Viterbi Algorithm

181IT102 - Adarsh Naidu
181IT106 - Aniruddh Patil
181IT123 - Kotla Karthik Reddy
181IT144 - Shivamani Patil

Table Of Contents

Introduction

Project Objective

Methodology

Results and Analysis

Introduction

- The Viterbi algorithm is a dynamic programming algorithm.
- Used for finding the most likely sequence of hidden states-called the Viterbi path- that results in a sequence of observed events, especially in the context Hidden Markov Models.
- Applications include communication for decoding such as in dial-up modems, satellite, deep-space communications and wireless LANs.
- It is now also commonly used in speech recognition, speech synthesis, natural language processing, computational linguistics and bioinformatics.

Hidden Markov Model

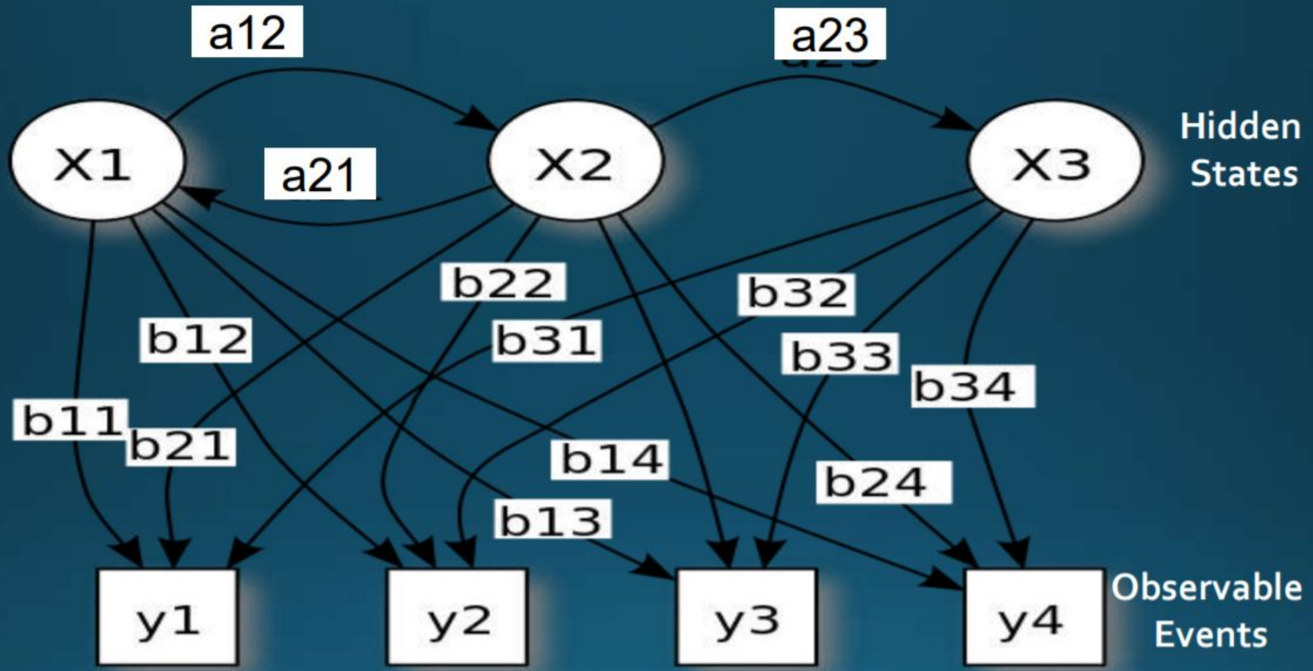
- Markov models are used to model sequences of events (or observations) that occur one after another .
- In a Hidden Markov model, the state is not directly visible, but the output/observations, dependent on the state, is visible.
- Each state has a probability distribution over the possible output .
- The sequence of observations generated by a HMM gives some information about the sequence of states.
- For HMM, one of the important tasks is to find the hidden sequence that is most likely to produce its observation sequence.

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. hidden) states.

A HMM model have the following parameters:

- A set of states : {S1,S2…..Sn}
- A set of observations : {O1,O2….Om}
- Initial probability - the initial probability of each hidden state.
- Transition probability - from one hidden state The probability of state transition to another hidden state.
- Emission probability - the probability that a certain hidden state produces a certain observation phenomenon

An example of Hidden Markov Model (State Diagram)



a_{ij} -> Probability of transition from one state to another

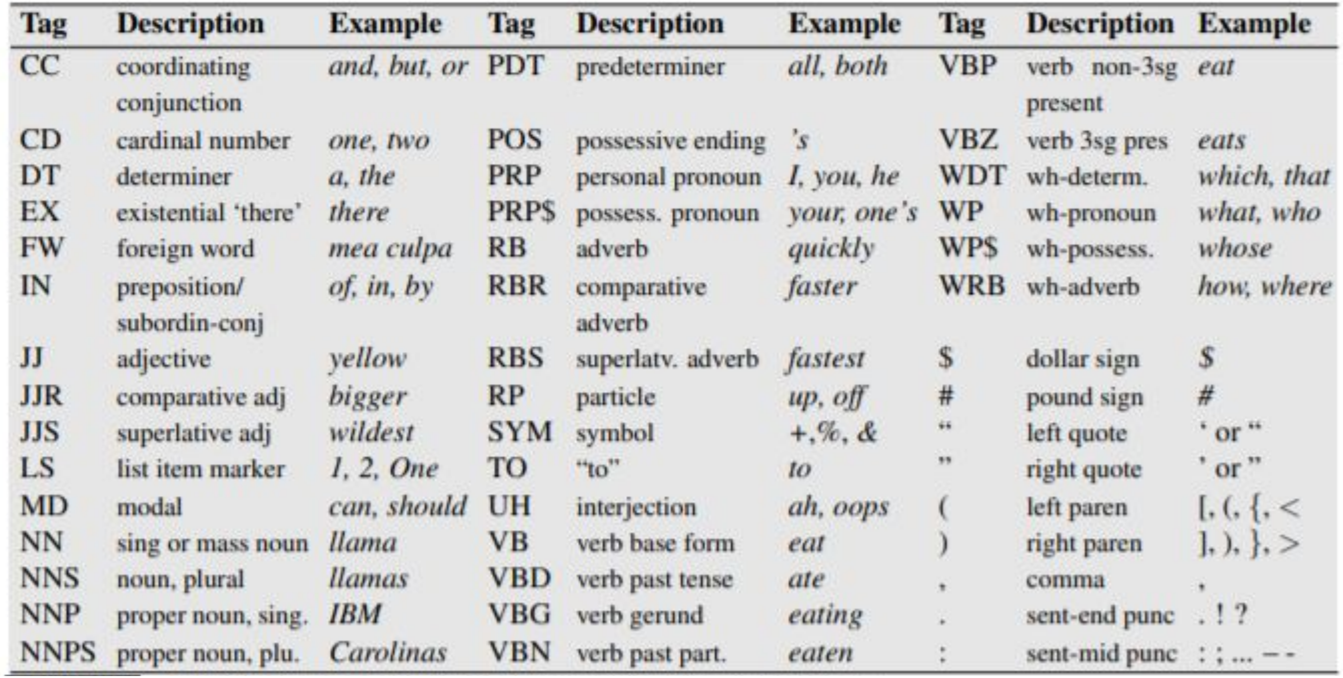
b_{ij} -> Probability of an observation for a state

Part of Speech Tagging

Part-of-speech tagging is the process of assigning a part-of-speech marker to each part-of-speech tagging word in an input text. The input to a tagging algorithm is a sequence of (tokenized) words and a tagset, and the output is a sequence of tags, one per token.

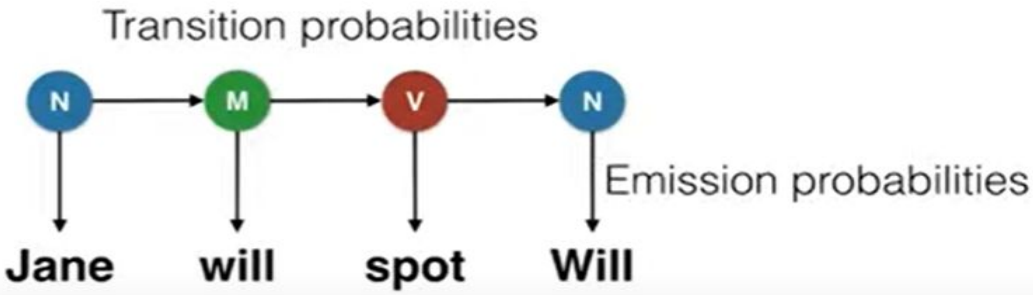
Tagging is a disambiguation task; words are ambiguous—have more than one ambiguous possible part-of-speech—and the goal is to find the correct tag for the situation.

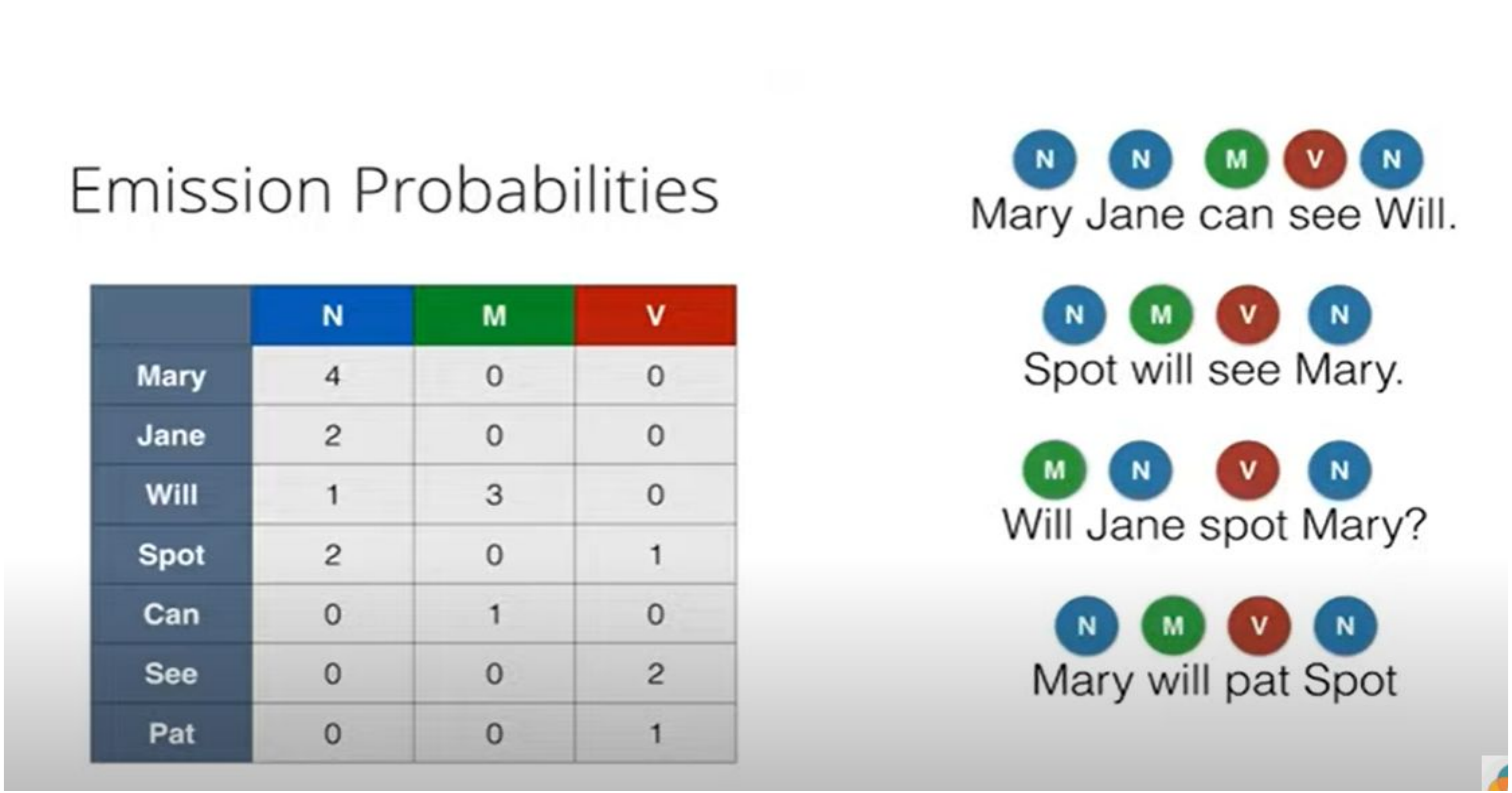
For example, book can be a verb (book that flight) or a noun (hand me that book).That can be a determiner (Does that flight serve dinner) or a complementizer (I thought that your flight was earlier). The goal of POS-tagging is to resolve these ambiguities, choosing the proper tag for the context.



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	"	left quote	' or "
LS	list item marker	*1, 2, One*	TO	"to"	*to*	"	right quote	' or "
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Part of Speech Tagging





Emission Probabilities

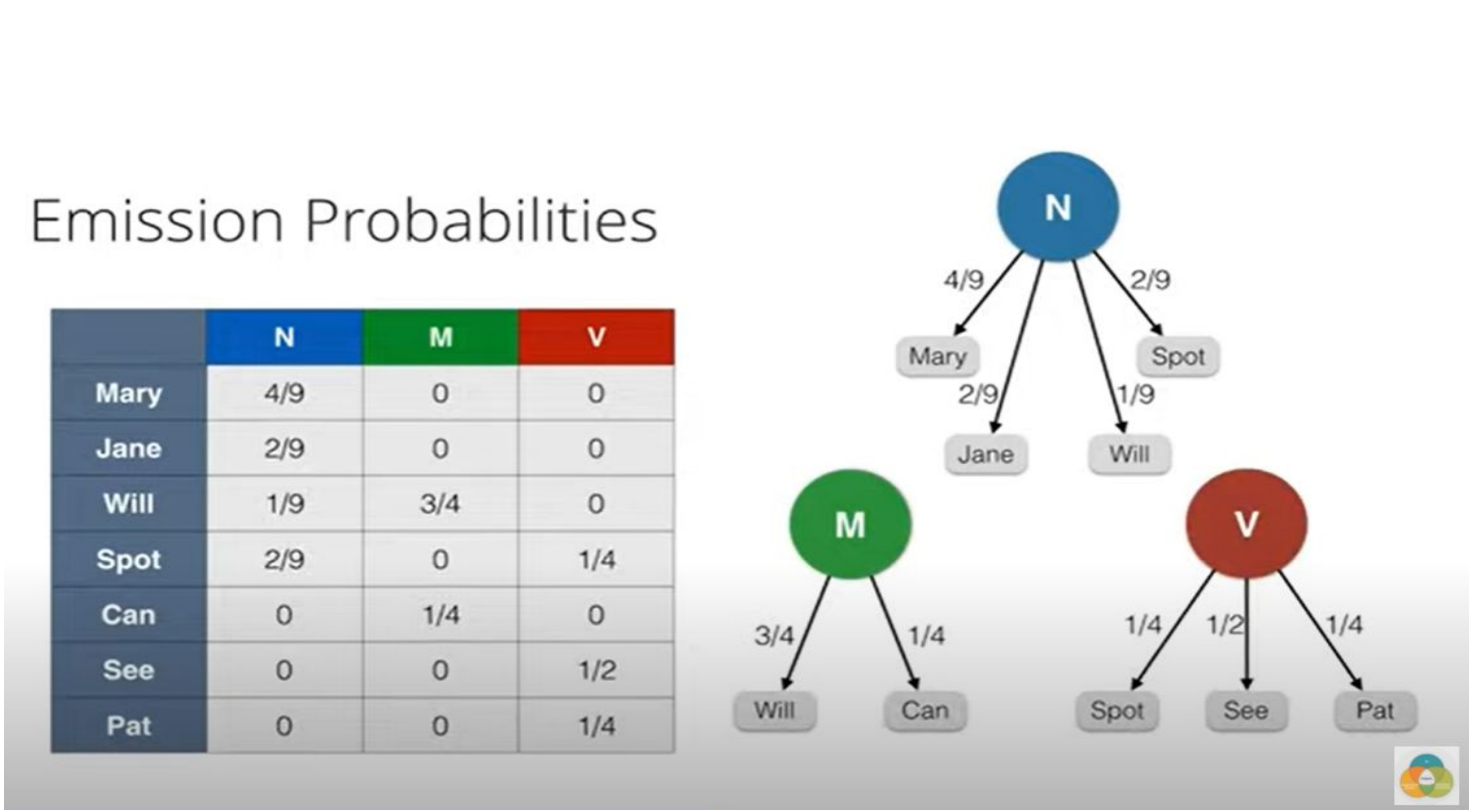
	N	M	V
Mary	4	0	0
Jane	2	0	0
Will	1	3	0
Spot	2	0	1
Can	0	1	0
See	0	0	2
Pat	0	0	1

N N M V N
Mary Jane can see Will.

N M V N
Spot will see Mary.

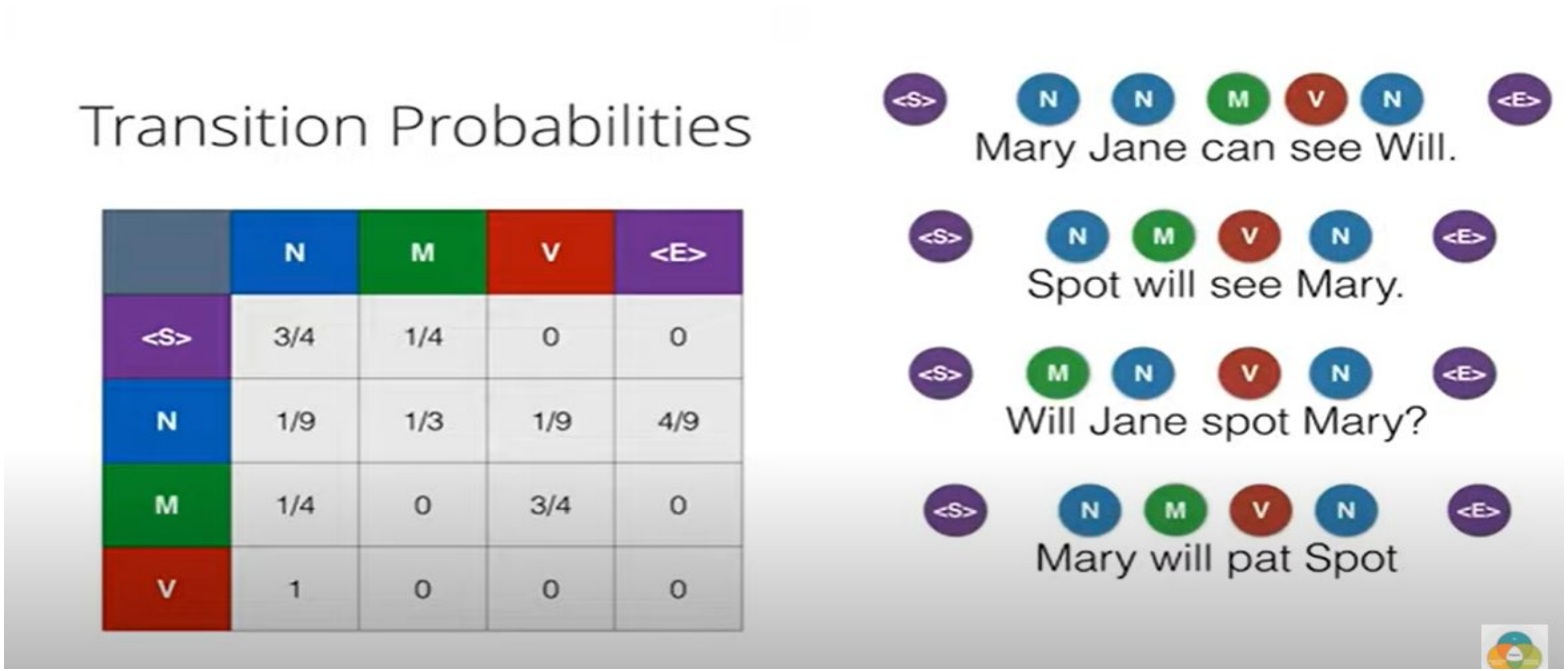
M N V N
Will Jane spot Mary?

N M V N
Mary will pat Spot



Emission Probabilities

	N	M	V
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	1/2
Pat	0	0	1/4



Transition Probabilities

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0

<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>

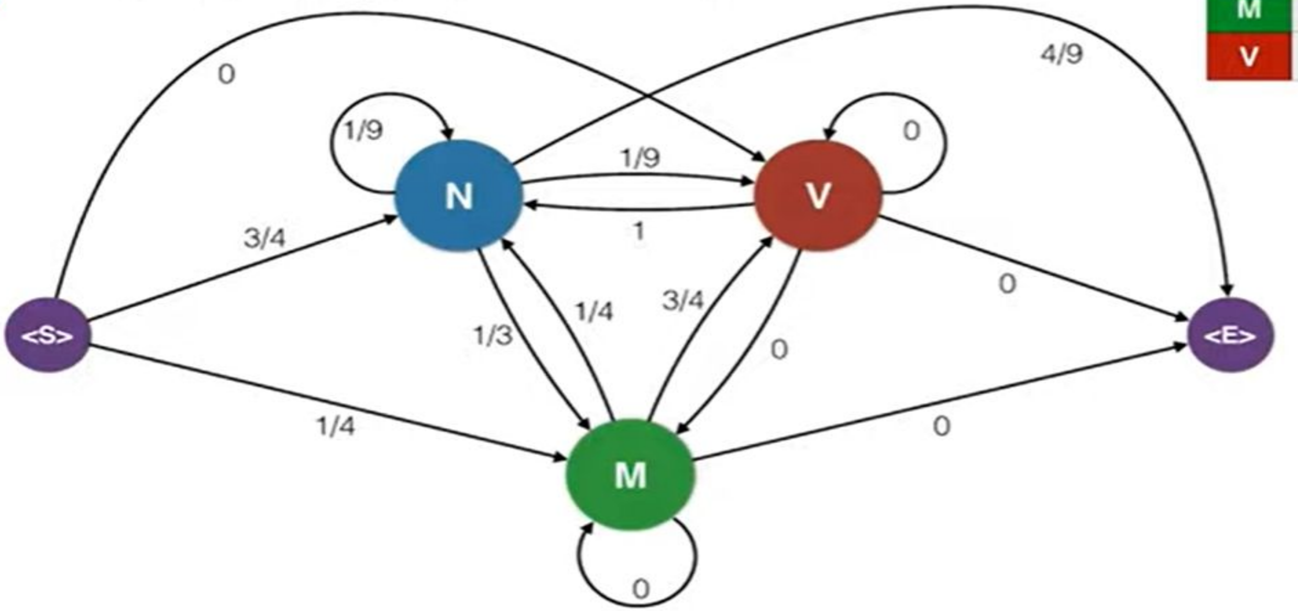
Will Jane spot Mary?

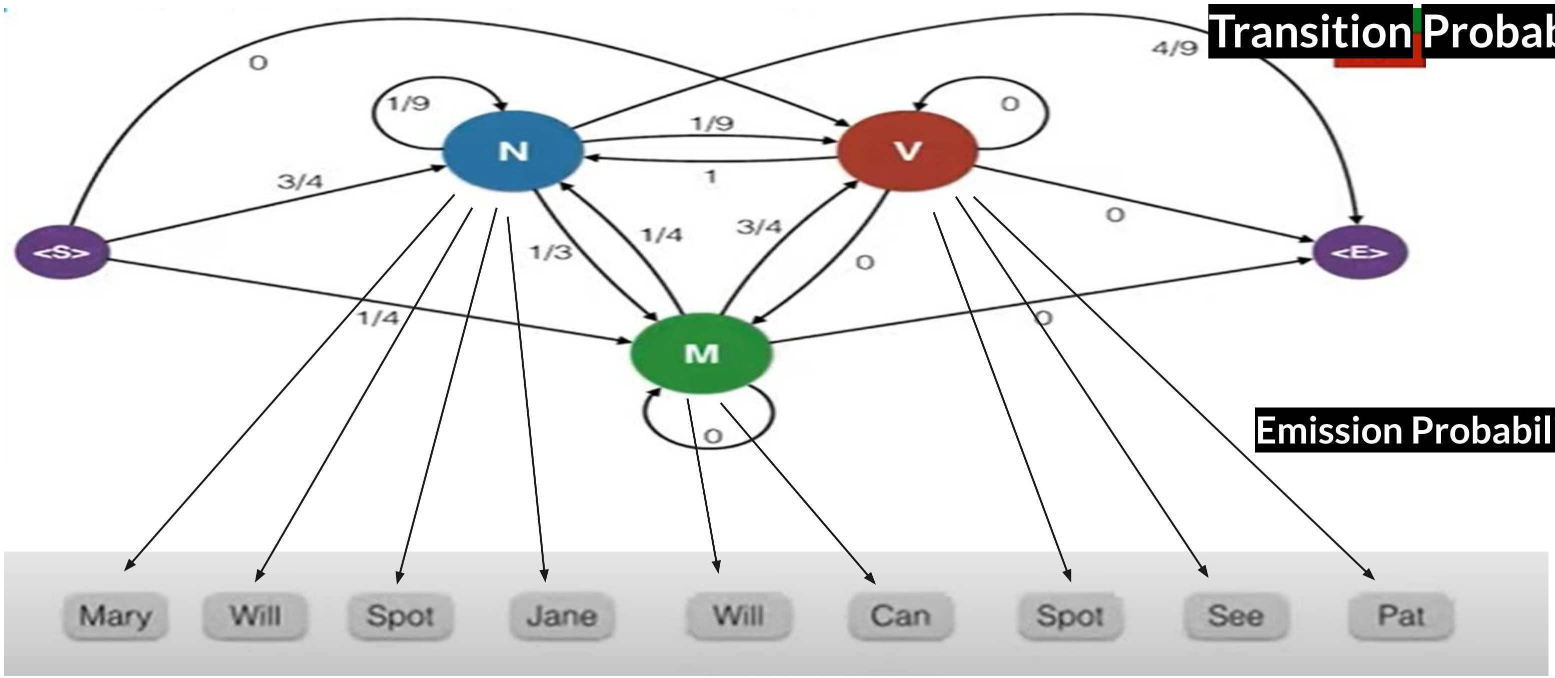
<S> N M V N <E>

Mary will pat Spot

Transition Probabilities

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0



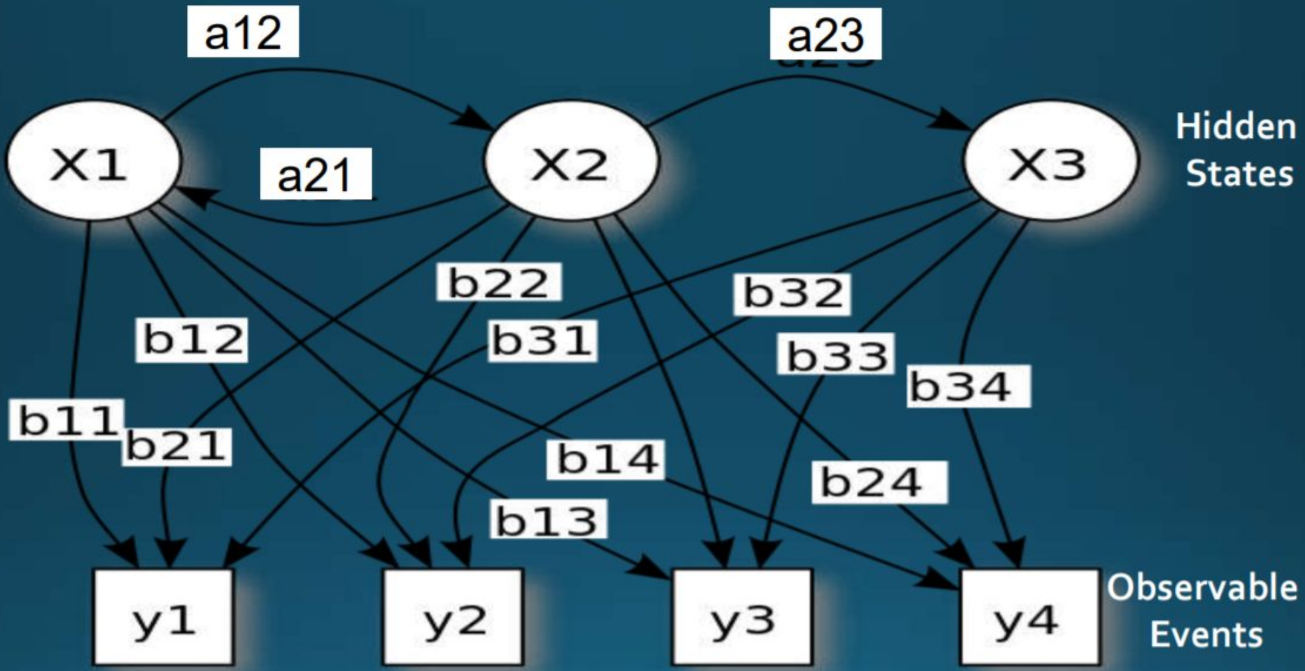


Transition Probabilities

Emission Probabilities

Hidden Markov Model

An example of Hidden Markov Model (State Diagram)



a_{ij} -> Probability of transition from one state to another

b_{ij} -> Probability of an observation for a state

Viterbi Algorithm

The Viterbi Algorithm is used for finding the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models.

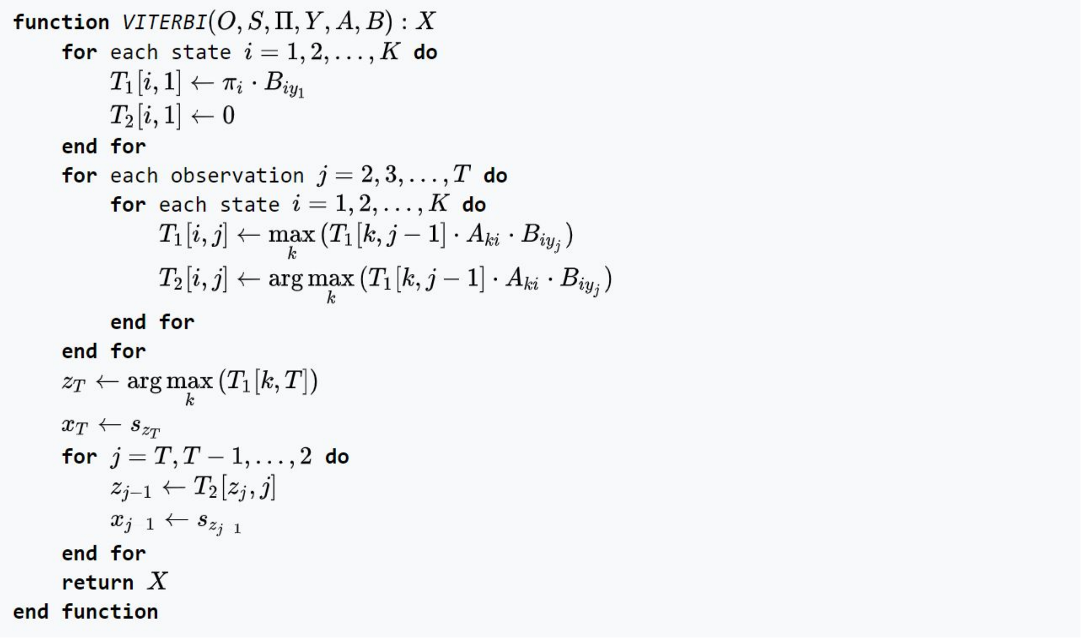
The decoding algorithm for HMMs is the Viterbi algorithm. As Viterbi algorithm is an instance of dynamic programming, Viterbi resembles minimum edit distance algorithm.

Viterbi Algorithm

The Viterbi algorithm first sets up a probability matrix or lattice, with one column for each observation **ot** and one row for each state in the state graph. Each column thus has a cell for each state **qi** in the single combined automaton.

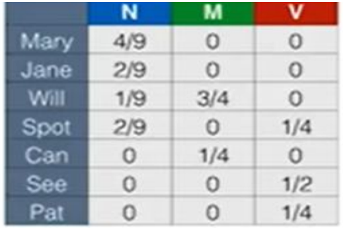
Each cell of the lattice,vt(j), represents the probability that the HMM is in statejafter seeing the first t observations and passing through the most probable state sequence q1,...,qt−1, given the HMM

Pseudocode

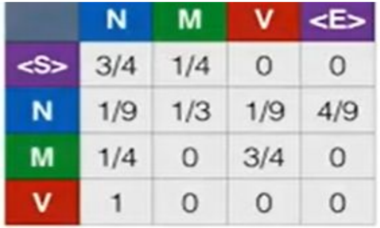


```
function VITERBI$(O, S, \Pi, Y, A, B) : X$
    for each state $i = 1, 2, \ldots, K$ do
        $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
        $T_2[i, 1] \leftarrow 0$
    end for
    for each observation $j = 2, 3, \ldots, T$ do
        for each state $i = 1, 2, \ldots, K$ do
            $T_1[i, j] \leftarrow \max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
            $T_2[i, j] \leftarrow \arg\max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $x_T \leftarrow s_{z_T}$
    for $j = T, T-1, \ldots, 2$ do
        $z_{j-1} \leftarrow T_2[z_j, j]$
        $x_{j\ 1} \leftarrow s_{z_{j\ 1}}$
    end for
    return $X$
end function
```

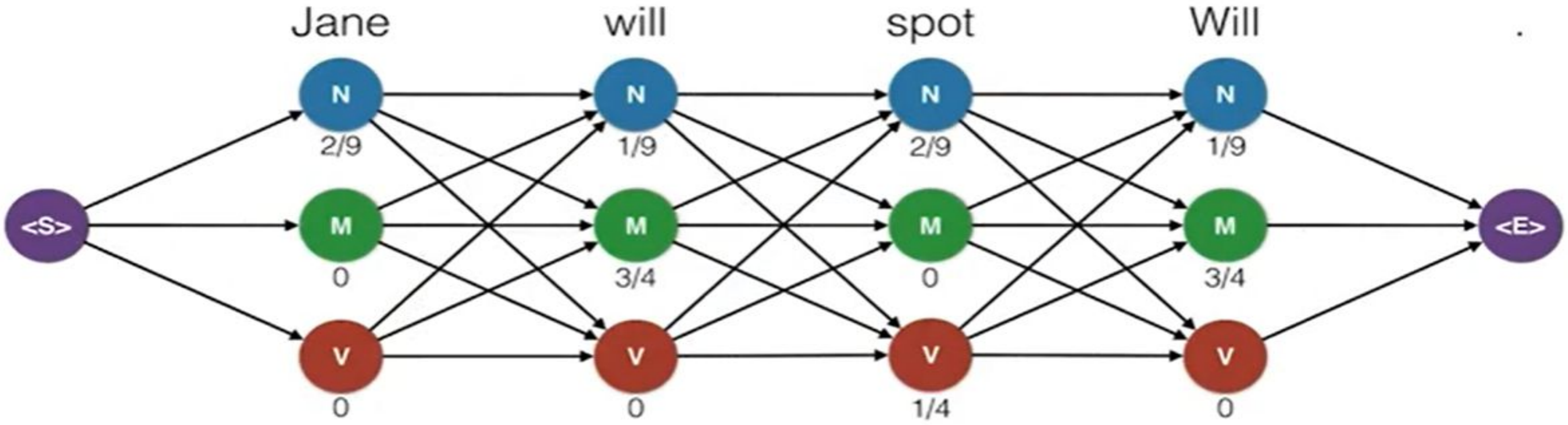
Viterbi Algorithm

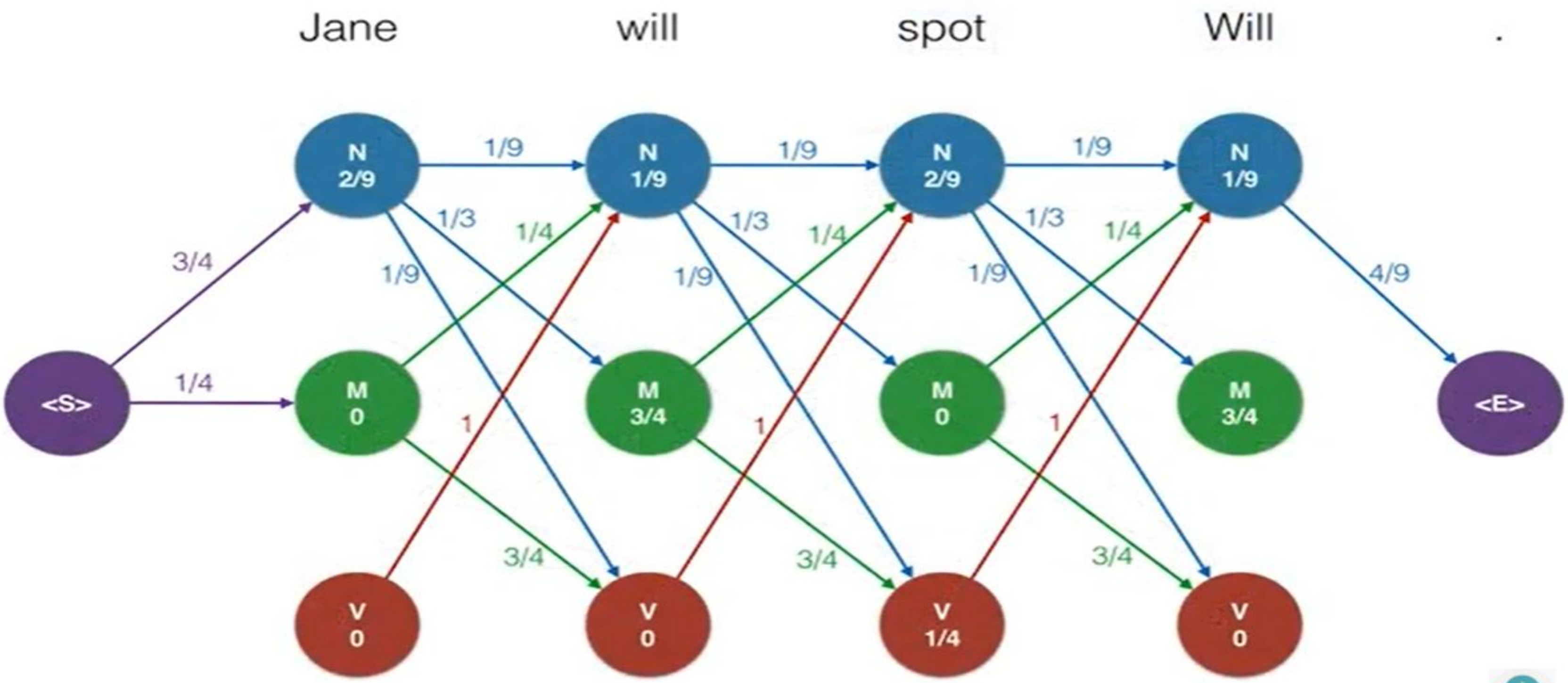


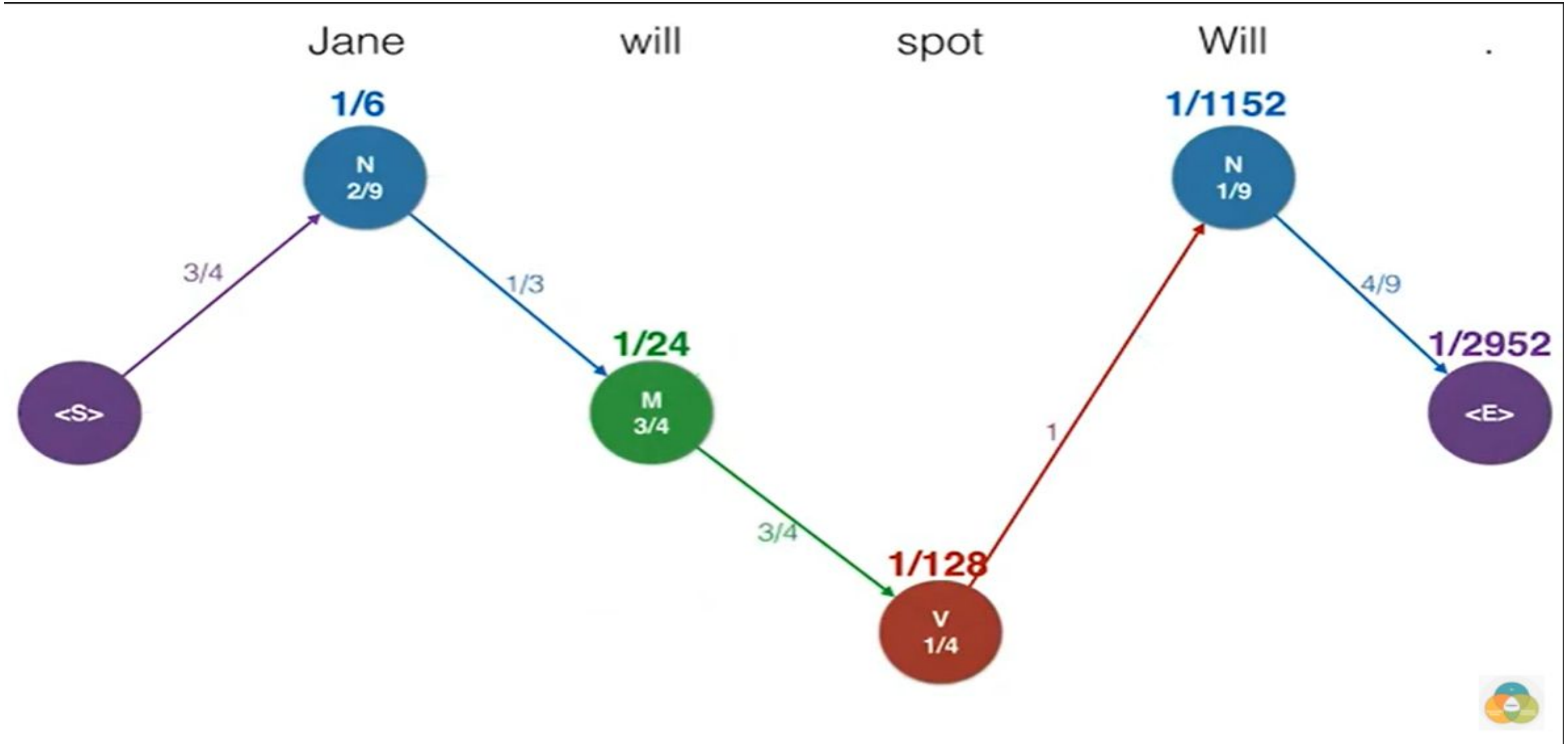
	N	M	V
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	1/2
Pat	0	0	1/4



	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0




Jane
will
spot
Will
.
<S>
<E>
3/4
1/4
N 2/9
M 0
V 0
N 1/9
M 3/4
V 0
N 2/9
M 0
V 1/4
N 1/9
M 3/4
V 0
1/9
1/9
1/9
1/3
1/3
1/3
1/4
1/4
1/4
1/9
1/9
1/9
1
1
1
3/4
3/4
3/4
4/9


Jane
will
spot
Will
.
1/6
N
2/9
1/1152
N
1/9
3/4
1/3
4/9
1/24
M
3/4
1/2952
<S>
<E>
1
3/4
1/128
V
1/4

Project Objectives

The goal of this project is to implement and train a part-of-speech (POS) tagger, as described in "Speech and Language Processing" (Jurafsky and Martin).

A hidden Markov model is implemented to estimate the transition and emission probabilities from the training data. The Viterbi algorithm is used for decoding, i.e. finding the most likely sequence of hidden states (POS tags) for previously unseen observations (sentences).

Methodology

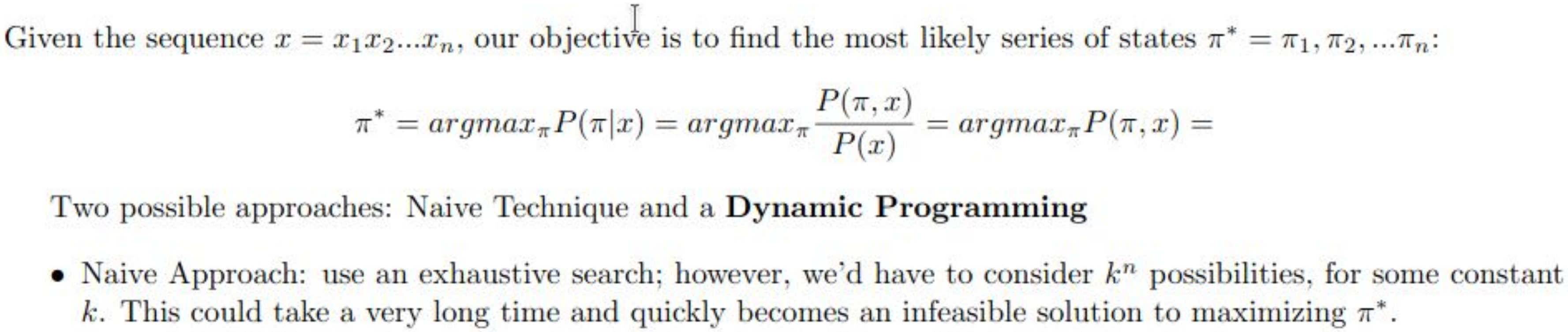
The HMM is trained on bigram distributions (distributions of pairs of adjacent tokens). The first pass over the training data generates a fixed list of vocabulary tokens. Any token occurring less than twice in the training data is assigned a special unknown word token based on a few selected morphological idiosyncrasies of common English word classes (e.g. most tokens with the suffix "-ism" are nouns). The second pass uses the transformed training data to collect the bigram transition and emission counts and saves them to a model file.

Methodology

To decode the input sequence it is first transformed according to the unknown word rules. The transition and emission counts are then converted to proper probability distributions, using additive smoothing to estimate probabilities for transitions/emissions that have not been observed in the training data. A pseudo count alpha > 0 is used as the smoothing parameter, with alpha = 0.001 giving the best results .

For training and decoding, the input sequences are treated as a continuous sequence of tokens. Sentence boundaries are marked by introducing an artificial "start-of-sentence" state ("--s--") occurring with "newline" tokens ("--n--"). It takes about 60 seconds to train the model and decode the development split.

Results And Analysis



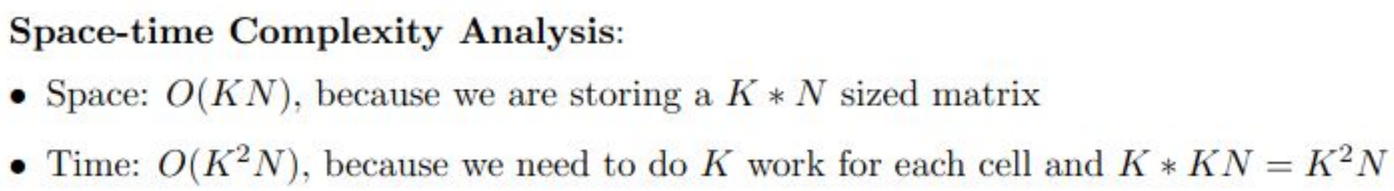
Given the sequence $x = x_1 x_2 ... x_n$, our objective is to find the most likely series of states $\pi^* = \pi_1, \pi_2, ...\pi_n$:

$$\pi^* = argmax_\pi P(\pi|x) = argmax_\pi \frac{P(\pi, x)}{P(x)} = argmax_\pi P(\pi, x) =$$

Two possible approaches: Naive Technique and a **Dynamic Programming**

- Naive Approach: use an exhaustive search; however, we'd have to consider k^n possibilities, for some constant k. This could take a very long time and quickly becomes an infeasible solution to maximizing π^*.

Results And Analysis



Space-time Complexity Analysis:

- Space: $O(KN)$, because we are storing a $K * N$ sized matrix
- Time: $O(K^2N)$, because we need to do K work for each cell and $K * KN = K^2N$

Project Demo

Thank You