CHAPTER III

Morphological Analysis

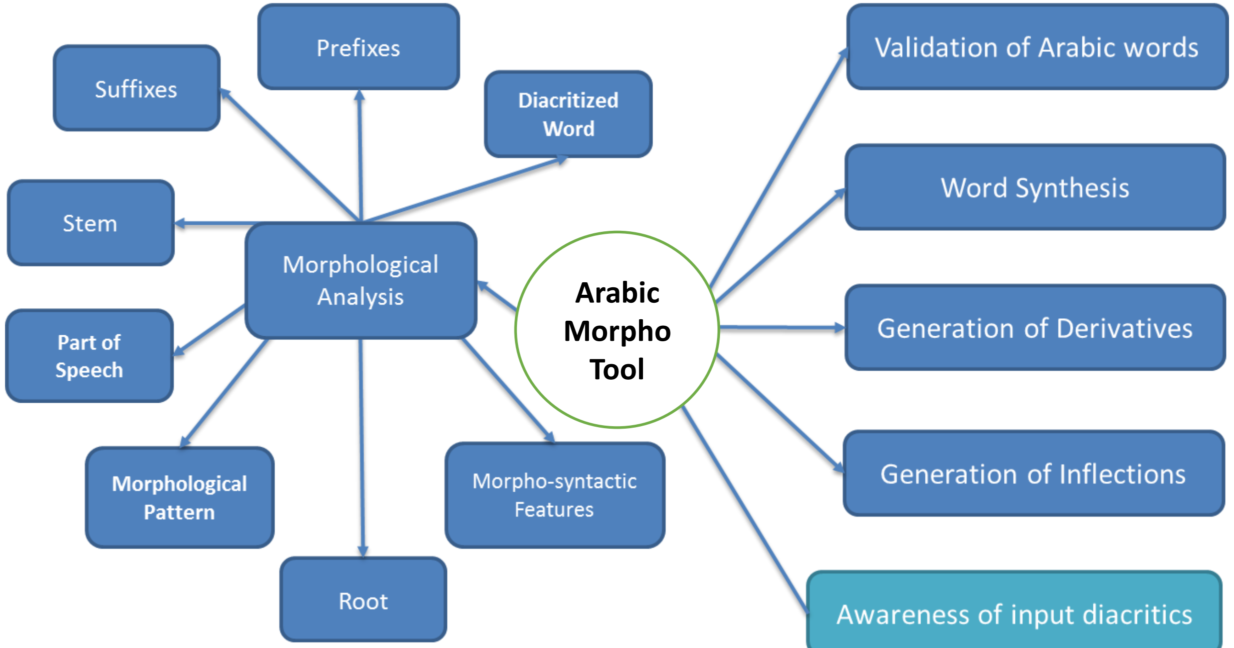
Morphological analysis, in NLP, refers to the computational processing of word structures

Morphological level

- Word formatting
- Inflection
- Derivation
- Stemming
- Lemmatization
- POS tagging

Morphological analysis
DEFINITION

Morphology is the branch of **linguistics** concerned with the **structure** and form of **words** in a language. It aims to break down words into their constituent parts, such as **roots**, **prefixes**, and **suffixes**, and understand their roles and meanings



Morphological analysis
IMPORTANCE

- **Understanding Word Formation**: Identifying the basic building blocks of words, which is crucial for language comprehension.

- **Improving Text Analysis**: Breaking down words into their roots and affixes, enhances the accuracy of text analysis tasks like sentiment analysis and topic modeling.

- **Enhancing Language Models**: Providing detailed insights into word formation, improving the performance of language models used in tasks like speech recognition and text generation.

- **Facilitating Multilingual Processing**: Handling the morphological diversity of different languages.

Morphological analysis

KEY TECHNIQUES

- **Stemming**: reduces words to their base or root form, usually by removing suffixes. The resulting stems are not necessarily valid words but are useful for text normalization

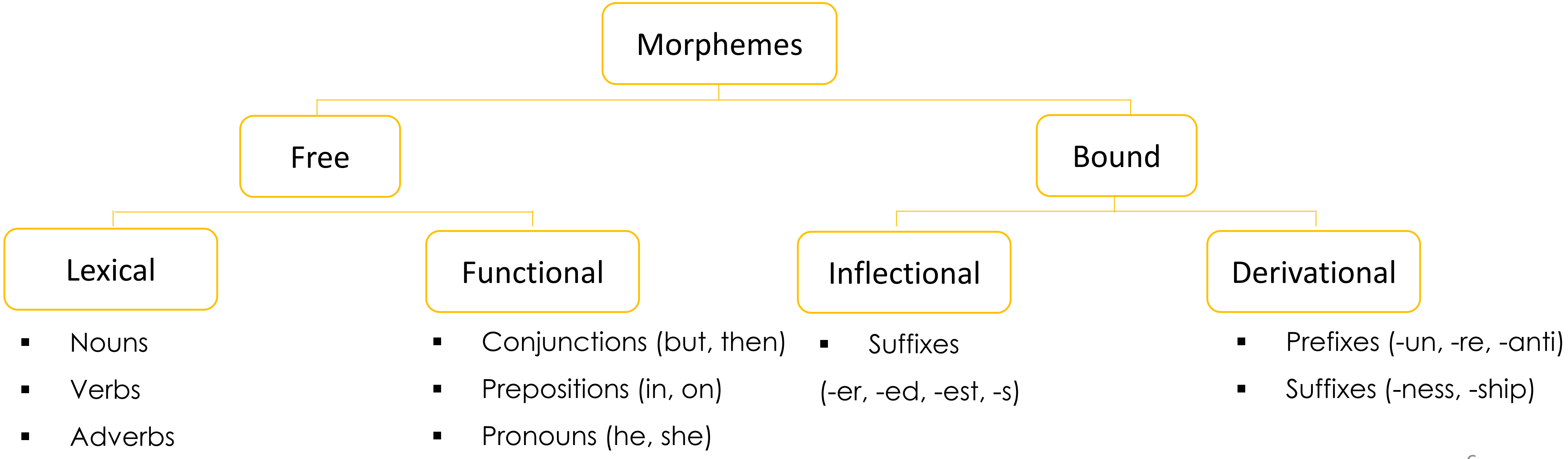
- **Lemmatization**: reduces words to their base or dictionary form (lemma). It considers the context and part of speech, producing valid words (using lexical databases)

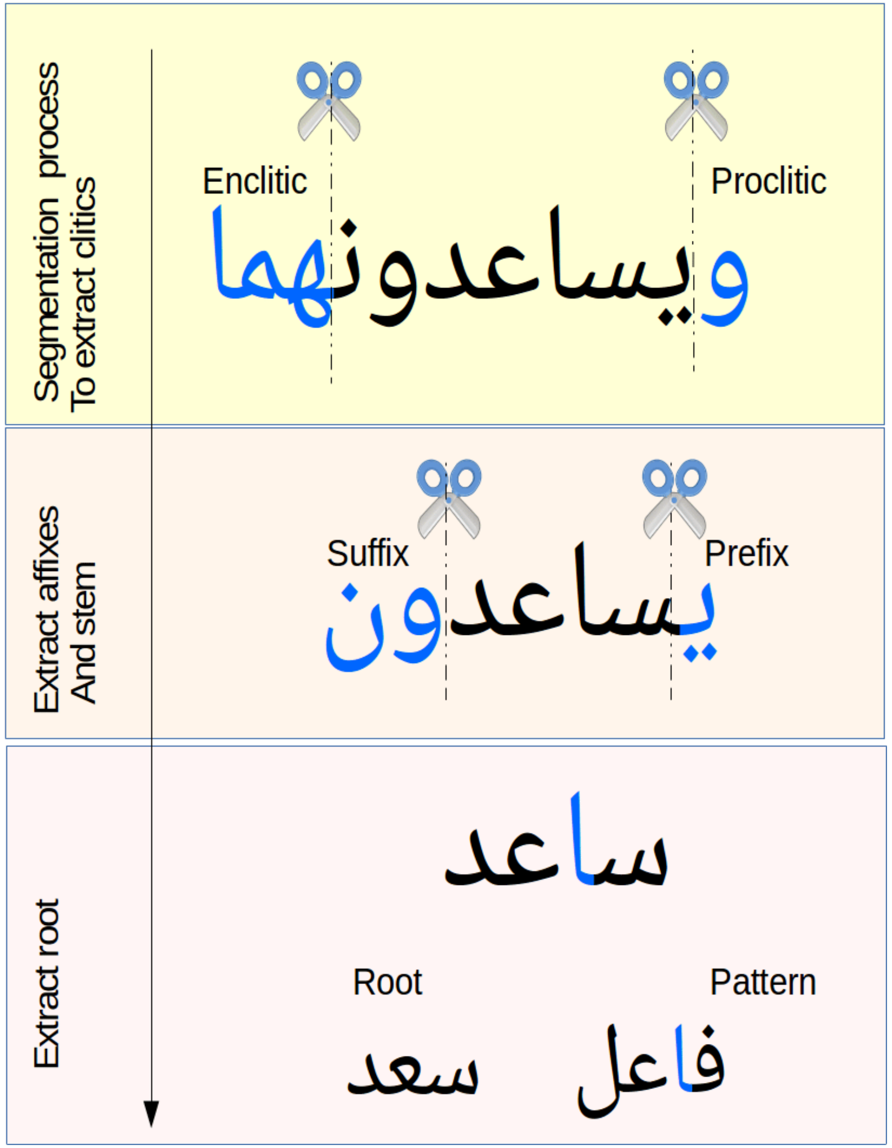
- **Morphological parsing**: involves analyzing the structure of words to identify their morphemes (roots, prefixes, suffixes). It requires knowledge of morphological rules and patterns.

MORPHEME

The words are often made up of smaller units called morphemes (smallest meaningful units).

- Lexical morpheme (Entries of a dictionary): Base form
- Grammatical morpheme (Affixes)




Segmentation process
To extract clitics
Enclitic
Proclitic
ويساعدونهما
Extract affixes
And stem
Suffix
Prefix
يساعدون
Extract root
ساعد
Root
Pattern
سعد
فاعل

INFLECTION

Morphological **Inflection** (التصريف) is the task of **generating** a target (**inflected form**) word from a source word (**base form**), given a morphological attribute (e.g. number, tense, and person)

Affixation (الزيادة)	Duplication (التضعيف)	Alteration (الإبدال)
Petit, Petite, Petites	Zigzag	Blanc, Blanche
Work, Works, Working	Byebye, Pingpong	Major, Minor
درس، يدرس، درست	زلزل	قال، قول
طالب، طالبة	كيف كيف	اصطبر، اصتبر، فم، فو

DERIVATION

A derivation (الاشتقاق) derives a **new word** from an existing word by **adding**, **changing**, **or removing** an non-inflectional **affix**

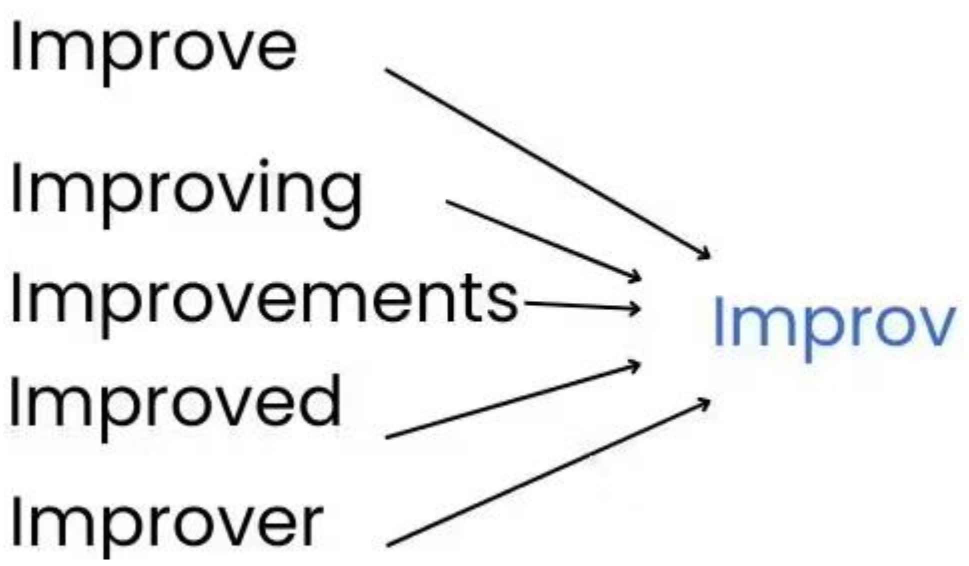
Affixation	With patterns (الأوزان)
Refaire, Fin**al**	فعل - كتب
Unhappy, Happi**ness**	فاعل - كاتب
كتب (مكتبة) ، مدرسة (درس)	مفعول - مكتوب
استعمل	مفعلة - مكتبة

WORD FORMATION

Composition (المركب المزجي)	Plate-Forme, TimeOut, حضرموت
Truncation (الترخيم)	Biblio, lab, exam, يا صاح
Portmanteau word (النحت)	Informatique, Transistor, بسملة (Information-Automatique) (Transfer-Resistor)
Acronym (الاختصار)	RADAR (RAdio Detection And Ranging)

STEMMING

Stemming merely removes common suffixes from word tokens



محمد، المسجد

Root-based Stemming

حمد، سجد

يسترجعون

Light stemming

يسترجع

LEMMATIZATION

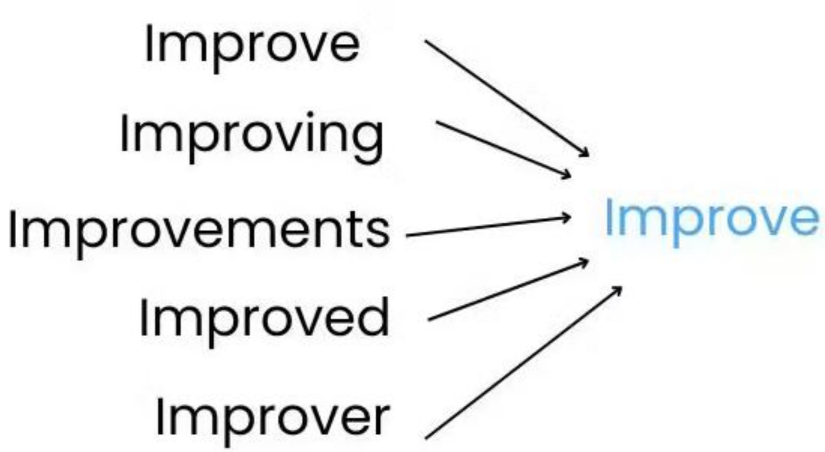
Lemmatization ensures the output word is an existing normalized form of the word (Lemma) that can be found in the dictionary

- ai
- as
- a
- avons
- avions

Avoir

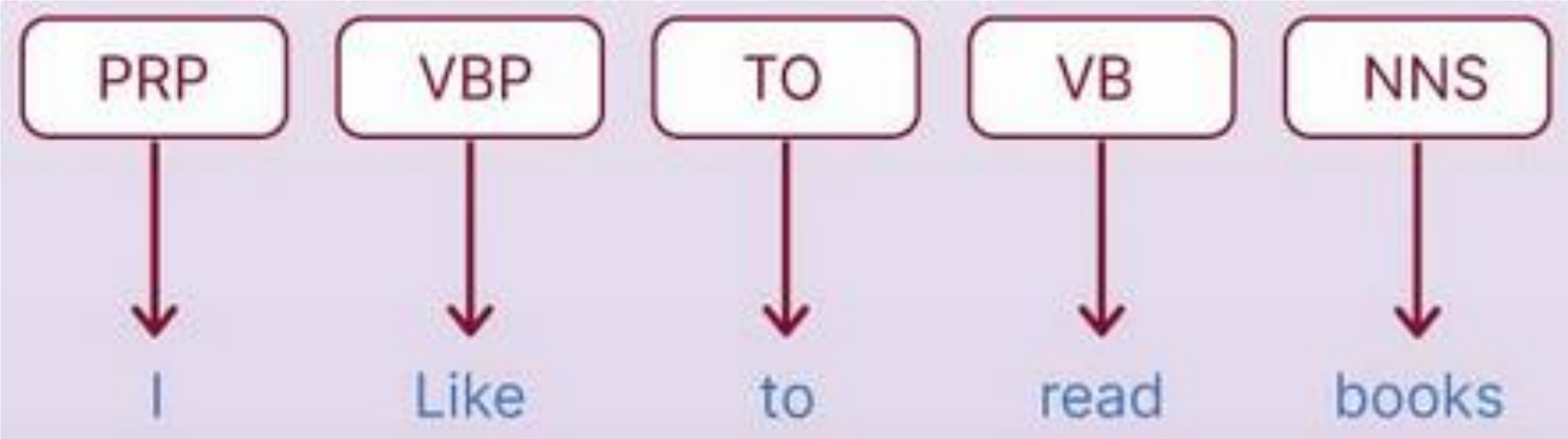
- قال
- يقول
- يقل
- قلت

قال



Parts-of-Speech Tagging

POS tagging is the process of assigning grammatical categories or “tags” to each word in a sentence based on its syntactic role



POS Use Cases

- **Information Retrieval**: Understand the relationships between words
- **Grammar Checking:** Identifying grammatical errors and suggesting corrections
- **Machine Translation:** Enhances the accuracy of translating sentences between languages.
- **Syntactic Analysis:** Understanding the grammatical structure of a sentence (e.g: identify the subject, verb, object)
- **Semantic Analysis:** Understanding the meaning of words in context. (e.g: distinguishing between a noun and a verb)
- **Named Entity Recognition (NER):** Providing information about the grammatical category of words. (e.g: "New York" is a proper noun)

POS Workflow

Tokenization	Loading Language Model	Text Preprocessing	Linguistic Analysis	POS Tagging
Divide the input text into discrete tokens	Utilize a library such as NLTK or SpaCy	Correct PoS labeling is aided by clear text	Determine the text's grammatical structure	Understanding each word's purpose inside the sentence, (adjective, verb, noun, ..)

Types of POS Tagging

- **Rule-Based:** involves assigning words their respective parts of speech using predetermined rules

  Example: (Rule: Assign the POS tag "noun" to words ending in "-tion" or "-ment.")

- **Statistical:** Utilizing probabilistic models (e.g: Using HMMs)

- **Neural-Based:** Use neural networks to learn patterns from data

POS Challenges

- **Ambiguity:** Words with multiple meanings
- **Out-of-Vocabulary Words:** Words not present in the training data
- **Complexity in Morphologically Rich Languages:** Complex word forms
- **Training Data Dependency:** Inadequate training data may lead to inaccurate tagging
- **Parsing Errors:** Errors in POS tagging can propagate to downstream parsing tasks

Sentence	POS tag of word "back"
The "back" door	ADJECTIVE
On my "back"	NOUN
Win the voters "back"	ADVERB
Promised to "back" the bill	VERB

The 30 commonly used POS tags

Abbreviation	Meaning	Example
CC	Coordinating Conjunction	and, but, or
CD	Cardinal Number	1, 2, 3, one, two, three
DT	Determiner	the, a, an
IN	Preposition or Subordinating Conjunction	in, on, after
JJ	Adjective	big, happy, green
MD	Modal	can, could, will
NN	Noun, Singular or Mass	cat, dog, love
NNS	Noun, Plural	cats, dogs
NNP	Proper Noun, Singular	London, Alex
PRP	Personal Pronoun	I, he, she
RB	Adverb	quickly, very
TO	to	to
UH	Interjection	oh, ah, wow
VB	Verb, Base Form	eat, walk, run
VBD	Verb, Past Tense	ate, walked, ran
VBG	Verb, Gerund or Present Participle	eating, walking, running
VBN	Verb, Past Participle	eaten, walked, run
VBP	Verb, Non-3rd Person Singular Present	eat, walk, run
VBZ	Verb, 3rd Person Singular Present	eats, walks, runs
WDT	Wh-Determiner	which, whatever
WP	Wh-Pronoun	what, who, whom
WRB	Wh-Adverb	where, when, how
PRP$	Possessive Pronoun	my, his, hers
JJR	Adjective, Comparative	bigger
JJS	Adjective, Superlative	biggest
RP	Particle	up, off, about
FW	Foreign Word	café, elite
NNPS	Proper Noun, Plural	Americans
EX	Existential There	there

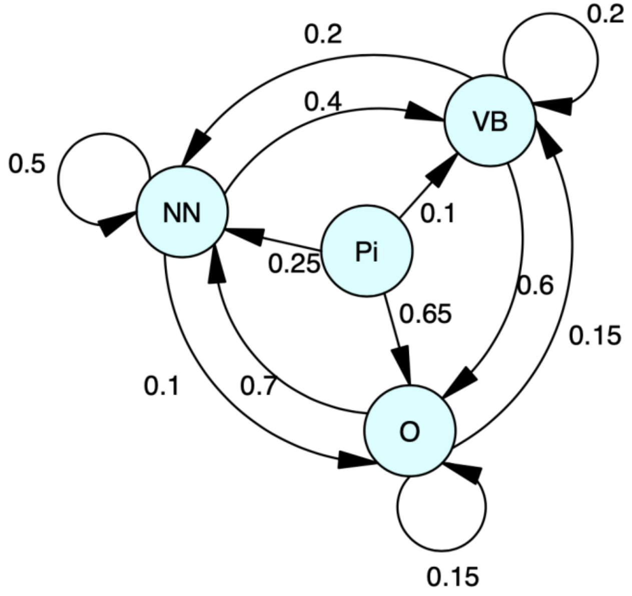
POS: Example

“The quick brown fox jumps over the lazy dog.”

- “The” (DT): Determiner
- “quick” (JJ): Adjective
- “brown” (JJ): Adjective
- “fox” (NN): Noun
- “jumps” (VBZ): Verb
- “over” (IN): Preposition
- “the” (DT): Determiner
- “lazy” (JJ): Adjective
- “dog” (NN): Noun

POS with HMMs

HMM-based POS tagging model undergoes **training** on a sizable annotated text **corpus** to discern patterns in various parts of speech. Leveraging this training, the model predicts the POS tag for a given word based on the **probabilities** associated with different **tags** within its context.



A =

	NN	VB	O
Pi	0.25	0.1	0.65
NN	0.5	0.4	0.1
VB	0.2	0.2	0.6
O	0.7	0.15	0.15

Origins of HMMs

- Hidden Markov Models (HMM) were introduced by **Baum** in the **1970s**; this model is inspired by **probabilistic automata**

- A **probabilistic automata** is defined by a structure composed of **states** and **transitions** and by a set of **probability** distributions over the transitions. Each transition is associated with a symbol from a finite **alphabet**. This symbol is generated each time the transition is taken

HMM: Definition

- An **HMM** is also defined by a structure composed of **states** and **transitions** and by a set of **probability** distributions over the transitions

- The essential **difference** with probabilistic automata is that the **generation of symbols occurs at the states rather than on the transitions**. Additionally, each **state** is associated not with a single symbol but with a **probability distribution over the symbols** of the alphabet

HMM applications

HMMs are used in the following fields:

- Speech recognition
- Handwritten text recognition
- DNA sequence recognition
- Information extraction
- POS tagging, etc.

HMM Formarilization

An HMM is defined by a quadruplet (S, ∑, T, G)

- H=(S, ∑, T, G)
- S : a set of N states, it contains two particular states : Start et End indicating the beginning and end of a sequence
- ∑ : an Alphabet composed of M symbols.
- T : a matrix that indicates the probabilities of transition between states
    - T = S-{end} x S-{start} → [0,1]
- G : a matrix that indicates the probabilities of emission for states
    - G : S-{start,end} x ∑ → [0,1]

HMM Formarilization

- Consider **P(o/s)**, the probability of generating the symbol **o** by the state **s.**
- We do associate to each state **s** :
  - a distribution of transition probabilities :

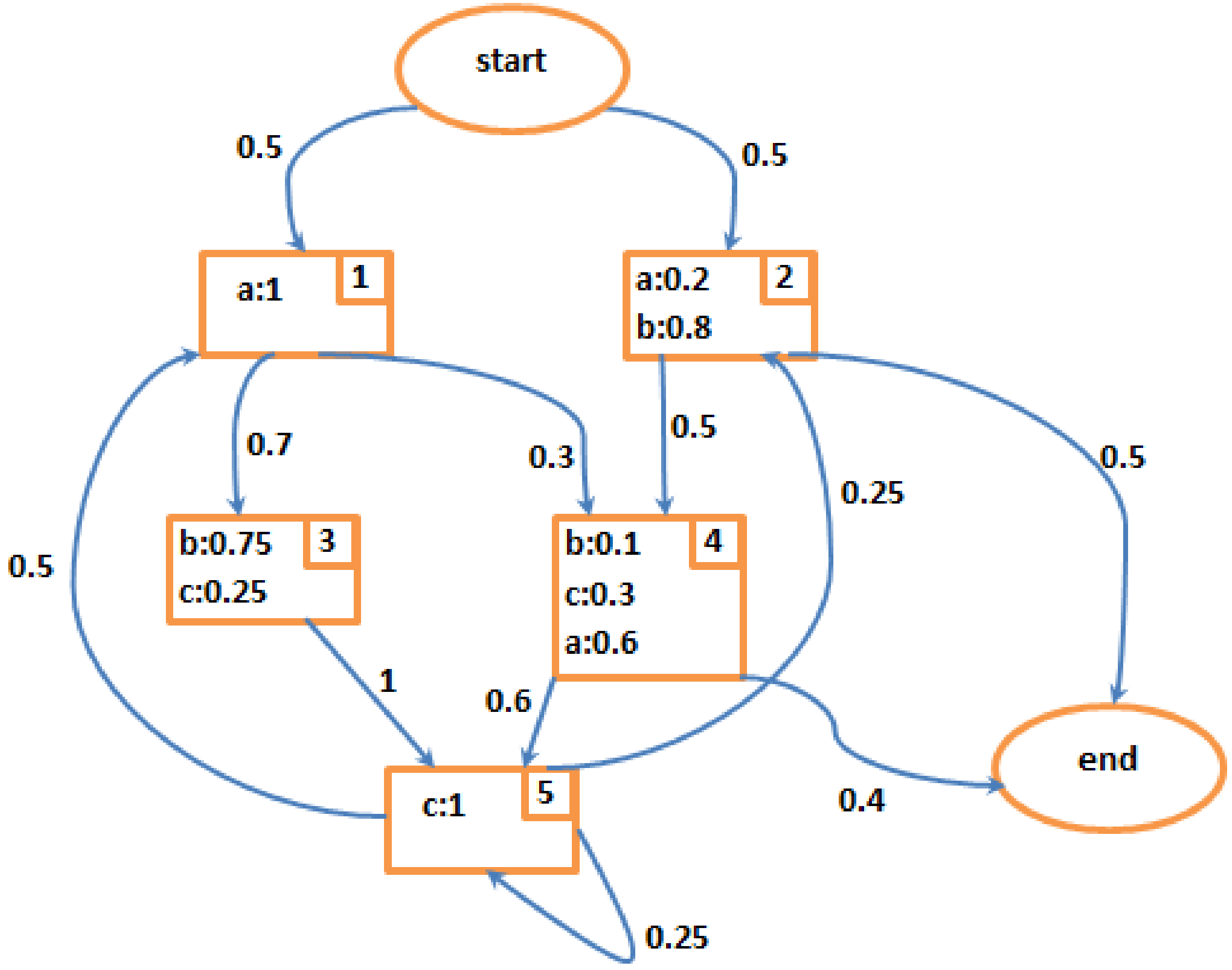
$$\sum_{s' \epsilon S} P(s \rightarrow s') = 1$$

  - a distribution of emission probabilities :

$$\sum_{o' \in \Sigma} P(o' \ / \ s) = 1$$

HMM: Example

- The figure shows an example of HMM with 7 states and 11 transitions :



HMM: Example

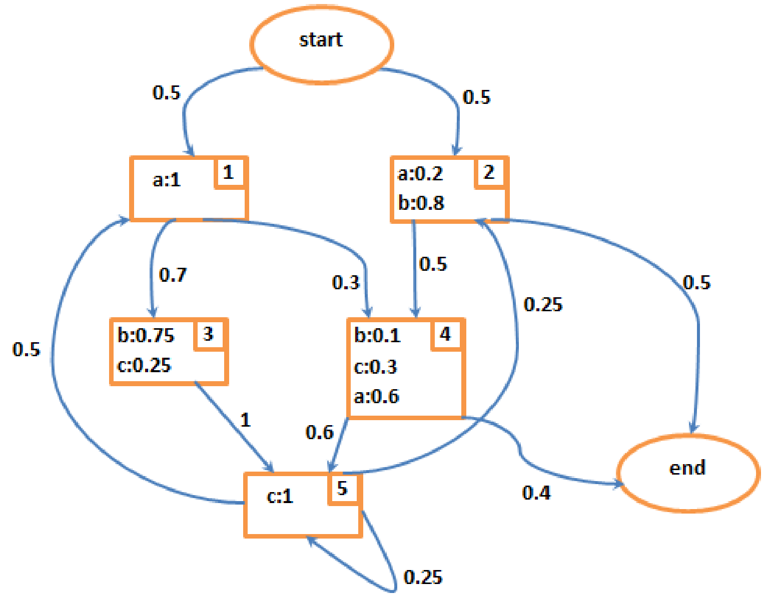
- S={start,1,2,3,4,5,end}
- ∑={a,c,b}

- T : Transition matrix

	1	2	3	4	5	end
start	0.5	0.5				
1			0.7	0.3		
2				0.5		0.5
3					1	
4					0.6	0.4
5	0.5	0.25			0.25	

- G : Emission matrix

	a	b	c
1	1		
2	0.2	0.8	
3		0.75	0.25
4	0.6	0.1	0.3
5			1



HMM: Example

This HMM allows to generate the following observable sequences:

abca, aacb, ab,…etc.

To these observable sequences correspond the following hidden sequences:

1-3-5-2, 1-4-5-2, 2-4

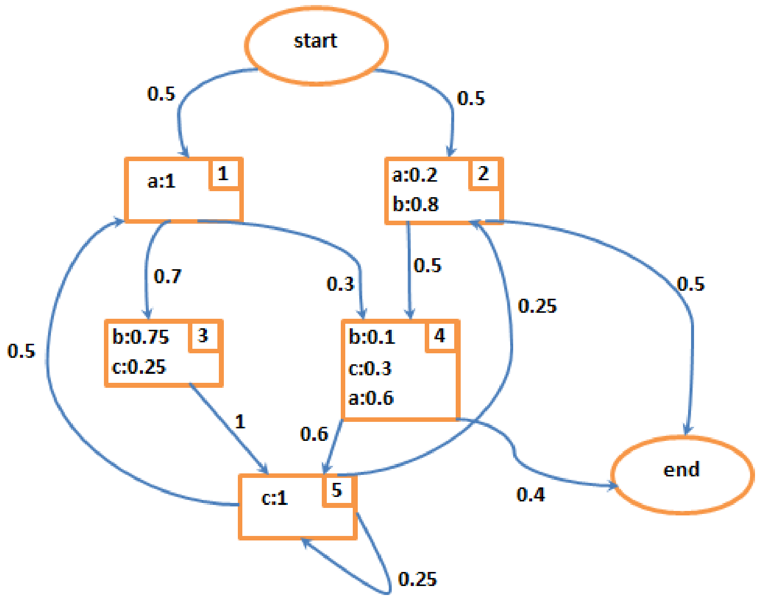
Each observable sequence could be generated by lot of possible paths.

For example, the sequence **abccb** could be generated by:

Path 1 : start-1-3-5-5-2-end

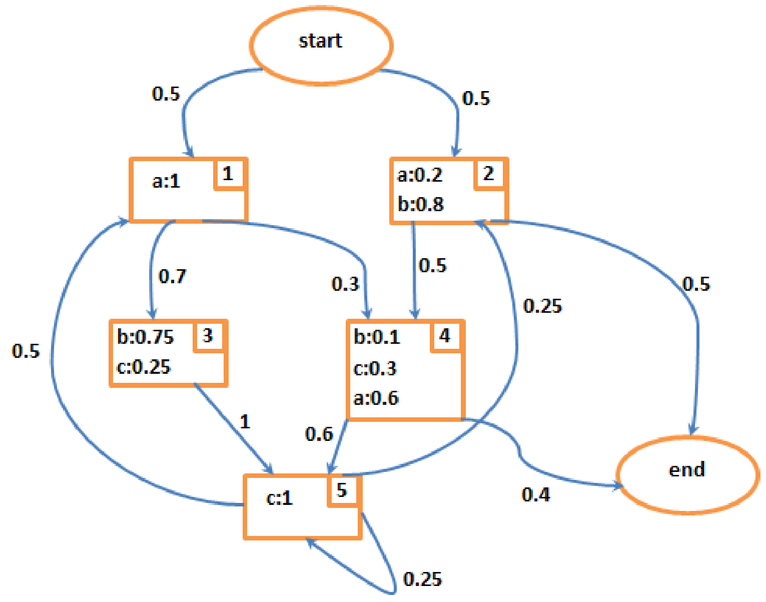
Path 2 : start-1-4-5-5-2-end

Path 3 : start-2-4-5-5-2-end



HMM: Example

What will be the probability of generating **abccb** by this HMM**?**



Path 1 : start-1-3-5-5-2-end

Path 2 : start-1-4-5-5-2-end

Path 3 : start-2-4-5-5-2-end

P(path 1) = $(0.5 \times 1) \times (0.7 \times 0.75) \times (1 \times 1) \times (0.25 \times 1) \times (0.25 \times 0.8) \times (0.5) = 6.5 \times 10^{-3}$

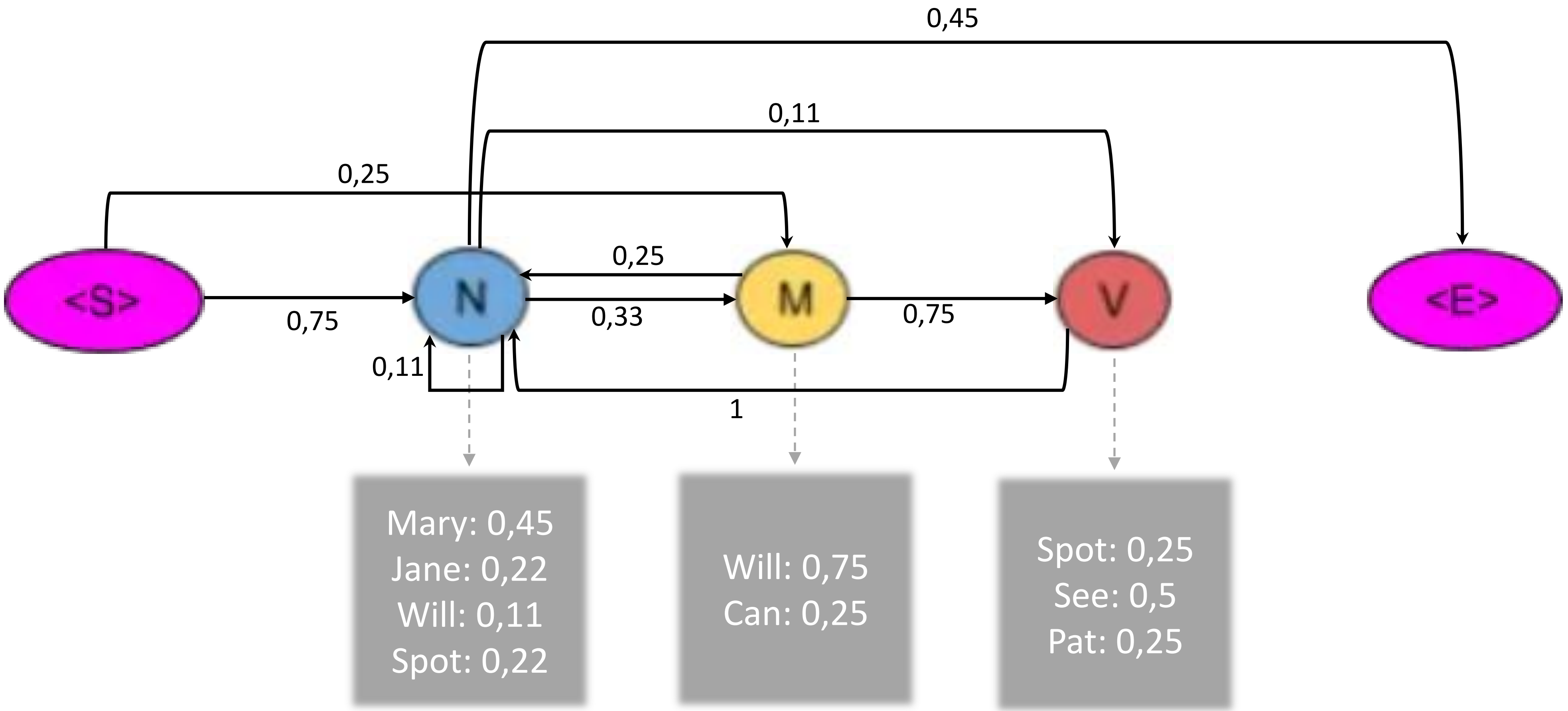
P(path 2) = $(0.5 \times 1) \times (0.3 \times 0.1) \times (0.6 \times 1) \times (0.25 \times 1) \times (0.25 \times 0.8) \times (0.5) = 2.2 \times 10^{-3}$

P(path 3) = $(0.5 \times 0.2) \times (0.5 \times 0.1) \times (0.6 \times 1) \times (0.25 \times 1) \times (0.25 \times 0.8) \times (0.5) = 0.75 \times 10^{-3}$

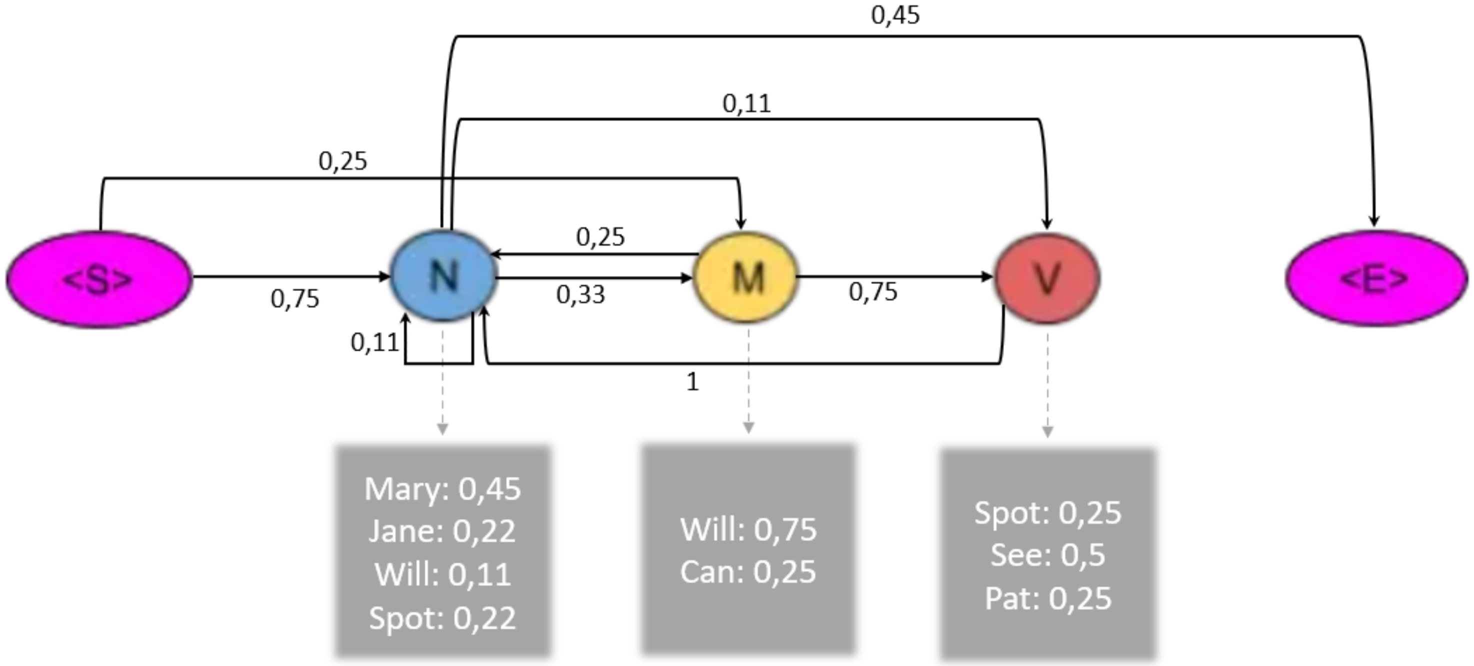
The probability of generating the sequence abccb by this HMM is:

P(abccb) = $(6.5 + 2.2 + 0.75) \times 10^{-3} = 9.45 \times 10^{-3}$

POS with HMM: Example



POS with HMM: Example



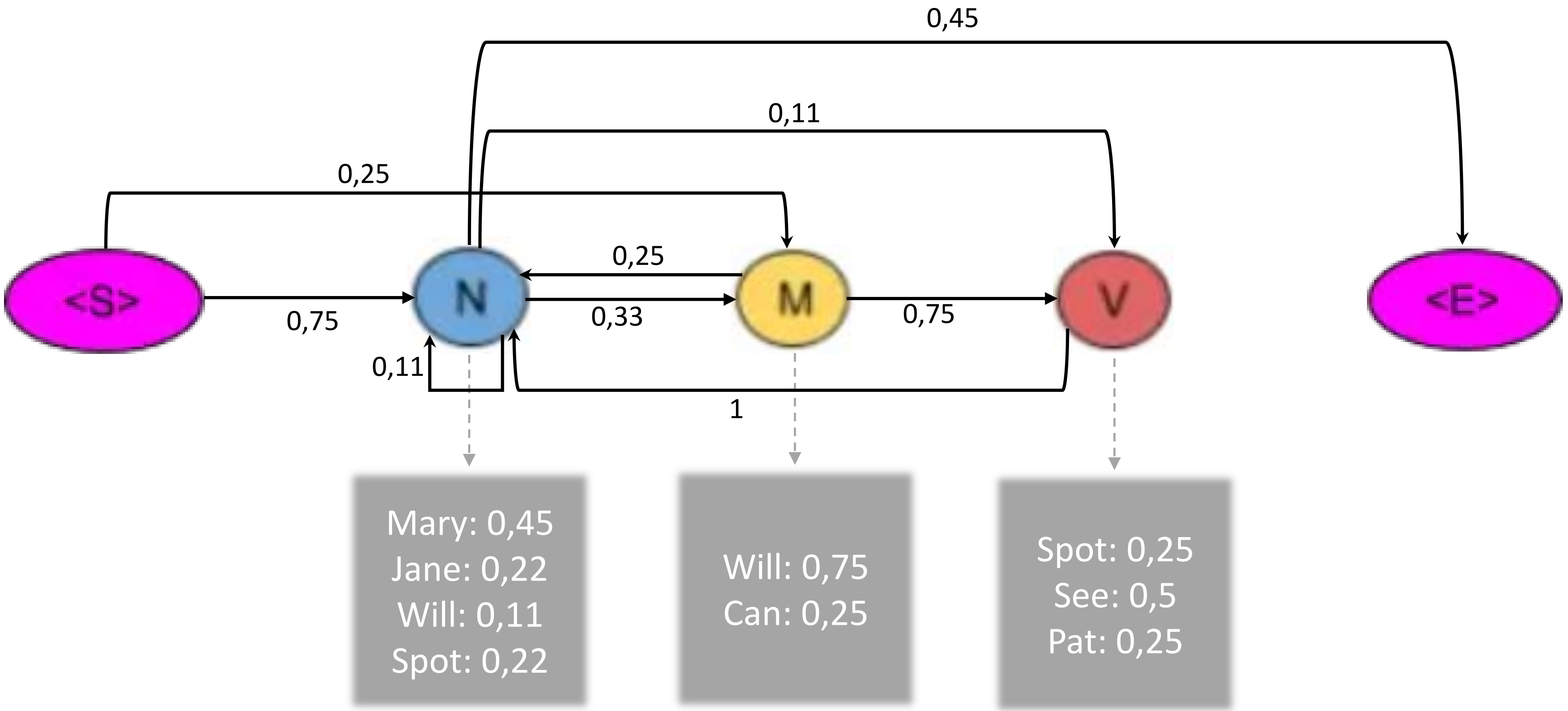
	N	M	V	<E>
<S>	0,75	0,25		
N	0,11	0,33	0,11	0,45
M	0,25		0,75	
V	1			

T: Transition matrix

	Mary	Jane	Will	Spot	Can	See	Pat
N	0,45	0,22	0,11	0,22			
M			0,75		0,25		
V				0,25		0,5	0,25

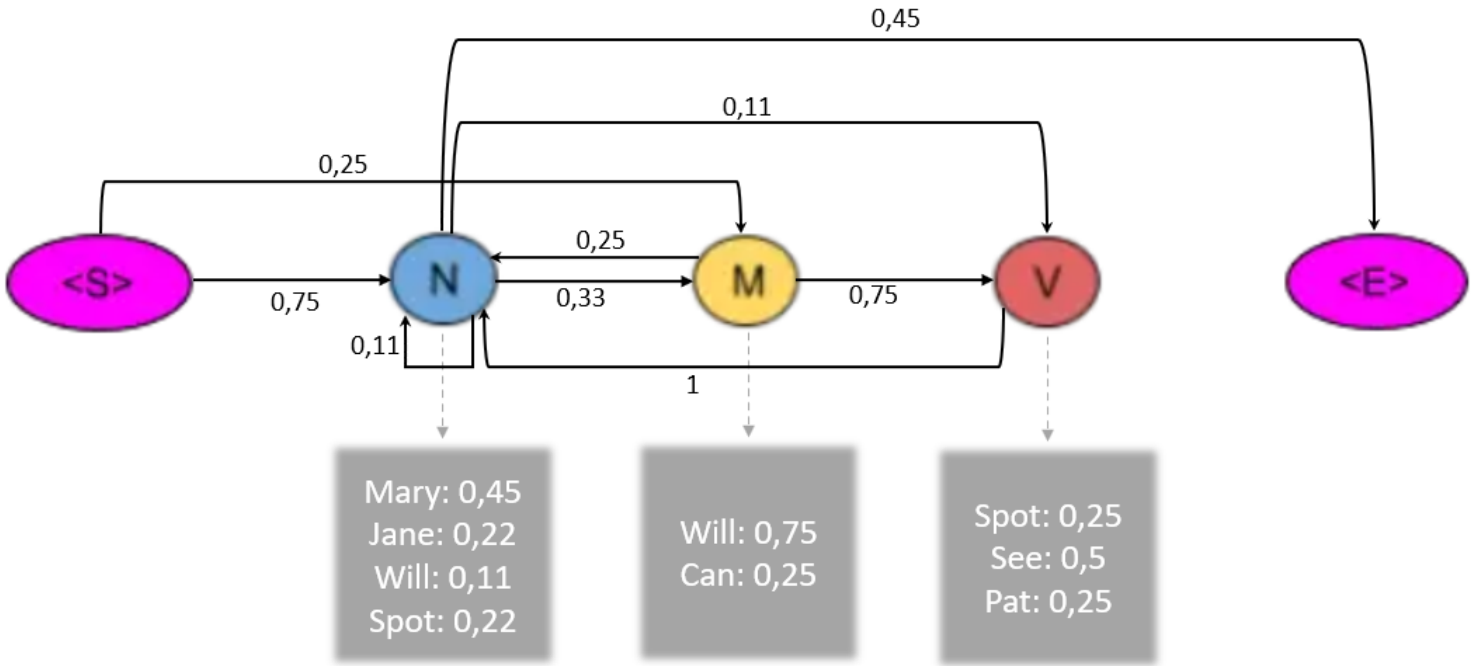
G: Emission matrix

POS with HMM: Example



POS of « Will can spot Mary » ?

POS of « Will can spot Mary » **?**



Path 1 = <S>→N→M→N→N→<E>
P(Path 1) = (0,75x0,11) x (0,33x0,25) x (0,25x0,22) x (0,11x0,45) x (0,45) = 0,0000083385

Path 2 = <S>→N→M→V→N→<E>
P(Path 2) = (0,75x0,11) x (0,33x0,25) x (0,75x0,25) x (1x0,45) x (0,45) = **0,00025842**

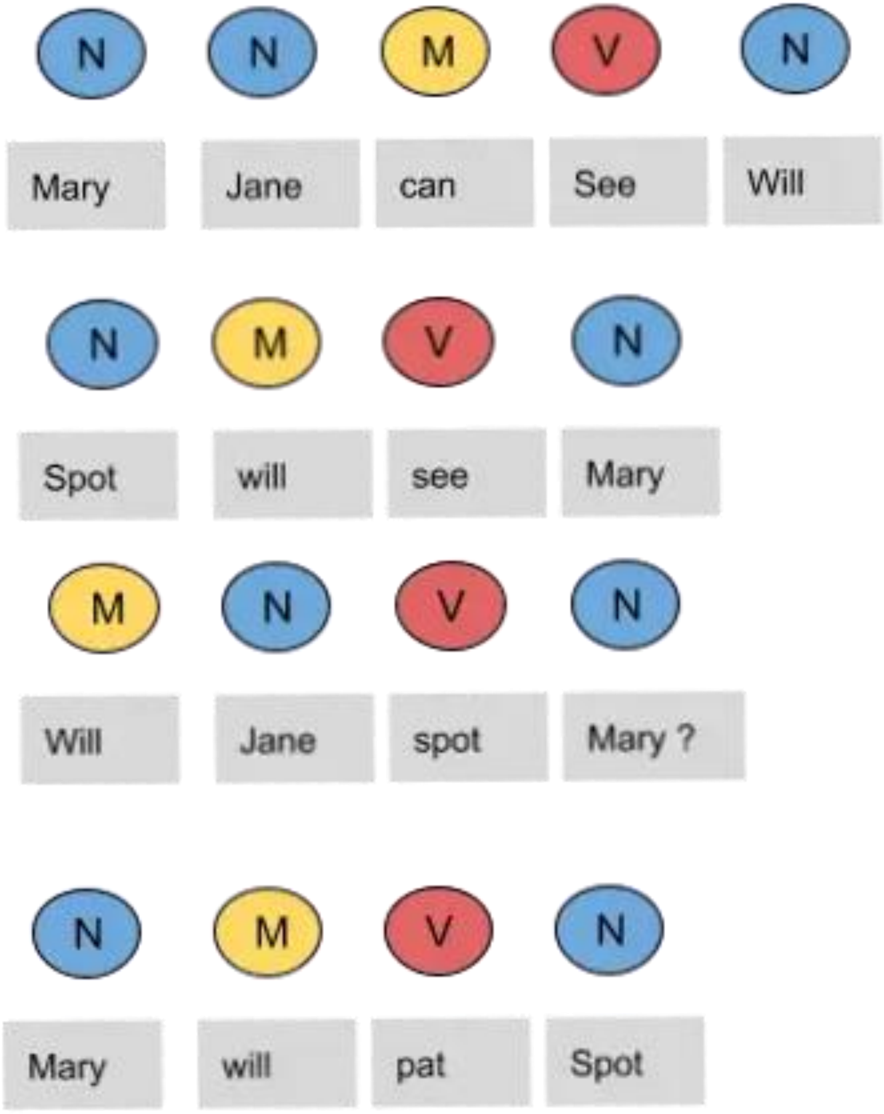
The probability of the second sequence is much higher

POS Tags : {Will : N, can : M, spot : V, Mary : N}

How to train HMM-Based POS tagger?

Let's consider the following corpus:

- Mary Jane can see Will
- Spot will see Mary
- Will Jane spot Mary**?**
- Mary will pat Spot



Emission matrix

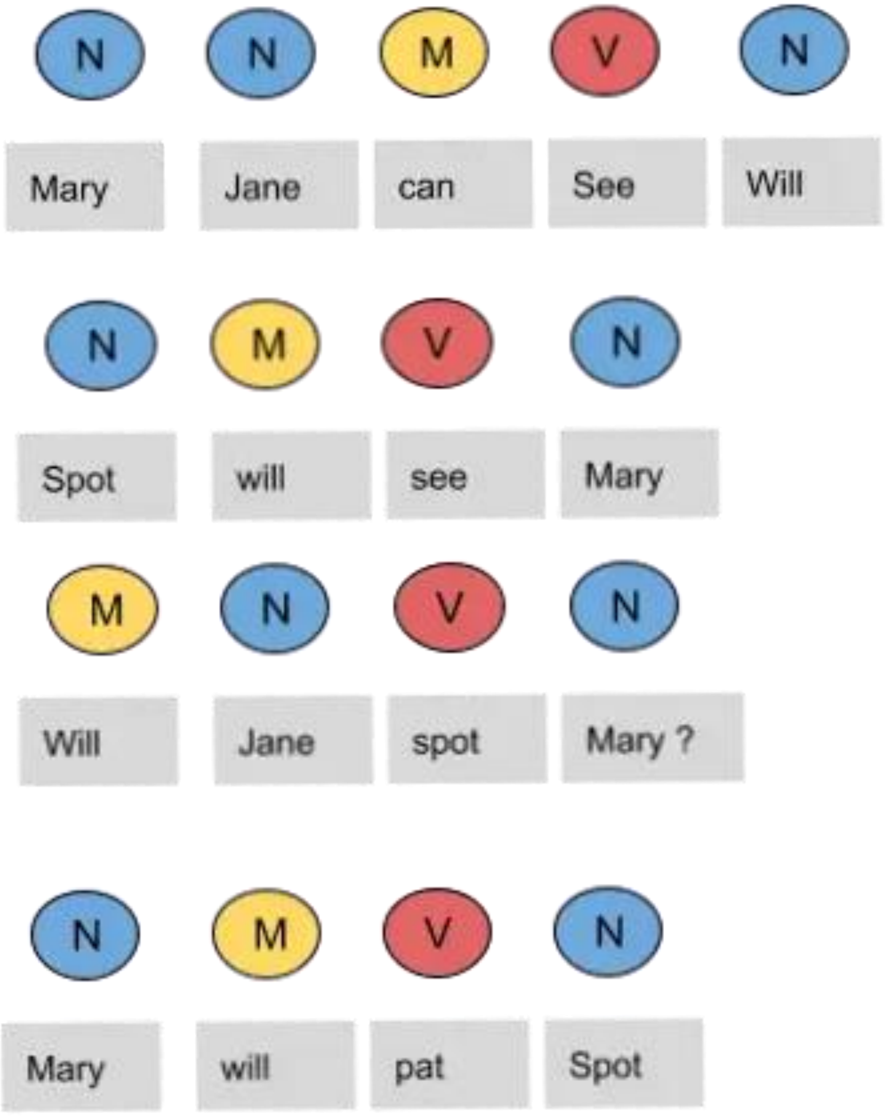
1.The word “Mary” appears four times as Noun. The word “Will” appears three times as Model and one time as Noun, etc.

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

2.Let’s divide each tag by the total number of their appearances (e.g: Noun appears nine times)

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

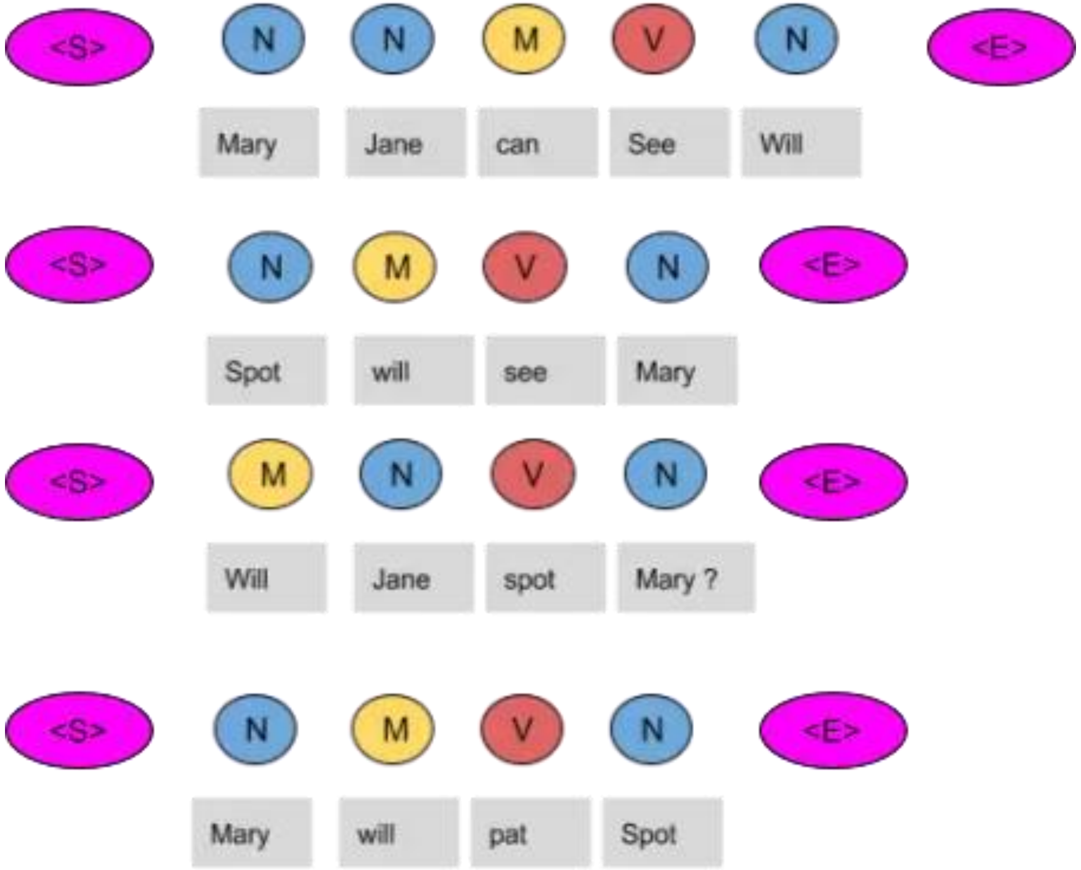
Emission matrix



Transition matrix

1.Let's count the co-occurrences of tags. (e.g: <S> is followed by Noun three times and by Model one time, etc.)

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



2.Let's divide each term in a row by the total number of co-occurrences of the tag (e.g: Model is followed by any other tag four times, etc.

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

Transition matrix

Emission matrix & Transition matrix

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

	Mary	Jane	Will	Spot	Can	See	Pat
N	0,45	0,22	0,11	0,22			
M			0,75		0,25		
V				0,25		0,5	0,25

G: Emission matrix

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

	N	M	V	<E>
<S>	0,75	0,25		
N	0,11	0,33	0,11	0,45
M	0,25		0,75	
V	1			

T: Transition matrix

HMM challenges

Let's consider H an HMM and a given sequence of symbols $O=O_1O_2\ldots O_t$

- What is the probability of generating O with H**?**

Solution: Forward-backward algorithm

- What is the sequence of states $S=S_1S_2\ldots S_t$ in H that has the maximum probability of generating O**?**

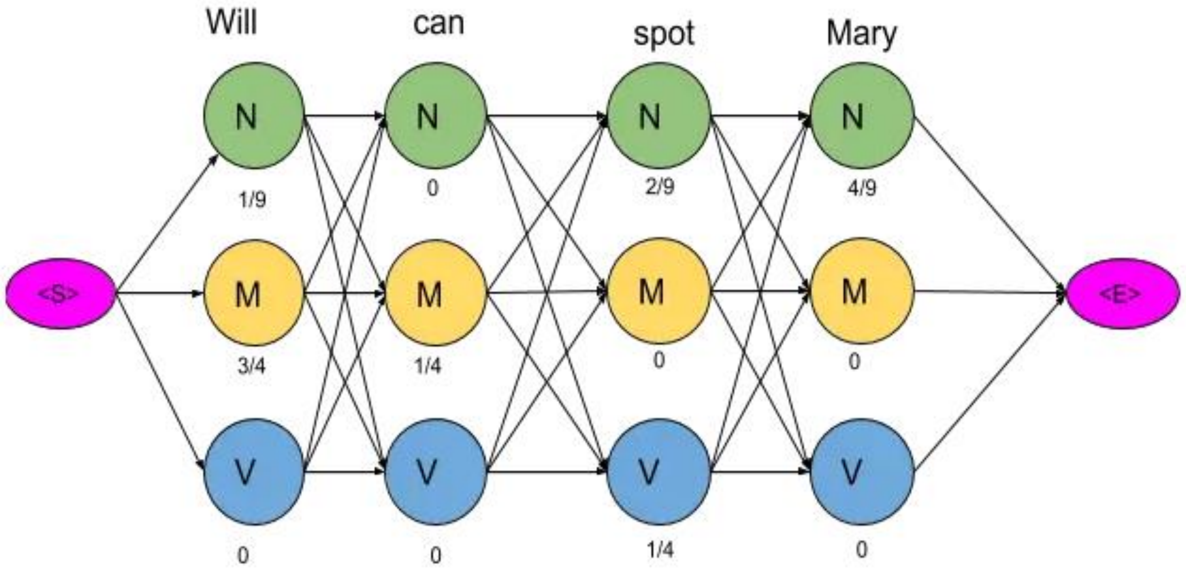
Solution: Viterbi algorithm

- How to adjust the parameters of H (transition and emission probabilities) to best represent the sequences being processed**?**

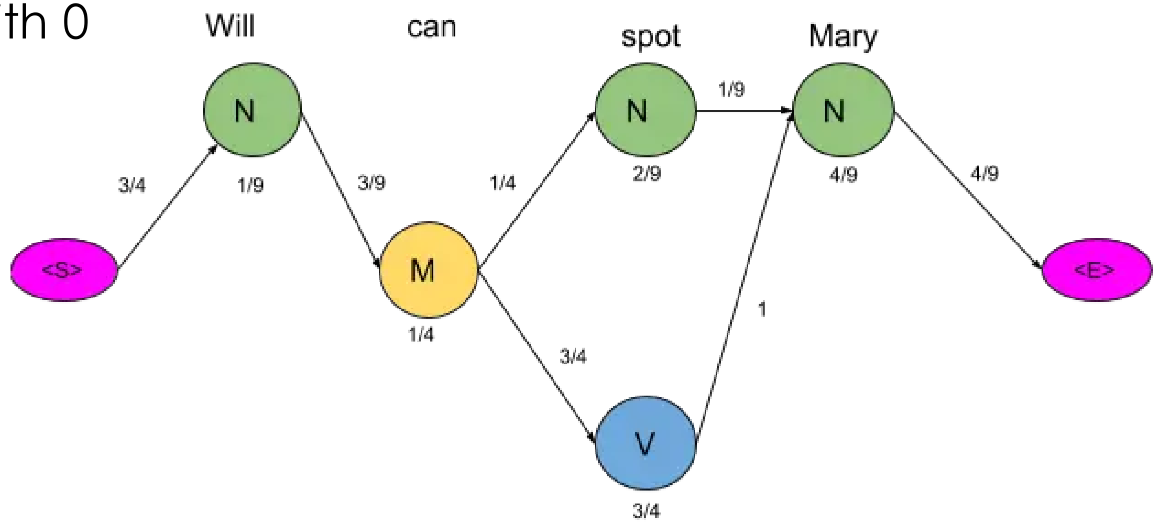
Solution: Baum-Welch algorithm

Viterbi Algorithm

1. Develop all possible paths

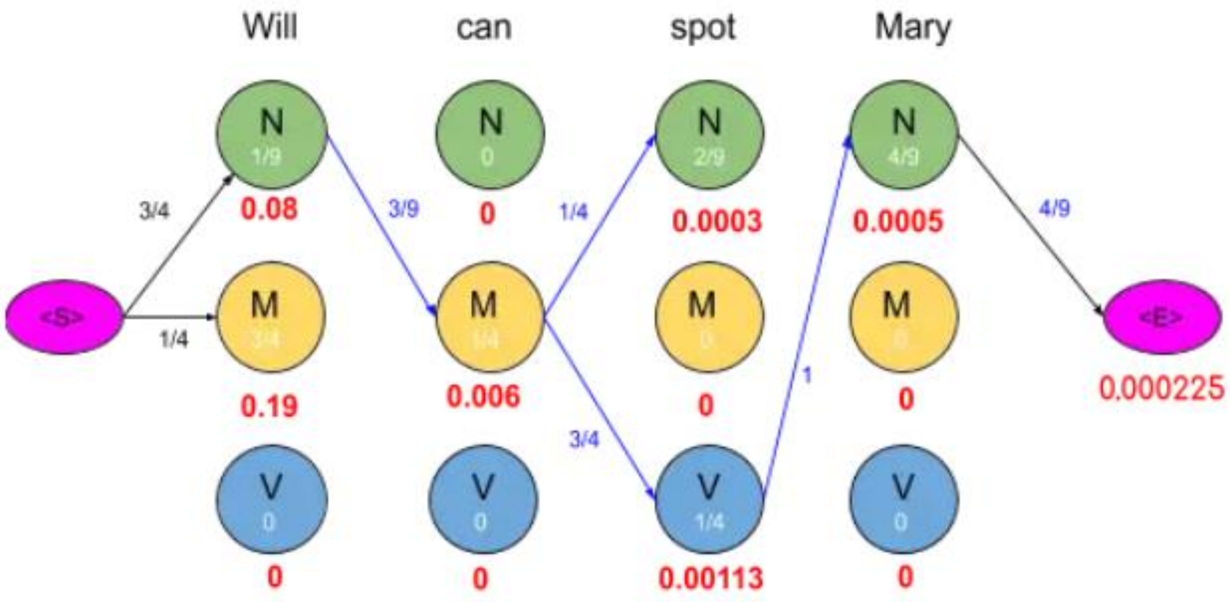


1. Remove edges and vertices with 0



Viterbi Algorithm

3. Calculate probability of all paths leading to a node then remove edges and paths with lower probability



4. start from the end and trace backward (since each state has only one incoming edge)