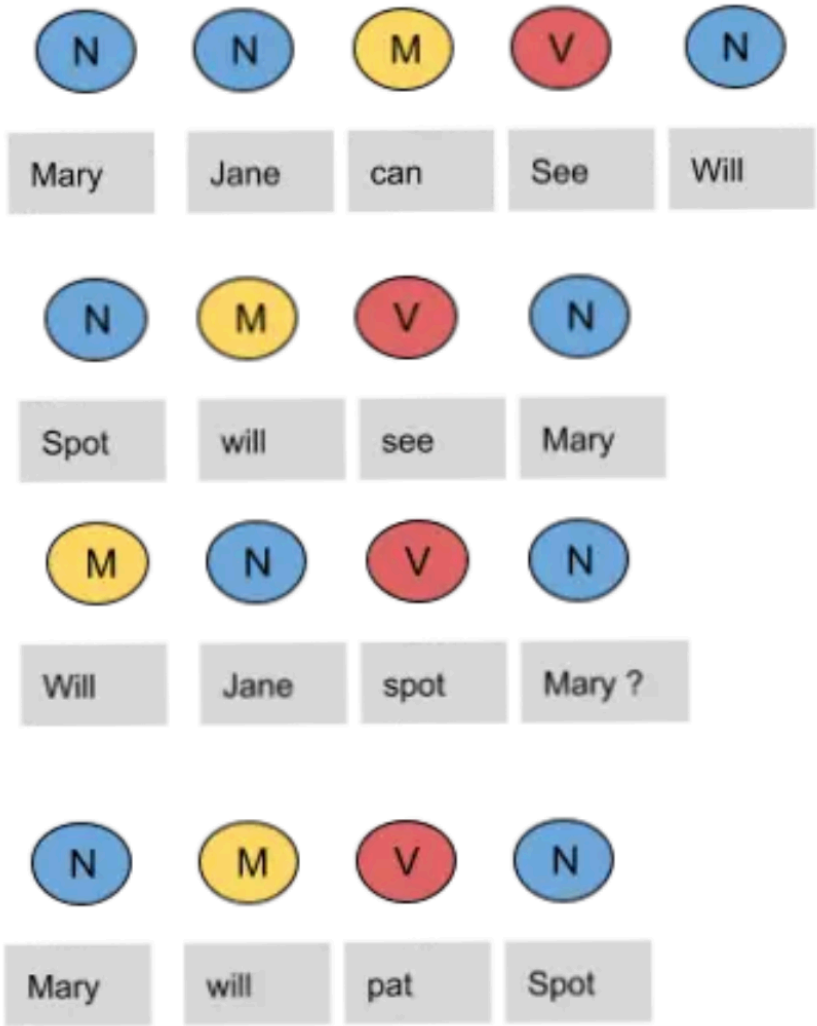
POS Tagging Example

- `John can see Will`
- In this example, we consider three POS tags: Noun, Model, and Verb.
- Let the sentence be tagged as **Noun, Model, Verb, Noun**.
- To calculate the probability associated with this particular sequence of tags, we need:
  - **Transition Probabilities:** The likelihood of a particular sequence of tags. For example:
    - How likely is it that a Noun is followed by a Model?
    - How likely is it that a Model is followed by a Verb?
    - How likely is it that a Verb is followed by a Noun?

POS Tagging Example

- `John can see Will`
- In this example, we consider three POS tags: Noun, Model, and Verb.
- Let the sentence be tagged as **Noun, Model, Verb, Noun**.
- To calculate the probability associated with this particular sequence of tags, we need:
  - **Transition Probabilities:** The likelihood of a particular sequence of tags. For example:
    - How likely is it that a Noun is followed by a Model?
    - How likely is it that a Model is followed by a Verb?
    - How likely is it that a Verb is followed by a Noun?
  - **Emission Probabilities:** The likelihood of each word given its tag. For example:
    - The probability that "John" is a Noun.
    - The probability that "can" is a Model.
    - The probability that "see" is a Verb.
    - The probability that "Will" is a Noun.
- For accurate tagging, both transition and emission probabilities should be high for the given sequence.

More Examples



Emission Probabilities Counting Table

- In the previous sentences, the word “Mary” appears four times as a Noun.

Emission Probabilities Counting Table

- In the previous sentences, the word "Mary" appears four times as a Noun.
- To calculate the emission probabilities, we create a counting table as follows:

Words	**Noun**	**Model**	**Verb**
Mary	4	0	0
Jane	2	0	0
Will	1	3	0
Spot	2	0	1
Can	0	1	0
See	0	0	2
pat	0	0	1

Table: Counting Table for Emission Probabilities

Normalized Emission Probabilities

- We divide each column by the total number of appearances for that POS tag.

Normalized Emission Probabilities

- We divide each column by the total number of appearances for that POS tag.
- For example, 'Noun' appears 9 times, so we divide each value in the 'Noun' column by 9.

Words	Noun	Model	Verb
Mary	$\frac{4}{9}$	0	0
Jane	$\frac{2}{9}$	0	0
Will	$\frac{1}{9}$	$\frac{3}{4}$	0
Spot	$\frac{2}{9}$	0	$\frac{1}{4}$
Can	0	$\frac{1}{4}$	0
See	0	0	$\frac{2}{4}$
pat	0	0	1

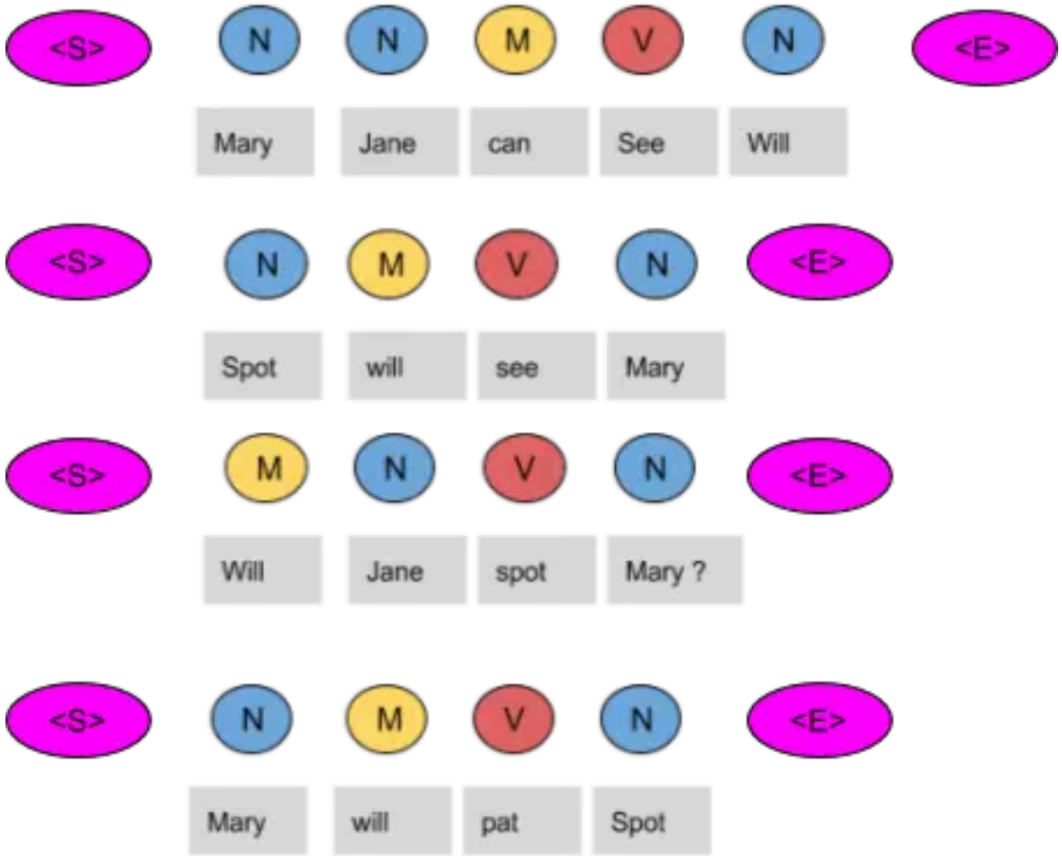
Table: Normalized Emission Probabilities

Calculating Transition Probabilities

- To calculate transition probabilities, we introduce two additional tags: <S> and <E>, <S> is placed at the beginning of each sentence and <E> is placed at the end of each sentence.

Calculating Transition Probabilities

- To calculate transition probabilities, we introduce two additional tags: <S> and <E>, <S> is placed at the beginning of each sentence and <E> is placed at the end of each sentence.
- These tags help to define the boundaries of the sentences and manage transitions between POS tags.



Transition Probability Table

- Let us create a table and fill it with the co-occurrence counts of the tags:

	N	**M**	**V**	**<E>**
<S>	3	1	0	0
N	1	3	1	4
M	1	0	3	0
V	4	0	0	0

Table: Co-occurrence Counts of POS Tags

- In the table:
  - The <S> tag is followed by the N tag three times, so the first entry is 3.
  - The Model tag follows the <S> once, so the second entry is 1.
  - The rest of the table is filled in a similar manner.

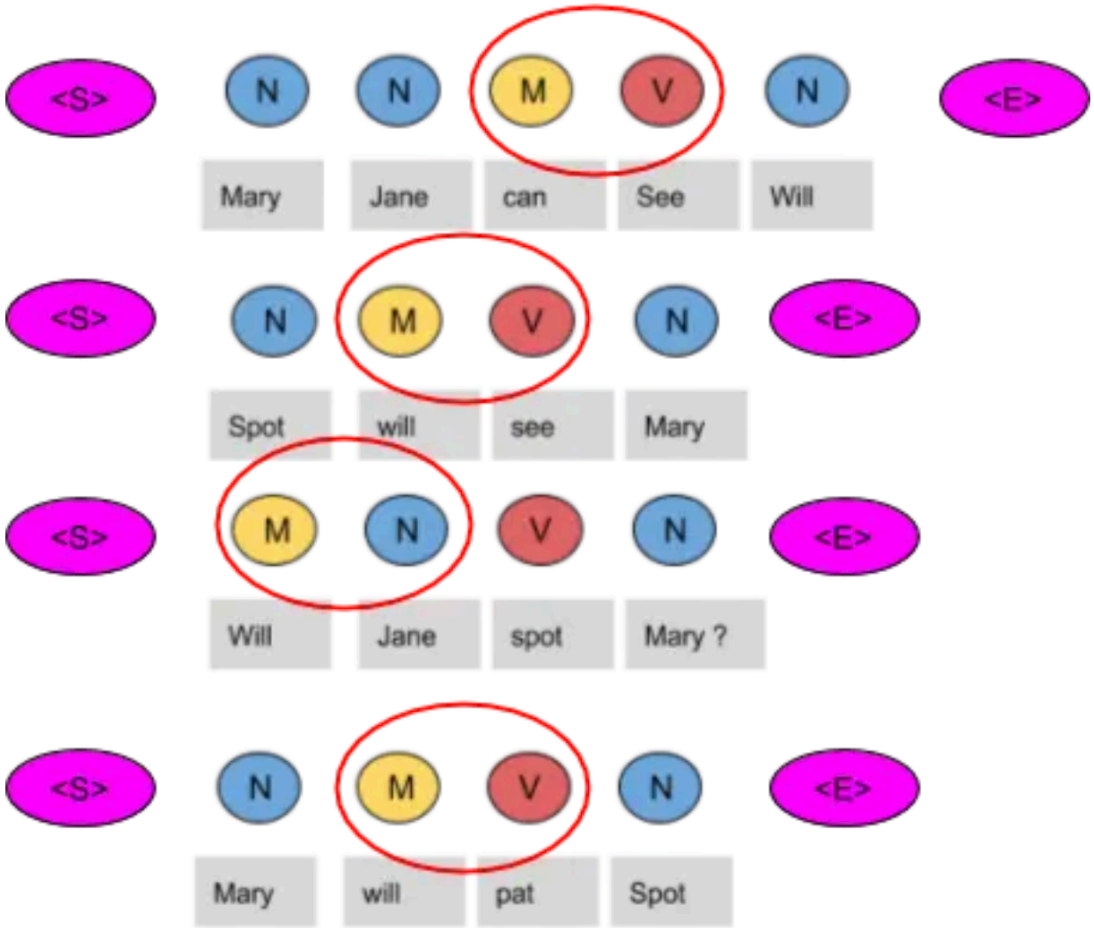
- Next, divide each term in a row by the total number of co-occurrences for that tag.

- Next, divide each term in a row by the total number of co-occurrences for that tag.
  - For example, the Model tag is followed by any other tag four times.
  - Divide each element in the third row by 4.

	N	**M**	**V**	**<E>**
<S>	$\frac{3}{4}$	$\frac{1}{4}$	0	0
N	$\frac{1}{9}$	$\frac{3}{9}$	$\frac{1}{9}$	$\frac{4}{9}$
M	$\frac{1}{4}$	0	$\frac{3}{4}$	0
V	$\frac{4}{4}$	0	0	0

Table: Normalized Transition Probabilities

- These are the respective transition probabilities for the given sentences.
- Now, how does the HMM determine the appropriate sequence of tags for a new sentence using these probabilities?

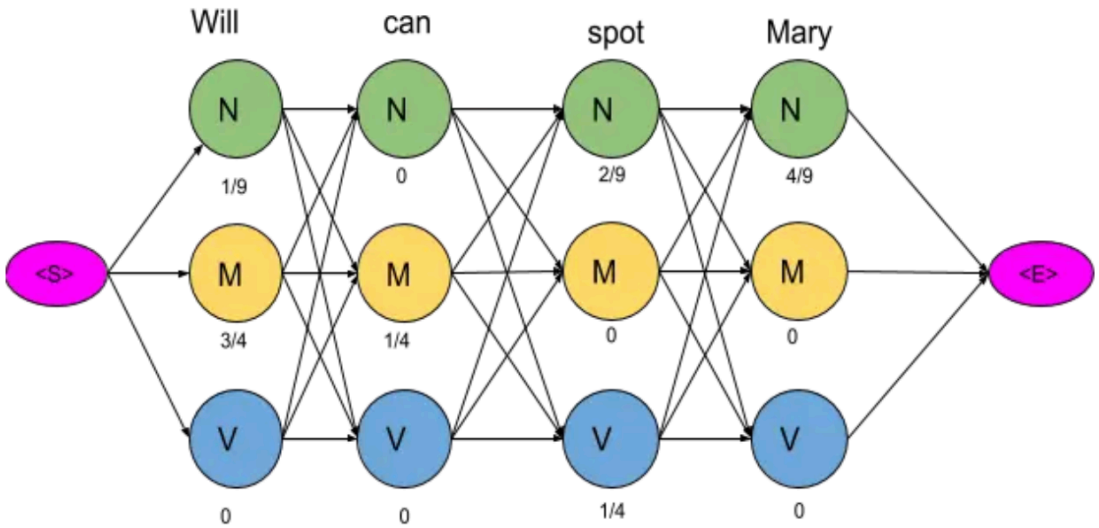

<S>
N
N
M
V
N
<E>
Mary
Jane
can
See
Will
<S>
N
M
V
N
<E>
Spot
will
see
Mary
<S>
M
N
V
N
<E>
Will
Jane
spot
Mary ?
<S>
N
M
V
N
<E>
Mary
will
pat
Spot

Example Sentence

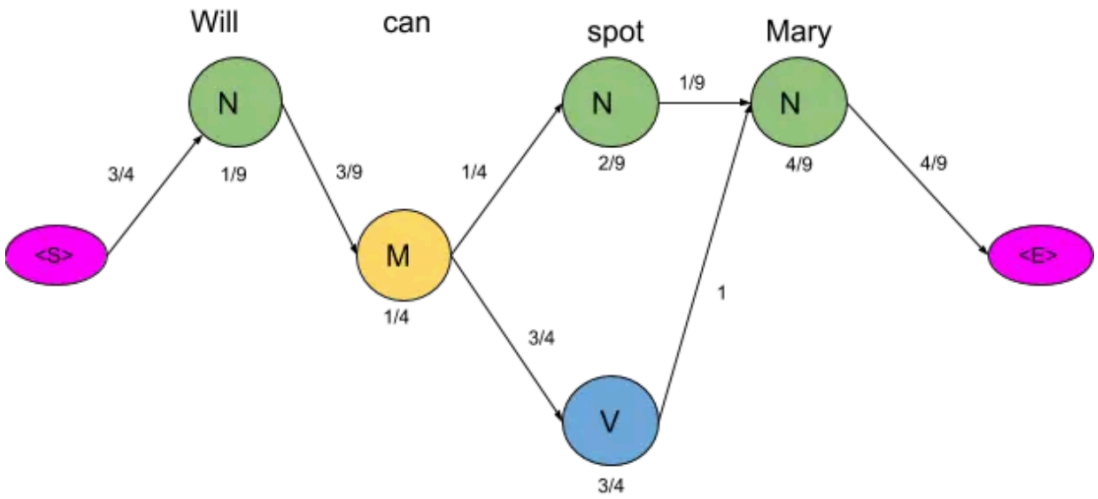
Consider the sentence: *"Will can spot Mary"*

- Suppose the sentence is tagged incorrectly as follows:
  - **Will** as a **Model**
  - **Can** as a **Verb**
  - **Spot** as a **Noun**
  - **Mary** as a **Noun**

81 possible combinations for the sentence 'Will can spot Mary'



Delete all the vertices and edges with probability zero and also include transition probability



Path 1

<S> → N → M → N → N → <E>

$$\text{Probability} = \frac{3}{4} \times \frac{1}{9} \times \frac{3}{9} \times \frac{1}{4} \times \frac{1}{4} \times \frac{2}{9} \times \frac{1}{9} \times \frac{4}{9} \times \frac{4}{9} = 0.00000846754$$

Path 2

<S> → N → M → V → N → <E>

$$\text{Probability} = \frac{3}{4} \times \frac{1}{9} \times \frac{3}{9} \times \frac{1}{4} \times \frac{3}{4} \times \frac{1}{4} \times 1 \times \frac{4}{9} \times \frac{4}{9} = 0.00025720164$$

- Clearly, the probability of the second sequence is much higher
- Hence the HMM tags each word in the sentence according to this sequence.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.
- **Vertex with Two Paths:**
  - Analyze the two mini-paths leading to the vertex.
  - Focus on the path with the lowest probability.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.
- **Vertex with Two Paths:**
  - Analyze the two mini-paths leading to the vertex.
  - Focus on the path with the lowest probability.

Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.
- **Vertex with Two Paths:**
  - Analyze the two mini-paths leading to the vertex.
  - Focus on the path with the lowest probability.
- **Path Pruning:**
  - Calculate probabilities for all paths leading to a node.
  - Remove edges with lower probability.
  - Nodes with zero probability have no edges.

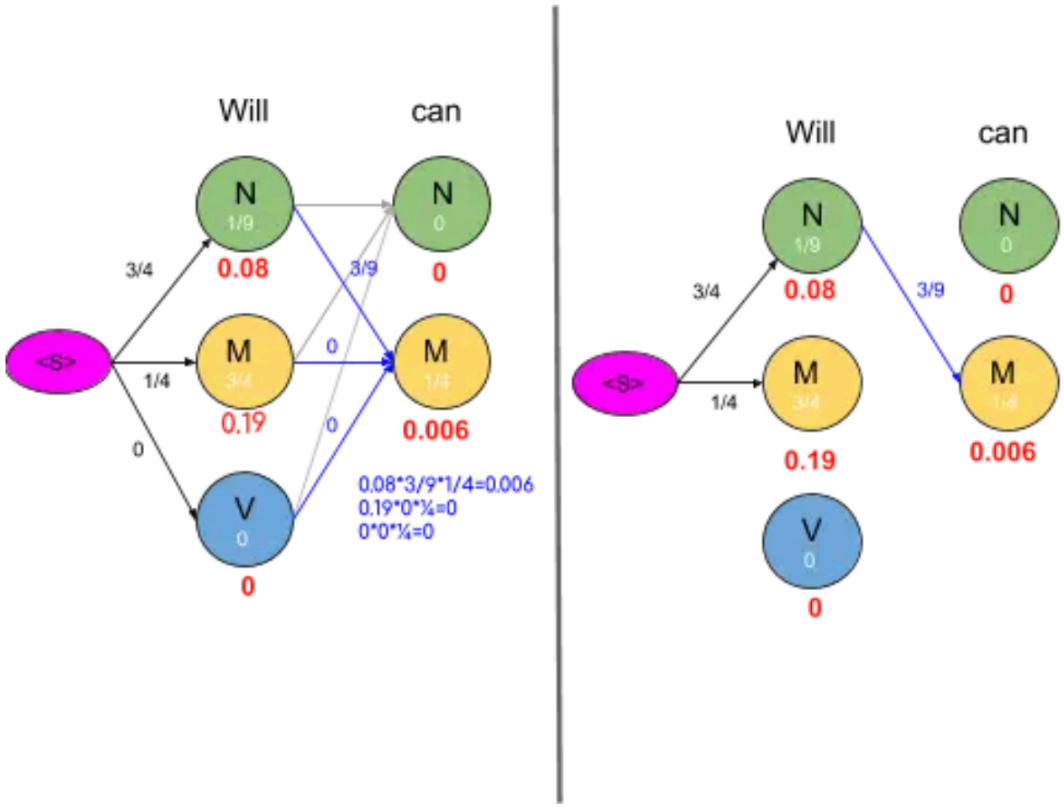
Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.
- **Vertex with Two Paths:**
  - Analyze the two mini-paths leading to the vertex.
  - Focus on the path with the lowest probability.
- **Path Pruning:**
  - Calculate probabilities for all paths leading to a node.
  - Remove edges with lower probability.
  - Nodes with zero probability have no edges.

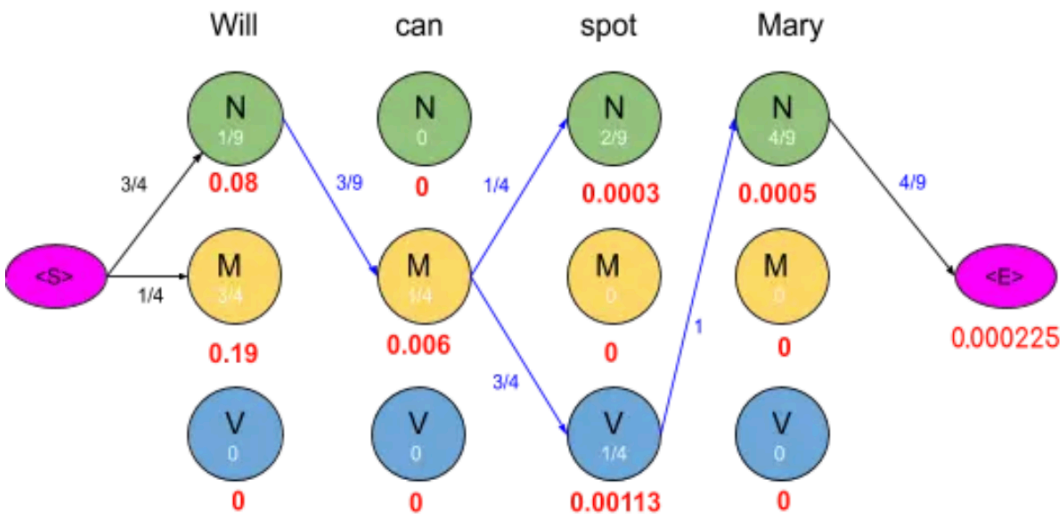
Optimizing HMM using the Viterbi Algorithm

- **Previous Optimization:**
  - Reduced the number of paths from 81 to 2.
- **Further Optimization with Viterbi:**
  - Apply the Viterbi algorithm to further reduce computations.
  - Consider the same example and apply the Viterbi algorithm.
- **Vertex with Two Paths:**
  - Analyze the two mini-paths leading to the vertex.
  - Focus on the path with the lowest probability.
- **Path Pruning:**
  - Calculate probabilities for all paths leading to a node.
  - Remove edges with lower probability.
  - Nodes with zero probability have no edges.

Consider the mini path having the lowest probability.



- **Nodes with Zero Probability:**
  - Some nodes may have a probability of zero.
  - Such nodes have no edges attached, as all paths to them have zero probability.



Key Algorithms in HMM

- **Forward Algorithm:**
  - Calculates the probability of an observation sequence given the model.
  - *Formula:*

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\alpha_t(i)$ is the probability of observing the sequence up to time t and being in state i.

Key Algorithms in HMM

- **Forward Algorithm:**
  - Calculates the probability of an observation sequence given the model.
  - *Formula:*

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\alpha_t(i)$ is the probability of observing the sequence up to time t and being in state i.

Key Algorithms in HMM

- **Forward Algorithm:**
  - Calculates the probability of an observation sequence given the model.
  - *Formula:*

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\alpha_t(i)$ is the probability of observing the sequence up to time t and being in state i.
- **Backward Algorithm:**
  - Calculates the probability of the ending state given the observation sequence.
  - *Formula:*

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \mid Q_t = S_i, \lambda)$$

  - Where $\beta_t(i)$ is the probability of observing the sequence from time $t+1$ to the end given the state i at time t.

Key Algorithms in HMM

- **Forward Algorithm:**
  - Calculates the probability of an observation sequence given the model.
  - *Formula:*

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\alpha_t(i)$ is the probability of observing the sequence up to time t and being in state i.
- **Backward Algorithm:**
  - Calculates the probability of the ending state given the observation sequence.
  - *Formula:*

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \mid Q_t = S_i, \lambda)$$

  - Where $\beta_t(i)$ is the probability of observing the sequence from time $t+1$ to the end given the state i at time t.

- **Viterbi Algorithm:**
  - Finds the most probable sequence of hidden states.
  - *Formula:*

$$\delta_t(i) = \max_{Q_1, Q_2, \ldots, Q_{t-1}} P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\delta_t(i)$ represents the highest probability of the state sequence up to time t ending in state i.

- **Viterbi Algorithm:**
  - Finds the most probable sequence of hidden states.
  - *Formula:*

$$\delta_t(i) = \max_{Q_1, Q_2, \ldots, Q_{t-1}} P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\delta_t(i)$ represents the highest probability of the state sequence up to time t ending in state i.

- **Viterbi Algorithm:**
  - Finds the most probable sequence of hidden states.
  - *Formula:*

$$\delta_t(i) = \max_{Q_1, Q_2, \ldots, Q_{t-1}} P(O_1, O_2, \ldots, O_t, Q_t = S_i \mid \lambda)$$

  - Where $\delta_t(i)$ represents the highest probability of the state sequence up to time t ending in state i.
- **Baum-Welch Algorithm:**
  - Used for training HMM, updating model parameters using observed data.
  - *Procedure:* Iteratively adjusts transition and emission probabilities to maximize the likelihood of the observed data.

Challenges and Limitations of HMM

Assumption of Markov Property:	• Current state depends only on the previous state.
Fixed Number of States:	• HMMs require a predefined number of states.
Data Sparsity:	• Difficulty in estimating probabilities with limited data.
Long-Range Dependencies:	• HMMs struggle with capturing dependencies beyond the immediate previous state.

Maximum Entropy Markov Model

- MEMM is a discriminative model used for sequence labeling tasks like POS tagging and NER.
- It combines the strengths of Maximum Entropy and Markov models.

Maximum Entropy Markov Model

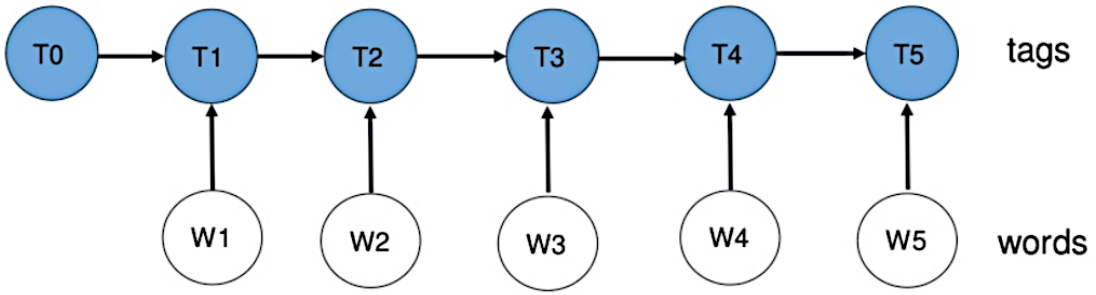
- MEMM is a discriminative model used for sequence labeling tasks like POS tagging and NER.
- It combines the strengths of Maximum Entropy and Markov models.
- **Conditional Probability:**
  - MEMM models the conditional probability of a tag given the previous tag and the current observation.

Maximum Entropy Markov Model

- MEMM is a discriminative model used for sequence labeling tasks like POS tagging and NER.
- It combines the strengths of Maximum Entropy and Markov models.
- **Conditional Probability:**
  - MEMM models the conditional probability of a tag given the previous tag and the current observation.
- **Feature-Based:**
  - MEMM allows the use of arbitrary features of the input data.
  - Examples include word suffixes, prefixes, surrounding words, capitalization, etc.

Maximum Entropy Markov Model

- MEMM is a discriminative model used for sequence labeling tasks like POS tagging and NER.
- It combines the strengths of Maximum Entropy and Markov models.
- **Conditional Probability:**
  - MEMM models the conditional probability of a tag given the previous tag and the current observation.
- **Feature-Based:**
  - MEMM allows the use of arbitrary features of the input data.
  - Examples include word suffixes, prefixes, surrounding words, capitalization, etc.



Discriminative model, model conditional probability Pr(T | W) directly.

$$\Pr(\boldsymbol{T}|\boldsymbol{W}) = \prod_{i=1}^{L} \Pr(t_i|t_{i-1}, w_i) = \prod_{i=1}^{L} \frac{\exp(\sum_j \beta_j f_j(t_{i-1}, w_i))}{Z(t_{i-1}, w_i)}$$

t_0 is a dummy start state.

Example of POS tagging with MEMM. Source: Gui et al. 2019, slide 46.

HMM vs. MEMM: Generative vs. Discriminative Models

- **HMM (Hidden Markov Model): Generative Model**
  - Words are modeled as observations generated from hidden states.
  - Probabilities are formulated using:
    - *Likelihood:* $P(W \mid T)$
    - *Prior:* $P(T)$
  - Maximizes the joint probability $P(W, T)$ for decoding the tag sequence.

HMM vs. MEMM: Generative vs. Discriminative Models

- **HMM (Hidden Markov Model): Generative Model**
  - Words are modeled as observations generated from hidden states.
  - Probabilities are formulated using:
    - *Likelihood:* $P(W \mid T)$
    - *Prior:* $P(T)$
  - Maximizes the joint probability $P(W, T)$ for decoding the tag sequence.

HMM vs. MEMM: Generative vs. Discriminative Models

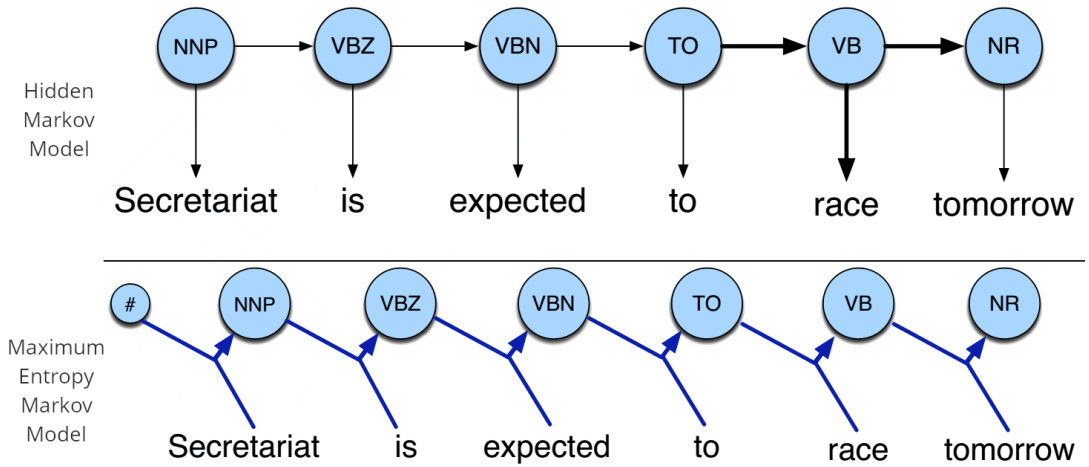
- **HMM (Hidden Markov Model): Generative Model**
  - Words are modeled as observations generated from hidden states.
  - Probabilities are formulated using:
    - *Likelihood:* $P(W \mid T)$
    - *Prior:* $P(T)$
  - Maximizes the joint probability $P(W, T)$ for decoding the tag sequence.
- **MEMM (Maximum Entropy Markov Model): Discriminative Model**
  - Directly models the posterior probability $P(T \mid W)$.
  - Discriminates among possible tag sequences given a word sequence.
  - Uses conditional probability, conditioned on the previous tag and current word.

HMM vs. MEMM: Generative vs. Discriminative Models

- **HMM (Hidden Markov Model): Generative Model**
  - Words are modeled as observations generated from hidden states.
  - Probabilities are formulated using:
    - *Likelihood:* $P(W \mid T)$
    - *Prior:* $P(T)$
  - Maximizes the joint probability $P(W, T)$ for decoding the tag sequence.
- **MEMM (Maximum Entropy Markov Model): Discriminative Model**
  - Directly models the posterior probability $P(T \mid W)$.
  - Discriminates among possible tag sequences given a word sequence.
  - Uses conditional probability, conditioned on the previous tag and current word.
- **Training and Flexibility**
  - **HMM:**
    - Probabilities are obtained by training on a text corpus.

HMM vs. MEMM: Generative vs. Discriminative Models

- **HMM (Hidden Markov Model): Generative Model**
  - Words are modeled as observations generated from hidden states.
  - Probabilities are formulated using:
    - *Likelihood:* $P(W \mid T)$
    - *Prior:* $P(T)$
  - Maximizes the joint probability $P(W, T)$ for decoding the tag sequence.
- **MEMM (Maximum Entropy Markov Model): Discriminative Model**
  - Directly models the posterior probability $P(T \mid W)$.
  - Discriminates among possible tag sequences given a word sequence.
  - Uses conditional probability, conditioned on the previous tag and current word.
- **Training and Flexibility**
  - **HMM:**
    - Probabilities are obtained by training on a text corpus.
  - **MEMM:**
    - Builds a distribution by adding features (e.g., capitalization, hyphens, word endings).
    - Selects the maximum entropy distribution given the feature constraints.
    - More flexible: Allows for diverse, non-independent features.



Source: Adapted from Jurafsky and Martin 2009, fig. 6.20.

Disadvantages of MEMM and the Need for CRF

- **Disadvantages of MEMM:**
  - **Label Bias Problem:**
    - MEMM assigns probabilities based on local decisions at each state.
    - States with fewer outgoing transitions (fewer choices) are often favored, regardless of the overall sequence quality.

Disadvantages of MEMM and the Need for CRF

- **Disadvantages of MEMM:**
  - **Label Bias Problem:**
    - MEMM assigns probabilities based on local decisions at each state.
    - States with fewer outgoing transitions (fewer choices) are often favored, regardless of the overall sequence quality.
  - **Limited Global Context:**
    - MEMM only considers the current word and previous tag for prediction.
    - It does not account for the entire sequence context, which may lead to suboptimal tagging.

3. Conditional Random Fields

- CRF is a type of discriminative probabilistic model used for labeling and segmenting structured data.
- Unlike HMM, which is generative, CRF models the conditional probability of the label sequence given the observation sequence.
- Unlike MEMM, CRF models the entire sequence jointly, avoiding the label bias problem associated with locally normalized models like MEMM.

3. Conditional Random Fields

- CRF is a type of discriminative probabilistic model used for labeling and segmenting structured data.
- Unlike HMM, which is generative, CRF models the conditional probability of the label sequence given the observation sequence.
- Unlike MEMM, CRF models the entire sequence jointly, avoiding the label bias problem associated with locally normalized models like MEMM.
- **Why Use CRF for POS Tagging?**
  - CRF considers the entire sentence when predicting the POS tags, leading to more accurate and context-aware tagging.
  - It overcomes the label bias problem present in models like MEMM.

How CRF Works in POS Tagging

- **Modeling Sequence Data:**
  - CRF models the conditional probability $P(T \mid W)$, where T is the sequence of tags and W is the sequence of words.

How CRF Works in POS Tagging

- **Modeling Sequence Data:**
  - CRF models the conditional probability $P(T \mid W)$, where T is the sequence of tags and W is the sequence of words.
  - The model uses features from the entire sequence to determine the most likely tag sequence.

How CRF Works in POS Tagging

- **Modeling Sequence Data:**
  - CRF models the conditional probability $P(T \mid W)$, where T is the sequence of tags and W is the sequence of words.
  - The model uses features from the entire sequence to determine the most likely tag sequence.
- **Feature Functions:**
  - CRF leverages feature functions to capture relevant patterns in the data, such as:
    - Word identity, prefixes, suffixes
    - Previous tags, capitalization, word shapes
  - These features help in making informed decisions about the POS tags.

Advantages of CRF in POS Tagging

- **Overcoming Label Bias:**
  - CRF does not suffer from the label bias problem because it normalizes across all possible tag sequences, ensuring consistency.

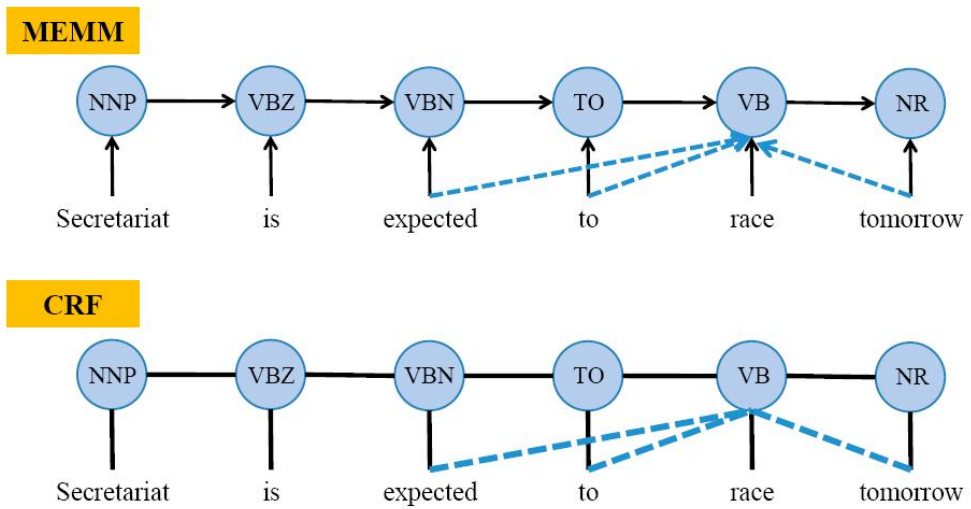
Advantages of CRF in POS Tagging

- **Overcoming Label Bias:**
  - CRF does not suffer from the label bias problem because it normalizes across all possible tag sequences, ensuring consistency.
- **Global Optimization:**
  - CRF optimizes the entire label sequence jointly, considering the full sentence context rather than making local decisions.

Advantages of CRF in POS Tagging

- **Overcoming Label Bias:**
  - CRF does not suffer from the label bias problem because it normalizes across all possible tag sequences, ensuring consistency.
- **Global Optimization:**
  - CRF optimizes the entire label sequence jointly, considering the full sentence context rather than making local decisions.
- **Flexibility:**
  - CRF allows for the inclusion of diverse and non-independent features, enhancing its adaptability to different POS tagging challenges.

MEMM v.s. CRF


MEMM
NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow
CRF
NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow

Conclusion: HMM, MEMM, and CRF

- **HMM (Hidden Markov Model):**
  - **Type:** Generative Model
  - **Probability:** Models joint probability $P(W, T) = P(W \mid T) \times P(T)$
  - **Advantages:** Simple, interpretable, uses well-established algorithms like Viterbi.
  - **Disadvantages:** Limited by assumptions of independence and can struggle with complex features.

Conclusion: HMM, MEMM, and CRF

- **HMM (Hidden Markov Model):**
  - **Type:** Generative Model
  - **Probability:** Models joint probability $P(W, T) = P(W \mid T) \times P(T)$
  - **Advantages:** Simple, interpretable, uses well-established algorithms like Viterbi.
  - **Disadvantages:** Limited by assumptions of independence and can struggle with complex features.
- **MEMM (Maximum Entropy Markov Model):**
  - **Type:** Discriminative Model
  - **Probability:** Models conditional probability $P(T \mid W)$
  - **Advantages:** Incorporates rich, non-independent features, flexible.
  - **Disadvantages:** Suffers from label bias problem due to local normalization.

Conclusion: HMM, MEMM, and CRF

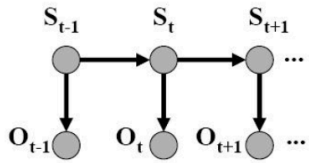
- **HMM (Hidden Markov Model):**
  - **Type:** Generative Model
  - **Probability:** Models joint probability $P(W, T) = P(W \mid T) \times P(T)$
  - **Advantages:** Simple, interpretable, uses well-established algorithms like Viterbi.
  - **Disadvantages:** Limited by assumptions of independence and can struggle with complex features.
- **MEMM (Maximum Entropy Markov Model):**
  - **Type:** Discriminative Model
  - **Probability:** Models conditional probability $P(T \mid W)$
  - **Advantages:** Incorporates rich, non-independent features, flexible.
  - **Disadvantages:** Suffers from label bias problem due to local normalization.
- **CRF (Conditional Random Field):**
  - **Type:** Discriminative Model
  - **Probability:** Models global conditional probability $P(T \mid W)$ across the entire sequence.
  - **Advantages:** Overcomes label bias, handles entire sequences jointly, allows for diverse features.
  - **Disadvantages:** Computationally intensive, complex to implement and train.

Summary

$\vec{s} = s_1, s_2, \ldots s_n \qquad \vec{o} = o_1, o_2, \ldots o_n$

HMM

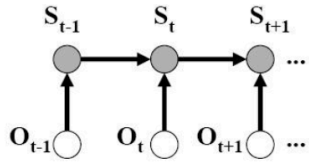
$$P(\vec{s}, \vec{o}) \propto \prod_{t=1}^{|\vec{o}|} P(s_t \mid s_{t-1}) P(o_t \mid s_t)$$



MEMM

$$P(\vec{s} \mid \vec{o}) \propto \prod_{t=1}^{|\vec{o}|} P(s_t \mid s_{t-1}, o_t)$$

$$\propto \prod_{t=1}^{|\vec{o}|} \frac{1}{Z_{s_{t-1}, o_t}} \exp\left(\sum_j \lambda_j f_j(s_t, s_{t-1}) + \sum_k \mu_k g_k(s_t, x_t) \right)$$



CRF

$$P(\vec{s} \mid \vec{o}) \propto \frac{1}{Z_{\vec{o}}} \prod_{t=1}^{|\vec{o}|} \exp\left(\sum_j \lambda_j f_j(s_t, s_{t-1}) + \sum_k \mu_k g_k(s_t, x_t) \right)$$