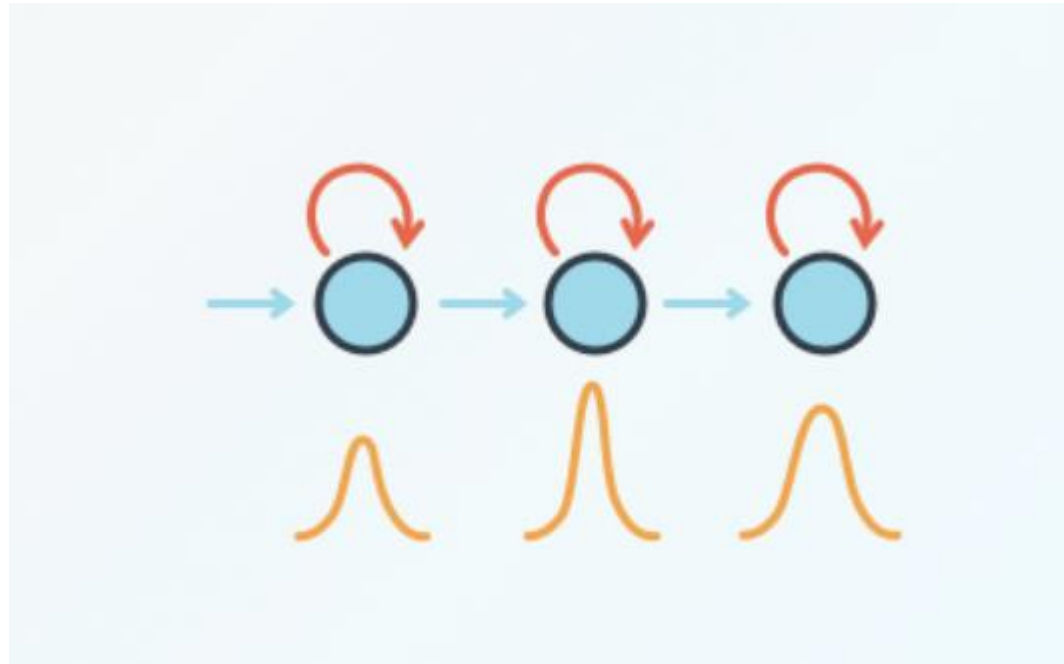


Part of Speech Tagging Example






POS Tagging using HMM

In this example will work on three tags:

- Noun
- Modal Verb: which is normally used with another verb
- Verb

For example:

-  Noun: house, car, Mary, Bob, etc.
-  Modal verb: can, would, should, may, etc.
-  Verb: run, see, jump, eat, walk, etc.

Look up table

1) Look up table.

Let data is in form of some sentences.
and Tag a sentence.

Data :-

Mary saw will

Mary (N)	saw (V)	Jane (N)
Jane (N)	saw (V)	will (N)

→ In look up table, each word is tagged with most common Part of Speech.

→ In given data, Mary, Jane and. will are tagged as nouns and saw is tagged as a verb.

→ In look up table, rows are words and columns are parts of speech and will write down how often each one appears in the existing data.

	N	V
Mary	1	0
saw	0	2
Jane	2	0
will	1	0

ii) Bigrams

→ In this data, sentences are bit more complicated.

Data

Mary	Will	see	Jane
(N)	(M)	(V)	(M)
Will	Will	see	Mary
(N)	(M)	(V)	(N)
Jane	Will	see	Will.
(N)	(M)	(V)	(N)

and. goal is to tag a sentence.

Mary will see will
(N) (M) (V) (M)

→ Using Normal look up table, Mary gets correctly tagged as Noun. Then will get correctly tagged as a Modal. See as a verb and will always get tagged as a Modal since it appears three times as modal and two times as a Noun, but in this sentence, we know will is a Noun since it is referring to our friend will.

⇒ This is a Problem, look up tables won't work very well if a word can have a two different tags, since it will always pick the most common tags that's associated with no matter the context.

Look up table

	<u>N</u>	<u>V</u>	<u>M</u>
Mary	2	0	0
See	0	3	0
Jane	2	0	0
Will	2	0	3

→ The simplest way to consider the context is to look at each word's neighbor

→ For e.g. suppose we tagged a consecutive pair, see-Jane as a Verb-Noun one time, as it appears once as a Verb-Noun and now onto the other sentence we will try the first word using the previous table.

→ So let's tag Mary as a Noun, Now
for each following word, we will tag
using the previous one and the look up
table to find the pair of tags that
correspond to them.

→ So for example to tag a word will,
we look at Mary - will in Particlar
where Mary is a Noun and we see
that the most common one is when will
is a Modal. So we'll tag will as a Modal.
We continue tagging see as a Verb, and the
second will correctly as a Noun, since
the pair see - will is tagged as a Verb-Noun.

So Bigrams for Given Data:

	<u>N-M</u>	<u>M-V</u>	<u>V-N</u>
<u>Mary-Will</u>	1	0	0
<u>Will-See</u>	0	3	0
<u>See-Jane</u>	0	0	1
<u>Will-Will</u>	1	0	0
<u>See-Mary</u>	0	0	1
<u>Jane-Will</u>	1	0	0
<u>See-Will</u>	0	0	1

* When bigrams won't work

→ consider the more complicated data.

Data

Mary Jane can see will

Spot will see Mary

will Jane spot Mary?

Mary will put spot

and we want to tag a sentence

Jane will spot will

→ The Problem is here, the first pair
"Jane will", ~~it~~ is not in our data.

This pair of words never appears, but
somehow it's pair that makes sense.

N-M M-V V-N -----

Mary - Jane

Jane - can

Can - see

See - will

Spot - will

Will - see

See - Mary

Will - Jane

Jane - Spot

→ So a Problem with bigram or n-gram is that.
Some times every pair or every n-gram does not
appear and we have no way to fill the extra
information.

iii) Hidden Markov Models

The idea of HMM is the following -

→ Let's say that a way of tagging

the sentence - "Jane will spot will" is a

noun - modal - verb - noun and will calculate

a ~~probab~~ probability associated with this

tagging.

→ Two things are required here -

→ how likely is it that a noun is

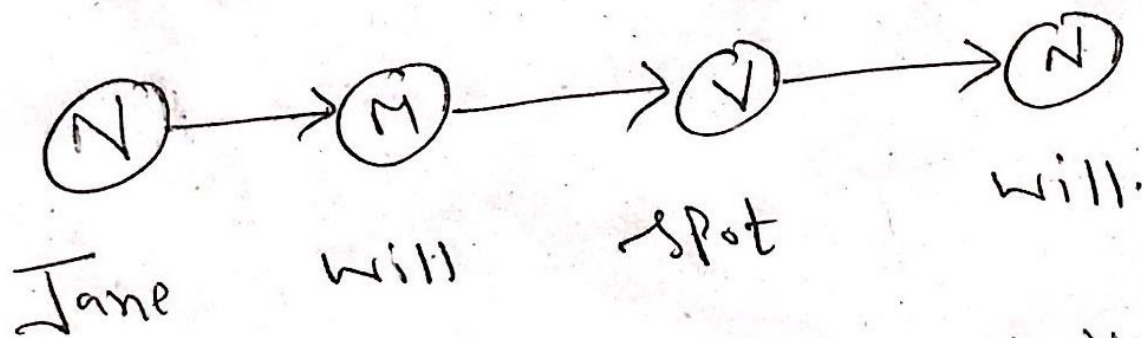
followed by modal and a modal by a verb

by a noun. These need to be high in order

for this tagging to be likely. These are

called transition probabilities.

Transition Probabilities.



→ The second set of probabilities we need to calculate are these :- what is the probability that a noun will be one word Jane and that a modal will be one word will, etc. This is also need to be relatively high for our tagging to be likely. These are called Emission Probabilities

Calculating emission and Transition probabilities

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

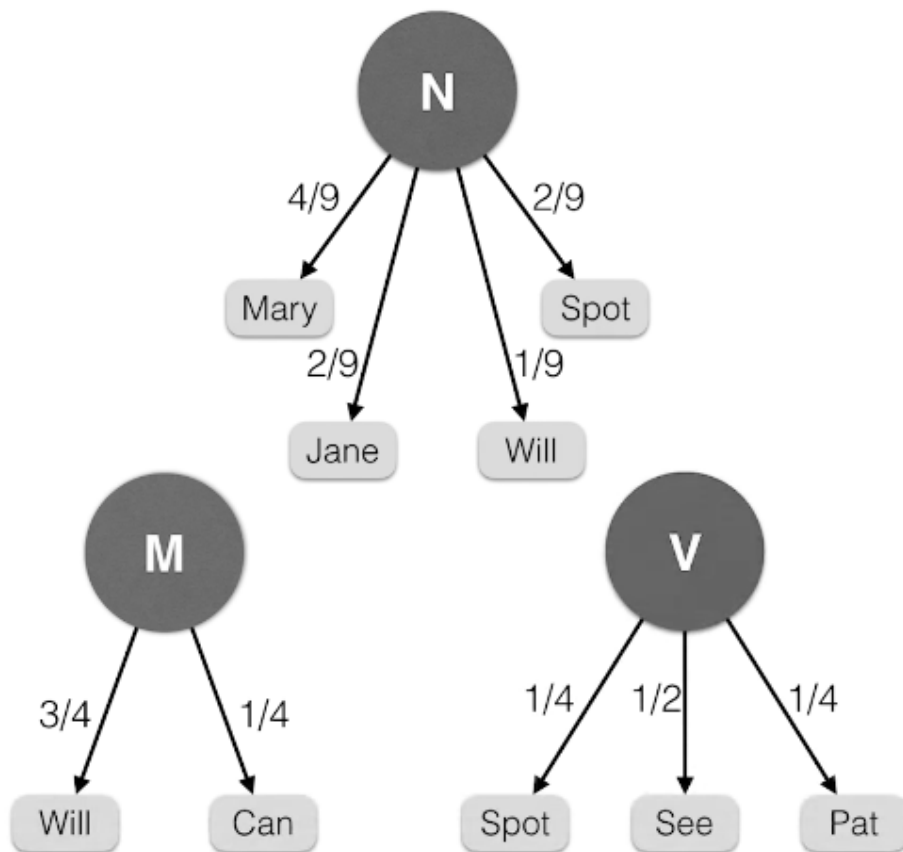
So, here are sentences with their corresponding tags and we're going to calculate the Emission Probabilities, that is the probability that if a word is, say, a noun that that word will be Mary or Jane, etc.

So, to do this we again do a counting table, for example, the entry on the Mary row and the noun column is four because Mary appears four times as a noun.

Now, in order to find the probabilities, we divide each column by the sum of the entries and we obtain the following numbers.

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	$1/4$



So, now let's calculate the transition probabilities. These are the probabilities that are part of speech follows another part of speech.

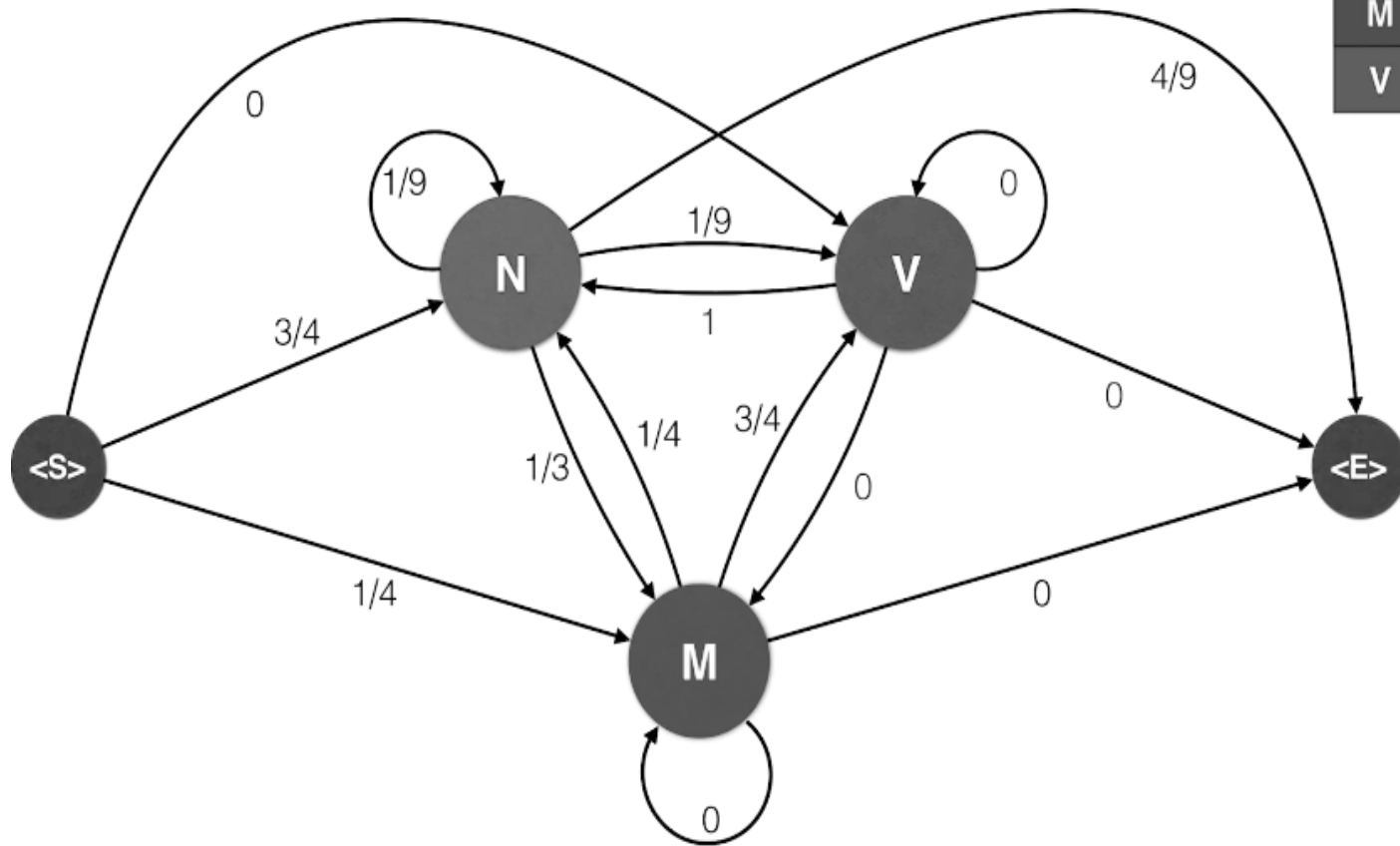
we'll add starting and ending tags on each sentence and we'll treat these tags as parts of speech as well.

And now we make a table of counts in this table. We count the number of appearances of each pair of parts of speech.

For example, this three here in the noun row and the modal column corresponds to the three occurrences of a noun followed by a modal.

Now, to find the probabilities, we divide each row by the sum of the entries in the row. And here's a nice graph of our transition probabilities.

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

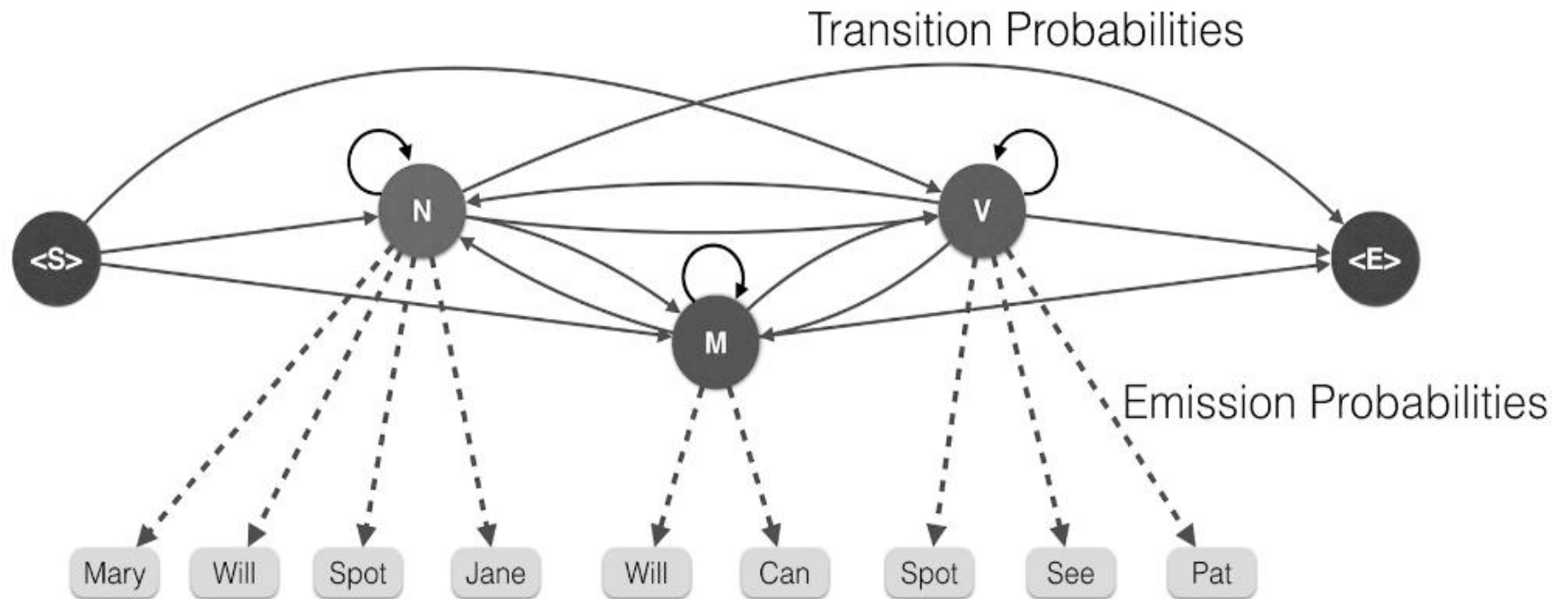
As a final step we combine the two previous graphs to form a Hidden Markov Model.

We have our words, which are observations. These are called the observations because they are the things we observe when we read the sentences.

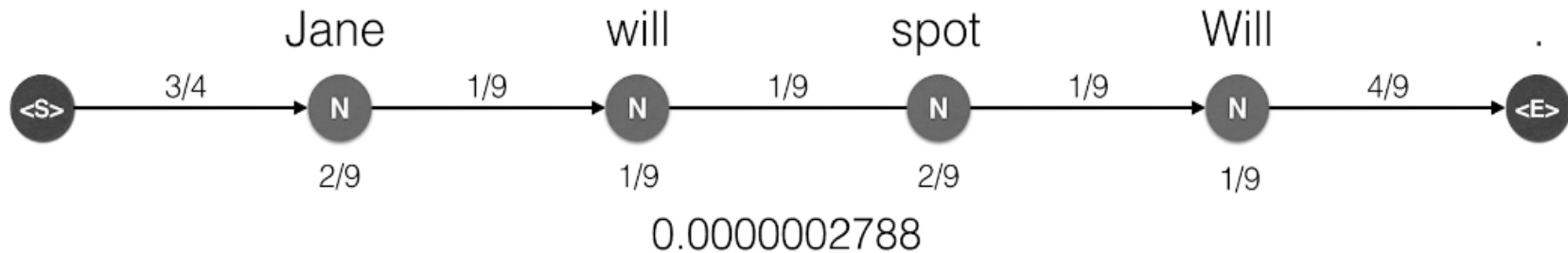
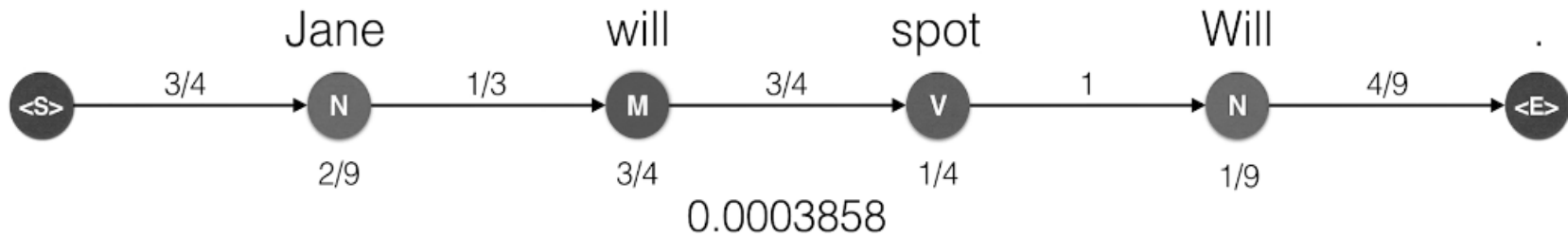
And the parts of speech are called the Hidden States, since they are the ones we don't know and we have to infer based on the words.

And among the Hidden States we have the transmission probabilities, and between the Hidden States and the observation we have the Emission Probabilities. **That's a Hidden Markov Model.**

Hidden Markov Model



How many paths?



The transition probability from the start state to N is three-quarters. So we'll record this three-quarters.

Now, at N we generate the word Jane. This happens with emission probability two-ninths, so we record that.

Now, we move to M and the transition probability is one-third and generate Will with emission probability of three-quarters.

Then, we move to V with transition probability three-quarters and generate spot with the emission probability one-quarter.

Then, we move back to N with transition probability one and we generate Will with emission probability one over nine.

Finally, we move to the end state with transition probability four over nine. So, since the transition moves are all independent of each other and the emission moves are conditional on the hidden state that were located at, the probability that this Hidden Markov Model generates a sentence is the product of these probabilities. We multiply them and get the total probability of the sentence being emitted is 0.0003858.

It looks small, but actually it's large considering the huge amount of sentences we can generate of many lengths.

So let's repeat this path for, Jane will spot Will. We have the parts of speech noun, modal, verb, noun and the probabilities, and their product is 0.0003858.

Let's see if another path can generate this sentence. What about the path **noun noun noun noun**?

We hope this is smaller since it's a bit of a nonsensical pattern of parts of speech. And indeed, it is very small. It is the product of these numbers which is 0.0000002788.

So, among these two, we pick the one on top because it generates the sentence with a higher probability. And in general, what we'll do is from all the possible combinations of parts of speech,

we'll pick the one that generates the sentence, Jane will spot Will, with the highest probability. This is called maximum likelihood.

All we have to do is go over all the possible chains of parts of speech that could generate the sentence.

How many of these chains are there?

Jane will spot Will.



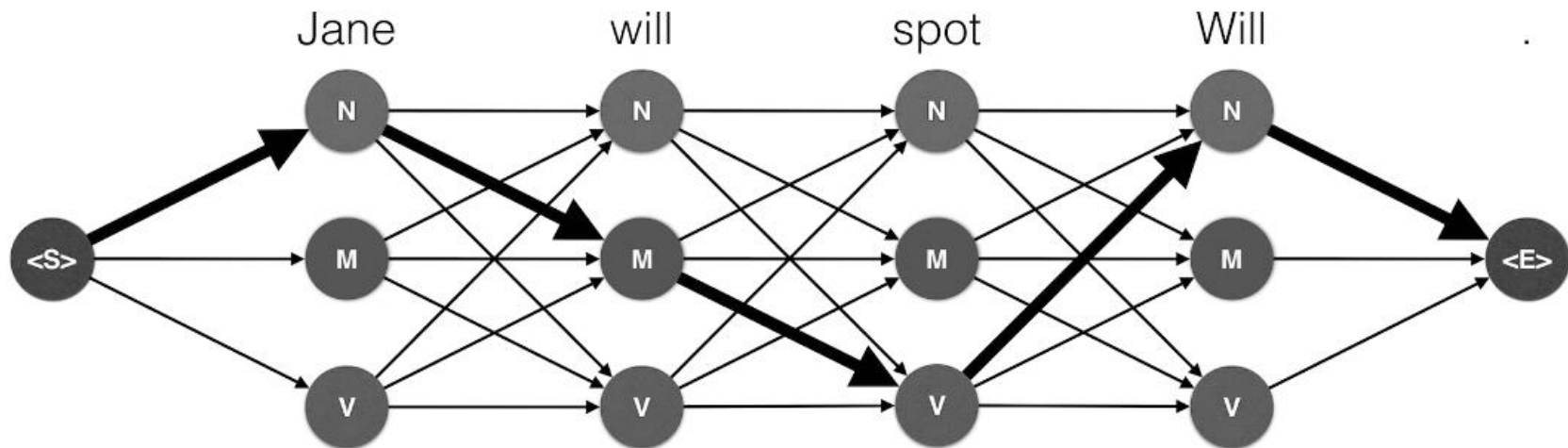
81 Possibilities:

And the reason is that we have three possibilities for Jane, three for will, three for spot, and three for Will. Here are the 81 possibilities. So this is not so bad.

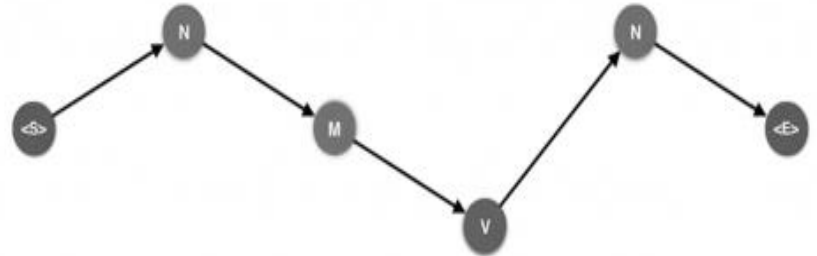
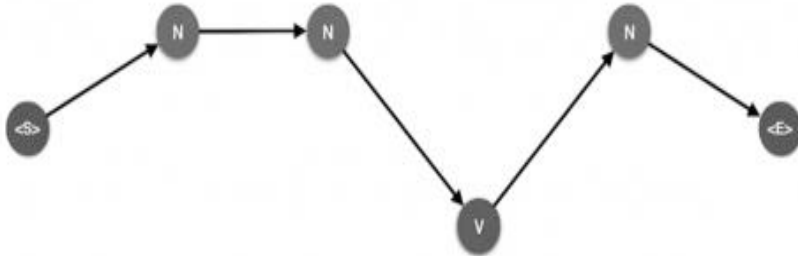
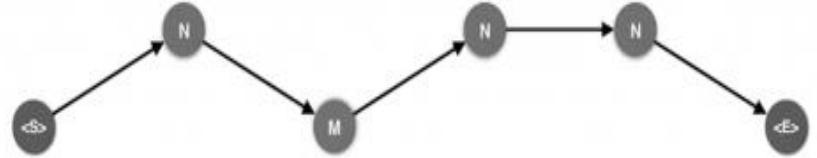
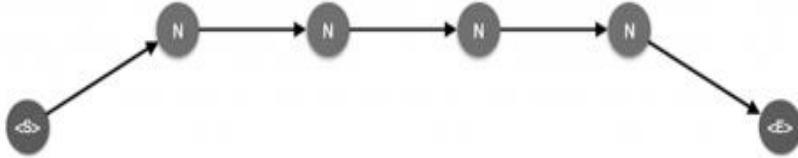
We could check 81 products of four things and find the largest one. But what if we have say 30 parts of speech and we need to take a sentence of 10 words. Then we have 30 to the 10 possibilities.

That starts being pretty large. In general, it's exponential

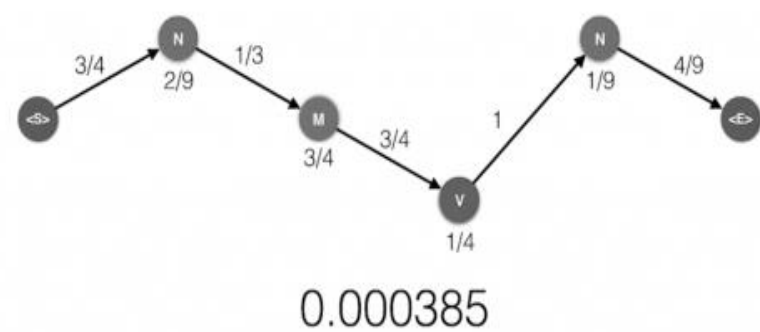
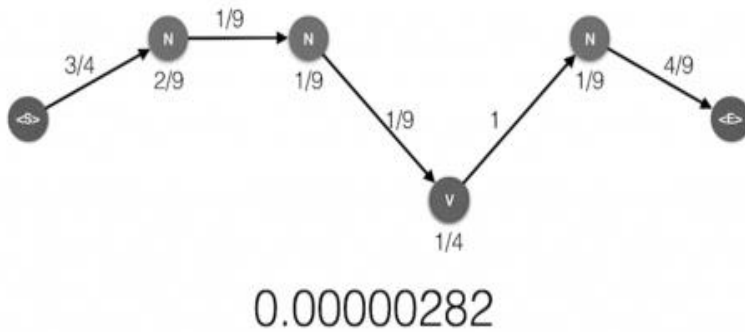
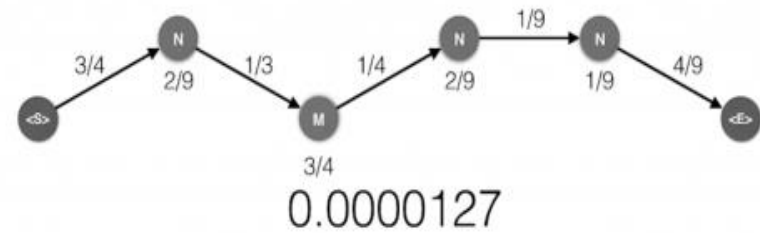
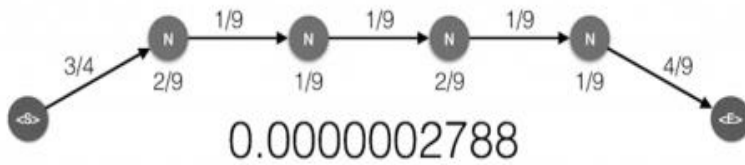
Hidden Markov Model



Which path is more likely?

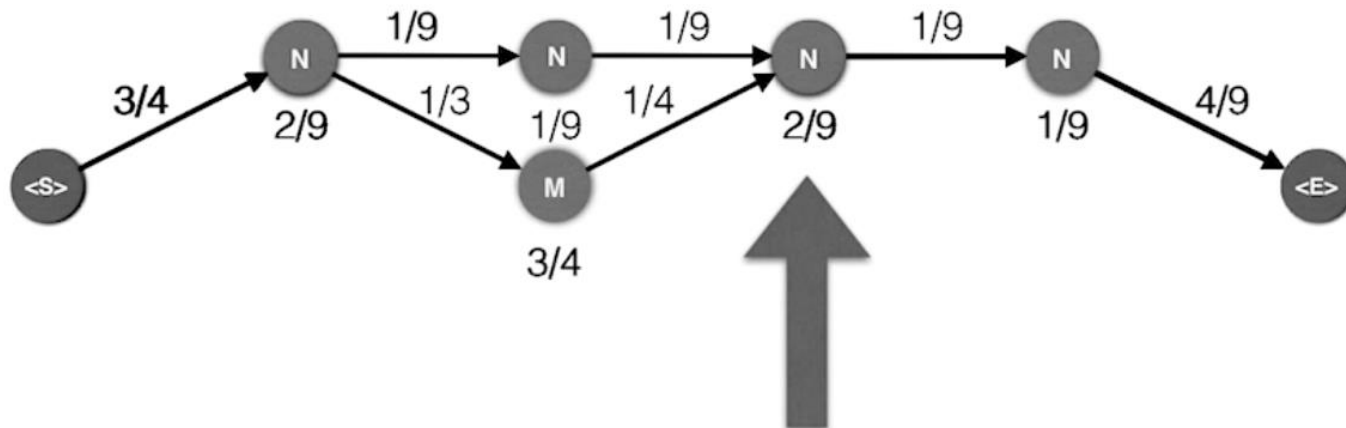
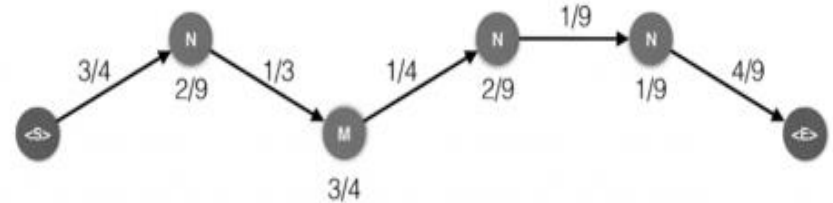
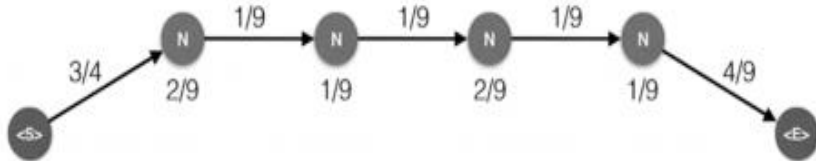


Solution: N-M-V-N



Viterbi Algorithm Idea

Let's look at only these two paths and let's merge them



Here are the two ways, And the probabilities for the mini path in the left is much smaller than that of the mini path on the right.

So, we're really just concerned with the mini path on the right, and we can forget about the other one.

So for every possible state, we will record the highest probability path that ends in that state.

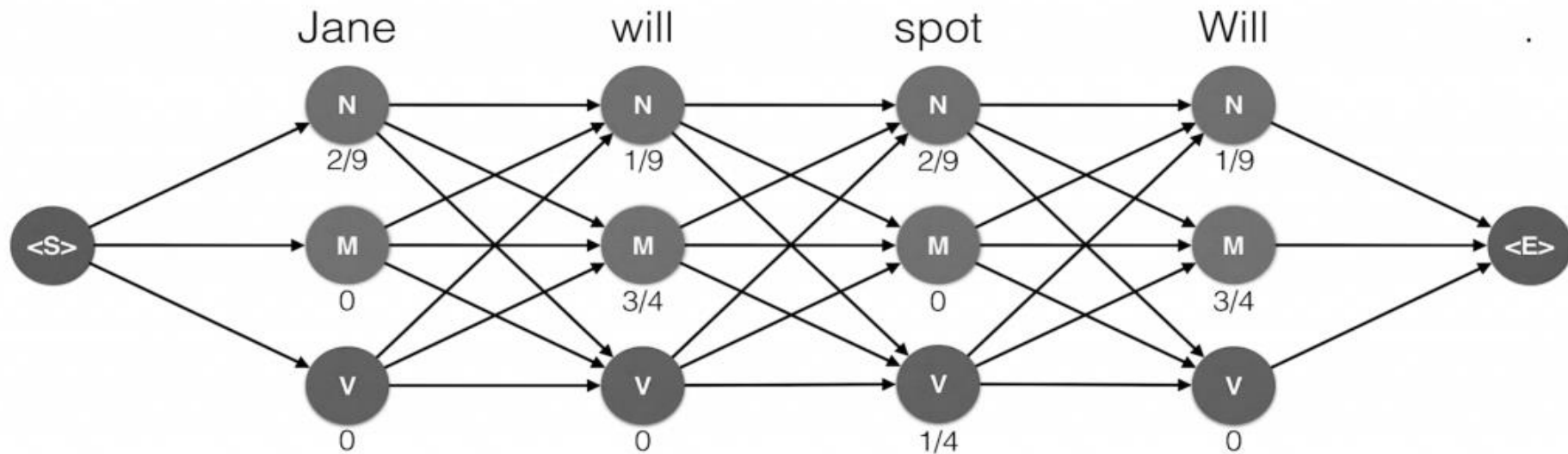
This known as the Viterbi algorithm. It is a dynamic programming approach.

So here's our trellised diagram with the two probability tables, emission and transition.

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



So let's start from the left. The probability that the word Jane is generated by the path that ends at the top end node is the product of the probability of getting to that node from the start node, which is three quarters, times the probability that the node generated the word Jane which is two over nine.

This product is one sixth that was recorded. Likewise, the probability of the M node is one-quarter times zero which is zero and the one on the V node is zero times zero which is zero. Now, let's go to the next word will.

There are three paths that end in the top right N node. One coming from the previous N, one from the previous M and one from the previous V. We need to look at the three and pick the one with the largest probability.

The probability for the path coming from the N node is one-sixth, which is a probability so far until the word Jane, times the transition probability one over nine to get to the new N node, times the emission probability one over nine for emitting the word will.

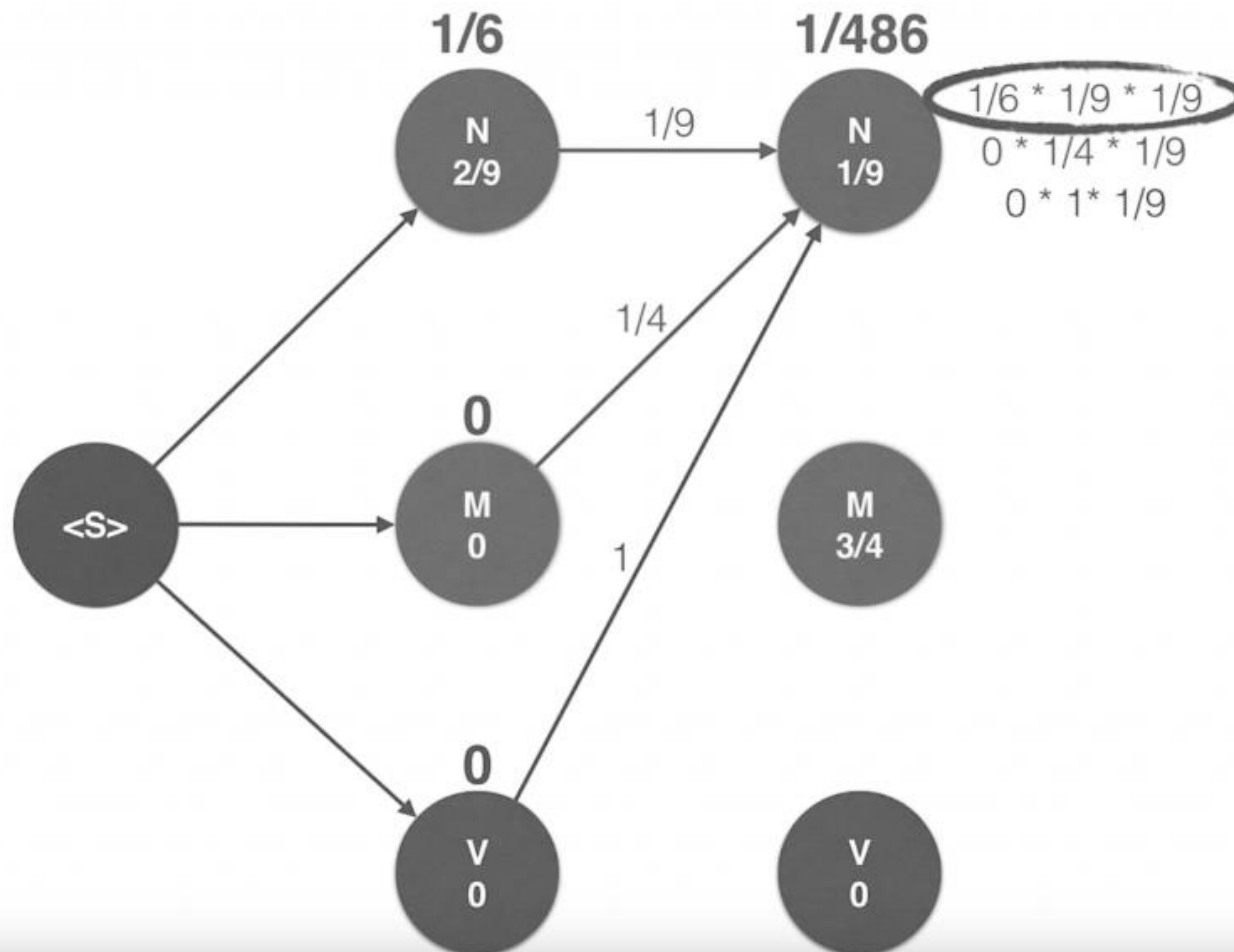
We'll record this product as one-sixth times one-ninth times one-ninth. Likewise, for the path that comes from the M, the probability is zero times one-quarter times one-ninth. And for the path that comes from the V, it's zero times one times one-ninth. Since two of these are zero, the largest one is the first one which is one over 486. If two edges give us the same value, We can break ties by picking one of them randomly.

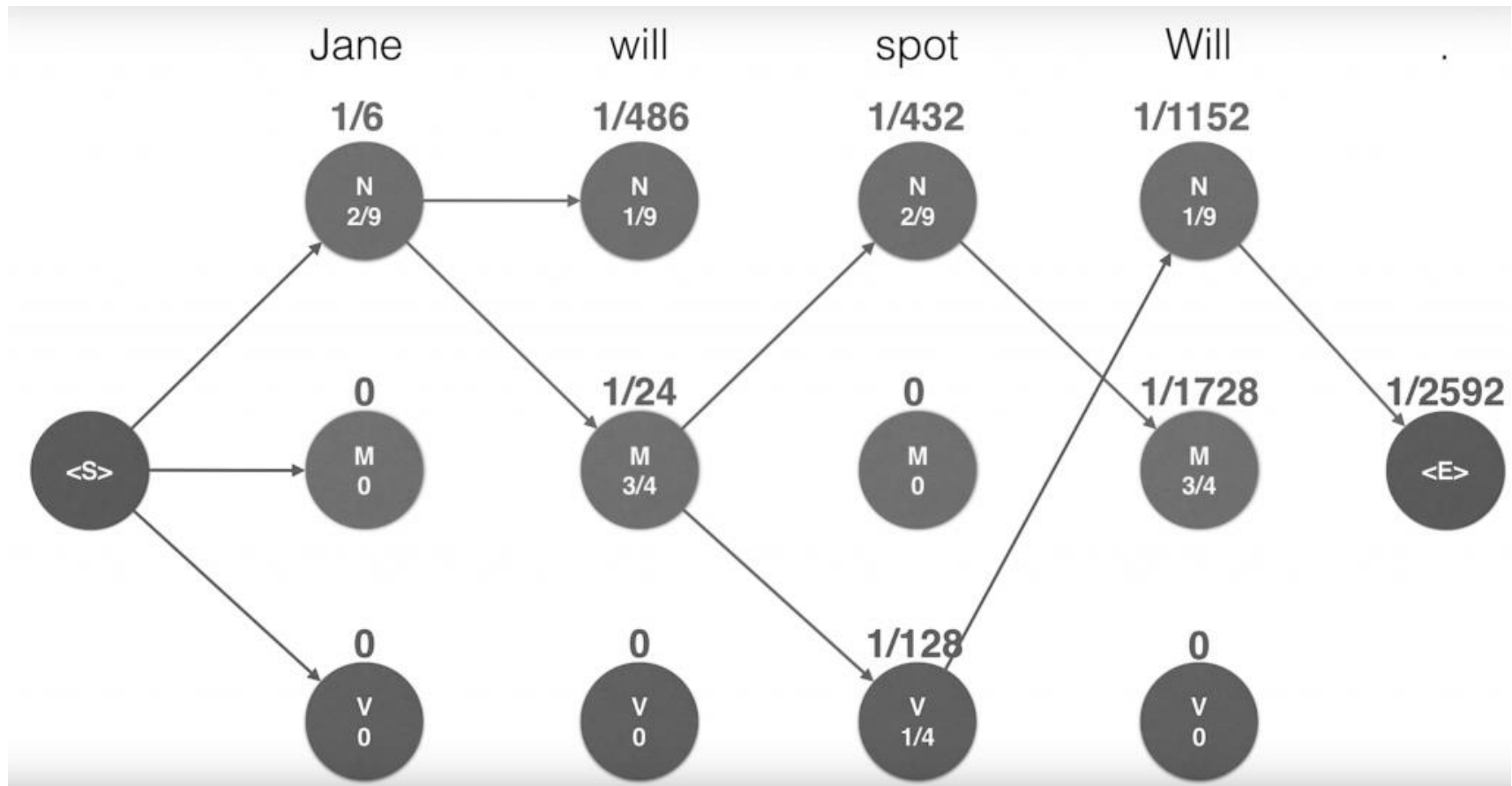
Jane

will

spot

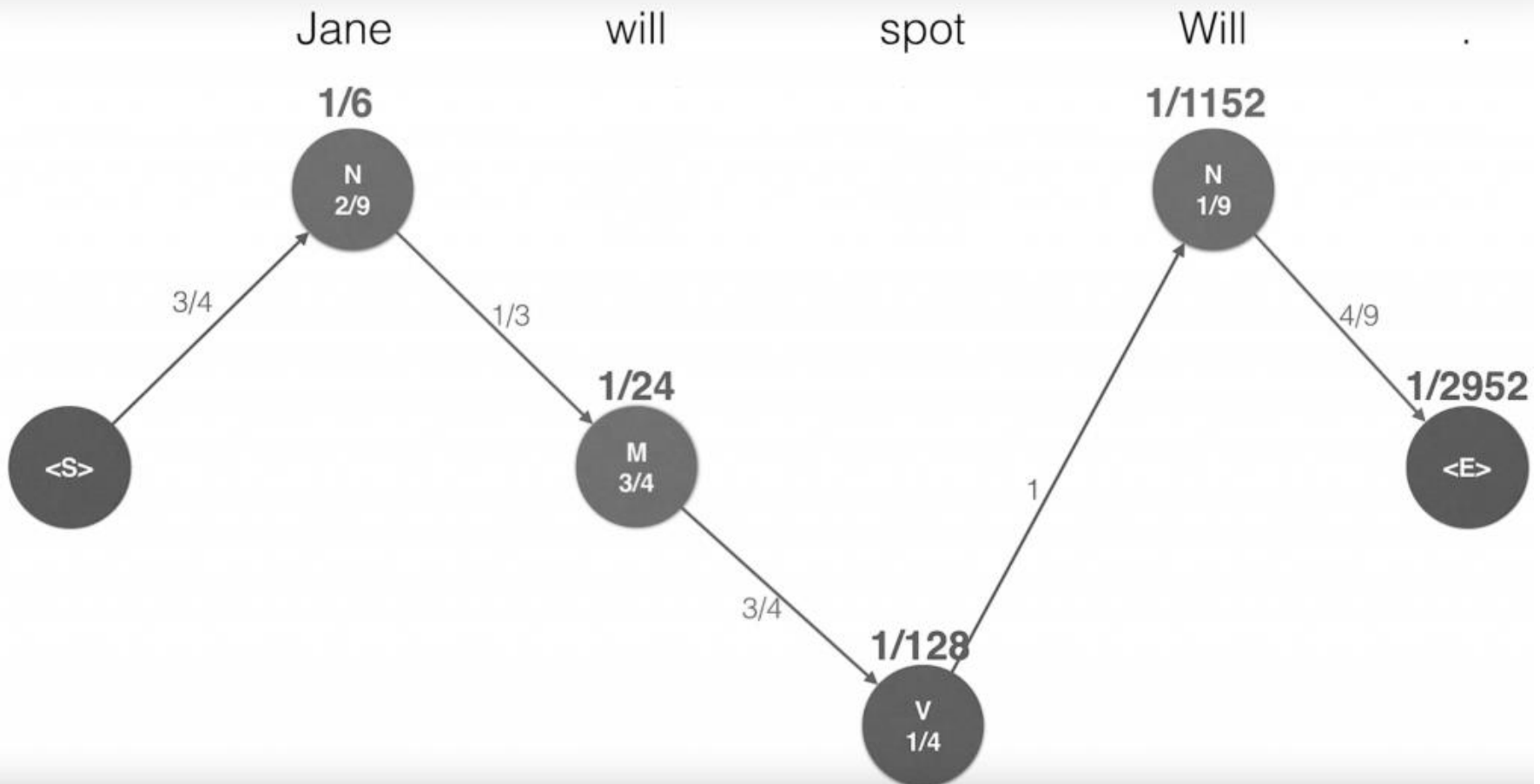
Will





This trellis diagram also can be written in matrix/table form, where rows are part of speech tags {N, M & V} and columns are words here in this case "Jane will spot will".

So how do we get our optimal path? Well, we'll start from the end and trace backwards. Since each state only has one incoming edge, then this should give us a path that comes all the way from the beginning. And that path is the winning path.



Final answer:

Jane



will



spot



Will

