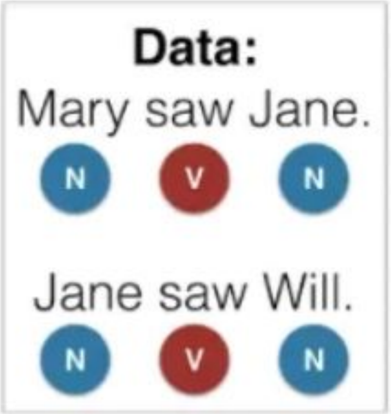
This is AI4001

GCR : t37g47w

POS Tags

Mary saw Will.



WHY DO WE NEED POS Tagging?

These tags reveal a lot about a word and its neighbors. (nouns are preceded by determiners and adjectives, verbs by nouns).

Gives an idea about syntactic structure (nouns are generally part of noun phrases), hence helping in text parsing.

Parts of speech are useful features for labeling named entities like people or organizations in information extraction.

Two classes of words: Open vs. Closed

Closed class words

- Relatively fixed membership
- Usually **function** words: short, frequent words with grammatical function
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***

Open class words

- Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: **oh, ouch, uh-huh, yes, hello**
- New nouns and verbs like *iPhone* or *to fax*

Part-of-Speech Tagging

Assigning a part-of-speech to each word in a text.

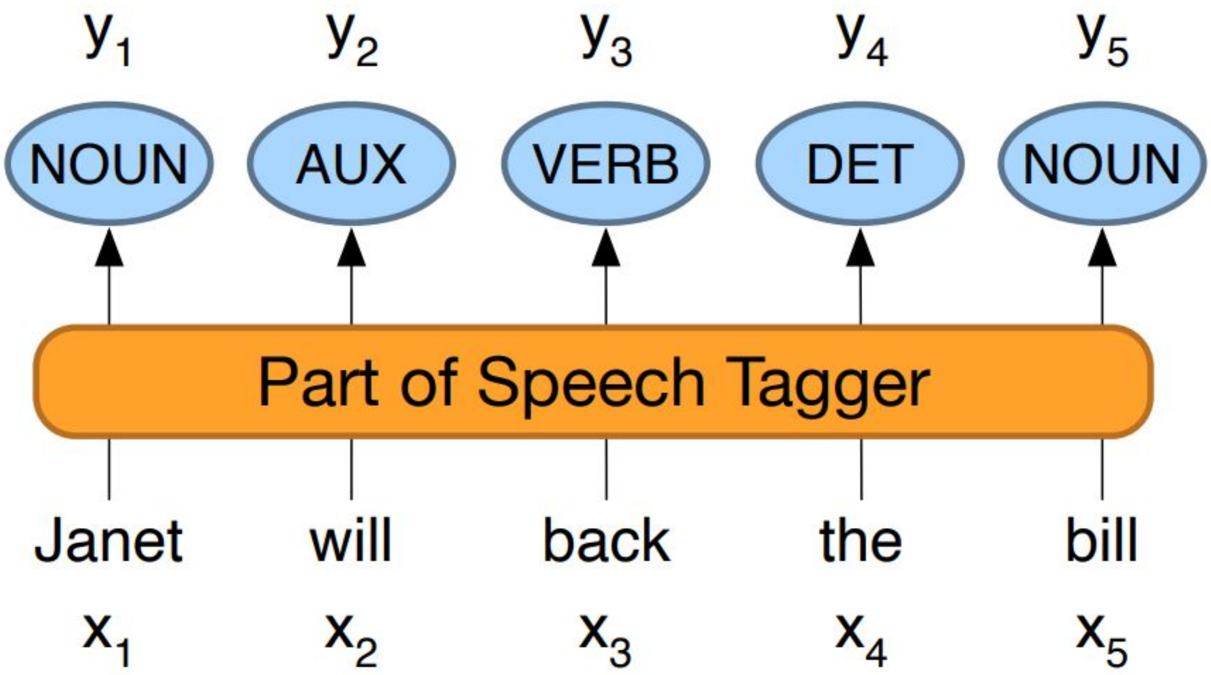
Words often have more than one POS.

book:

- VERB: (***Book*** *that flight*)
- NOUN: (*Hand me that* ***book***).

Part-of-Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



Standard algorithms for POS tagging

Supervised Machine Learning Algorithms:

- Hidden Markov Models
- Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Large Language Models (like BERT), finetuned

All required a hand-labeled training set, all about equal performance (97% on English)

All make use of information sources we discussed

- Via human created features: HMMs and CRFs
- Via representation learning: Neural LMs

1. Lexical Based Methods (Majority Wins)

For each word, it assigns the POS tag that most frequently occurs for that word in some training corpus which means will be wrongly tagged in some of the sentence. Also such a tagging approach cannot handle unknown/ambiguous words.

2. Rule-Based Methods (Follow the rules 👩‍🏫)

First assign the tag using the lexicon method and then apply predefined rules. The rules in Rule-based POS tagging are built manually. Some examples of rules are:

Change the tag to VBG for words ending with '-ing'

Changes the tag to VBD for words ending with '-ed'

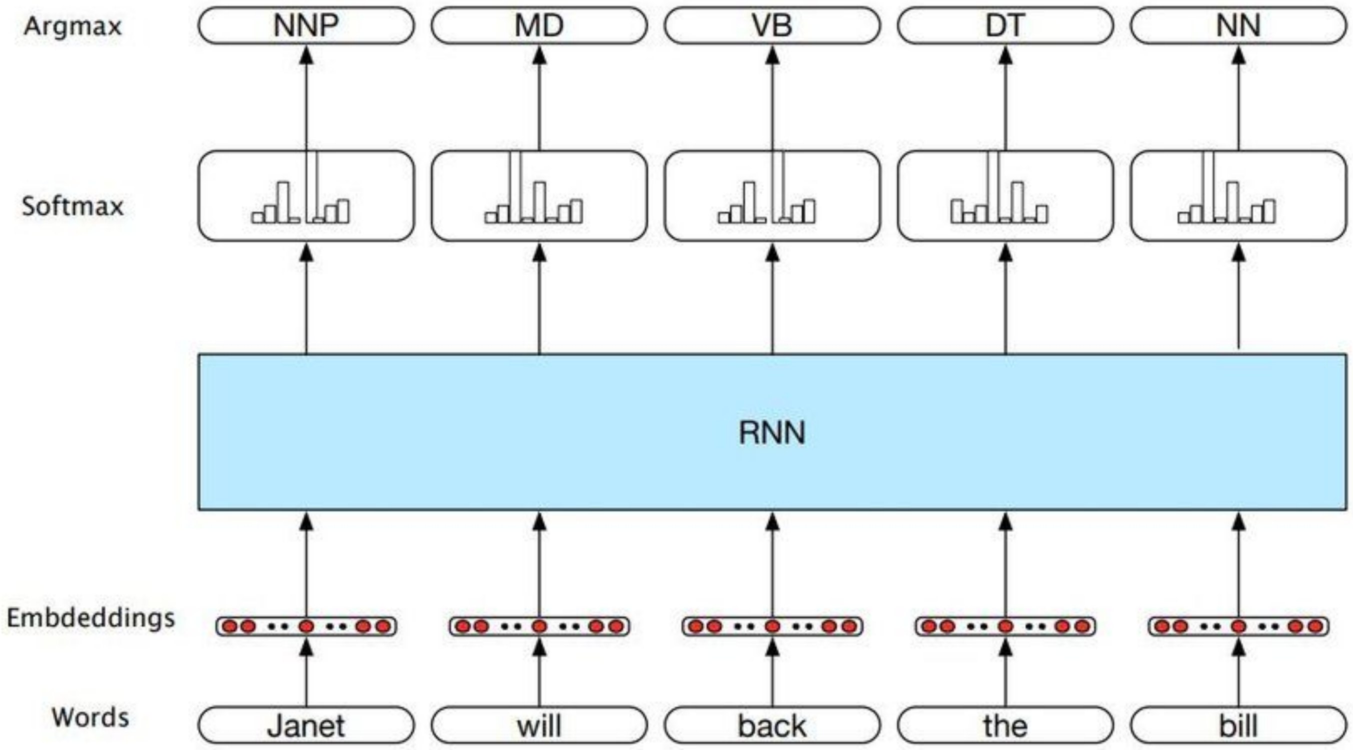
Replace VBD with VBN if the previous word is 'has/have/had'

3. Stochastic/Probabilistic Methods

Any model which somehow incorporates frequency or probability may be properly labelled stochastic. Its assign a PoS to a word based on the probability that a word belongs to a particular tag or based on the probability of a word being a tag based on a sequence of preceding/succeeding words. These are the preferred, most used and most successful methods so far.

Among these methods, there could be defined two types of automated Probabilistic methods: the Discriminative Probabilistic Classifiers (examples are Logistic Regression, SVM's and Conditional Random Fields – CRF's) and the Generative Probabilistic Classifiers (examples are Naive Bayes and Hidden Markov Models – HMM)

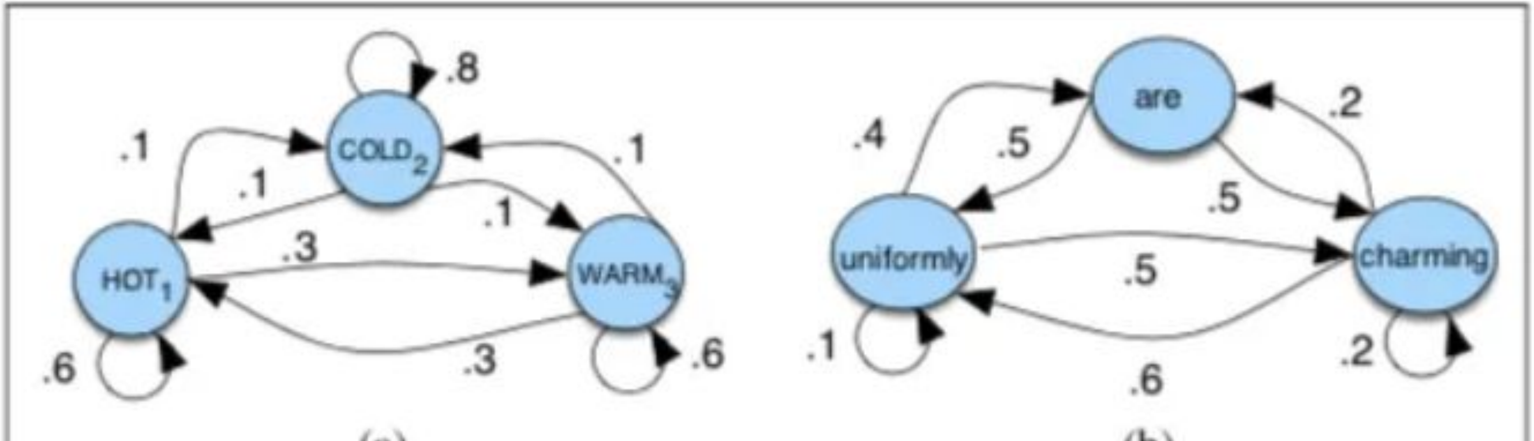
4. Deep Learning Methods – Recurrent Neural Networks



Markov Chain

A Markov chain is a model that tells us something about the probabilities of sequences of random states/variables. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. All the states before the current state have no impact on the future except via the current state.

Markov Chain



(a) (b)

HMM FOR POS-TAG

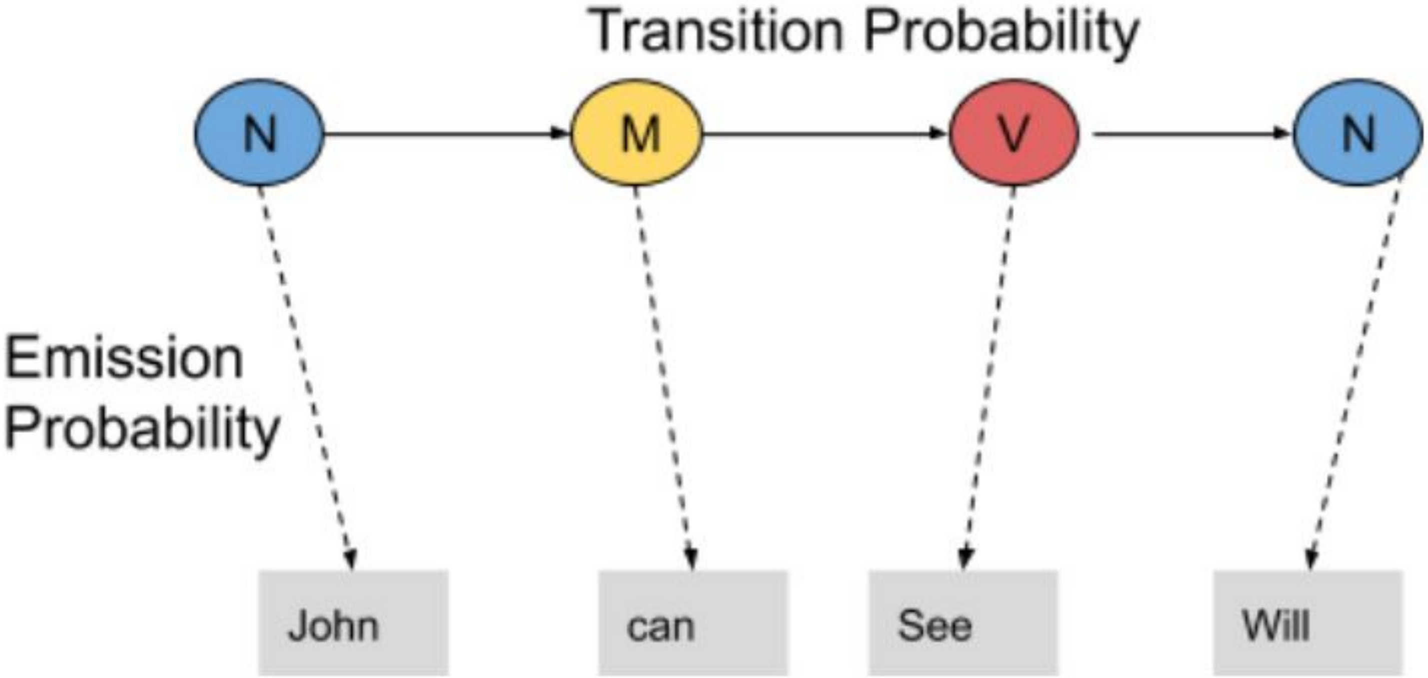
We can have the words in a sentence as Observable States (given to us in the data) but their POS Tags as Hidden states and hence we use HMM for estimating POS tags.

It must be noted that we call

Observable states ‘Observation’

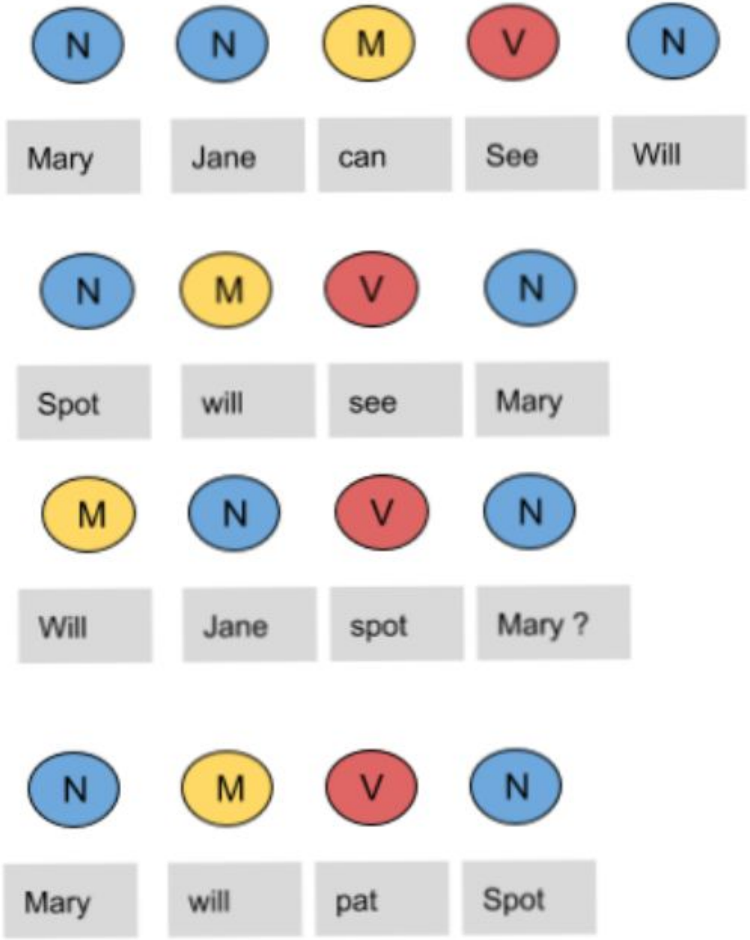
Hidden states ‘States’

Hidden Markov Model



Hidden Markov Model

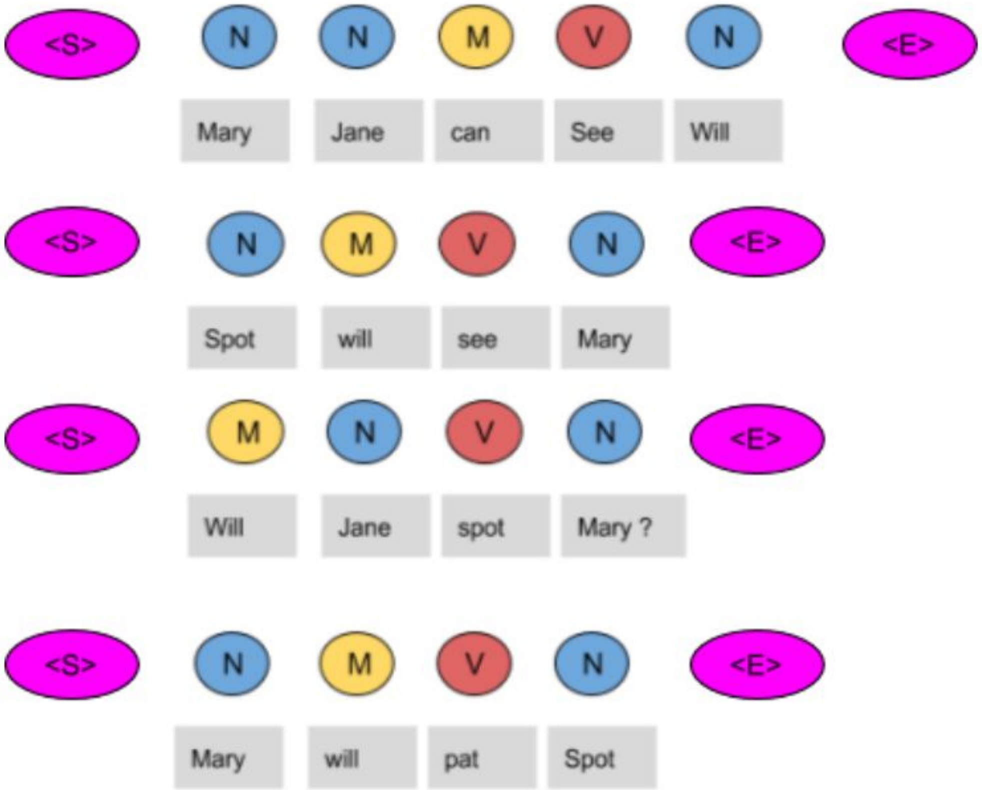
- Mary Jane can see Will
- Spot will see Mary
- Will Jane spot Mary?
- Mary will pat Spot



Hidden Markov Model - Emission Probabiliy

Words	Noun	Model	Verb
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	2/4
pat	0	0	1

Hidden Markov Model



Hidden Markov Model - Transition Probability

	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

Hidden Markov Model

1/4*3/4*3/4*0*1*2/9*1/9*4/9*4/9=0

Take a new sentence and tag them with wrong tags.

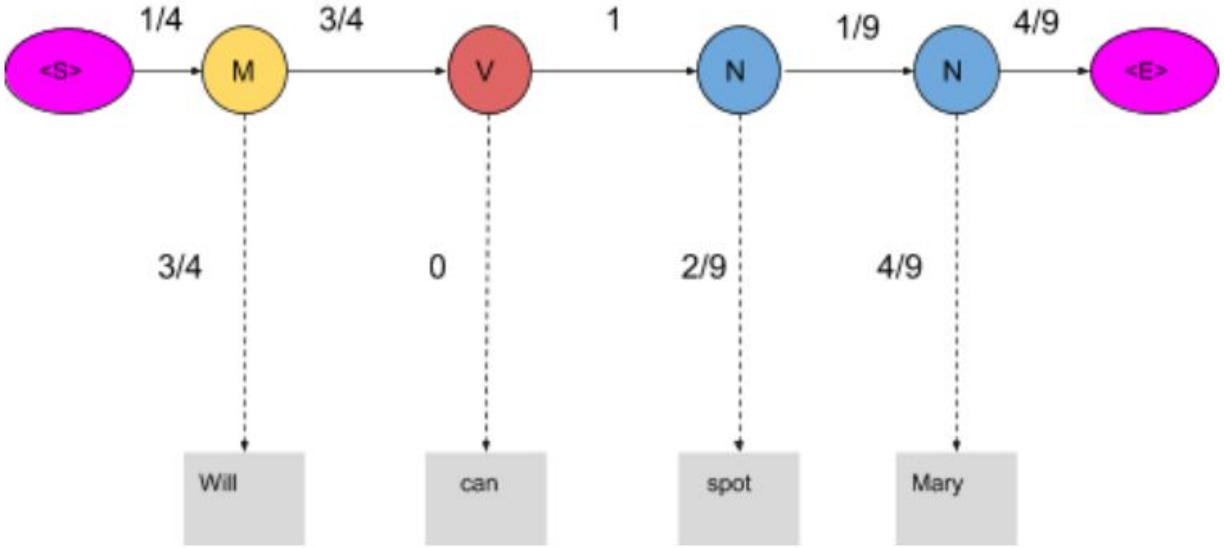
Let the sentence, ‘ Will can spot Mary’ be tagged as-

Will as a model

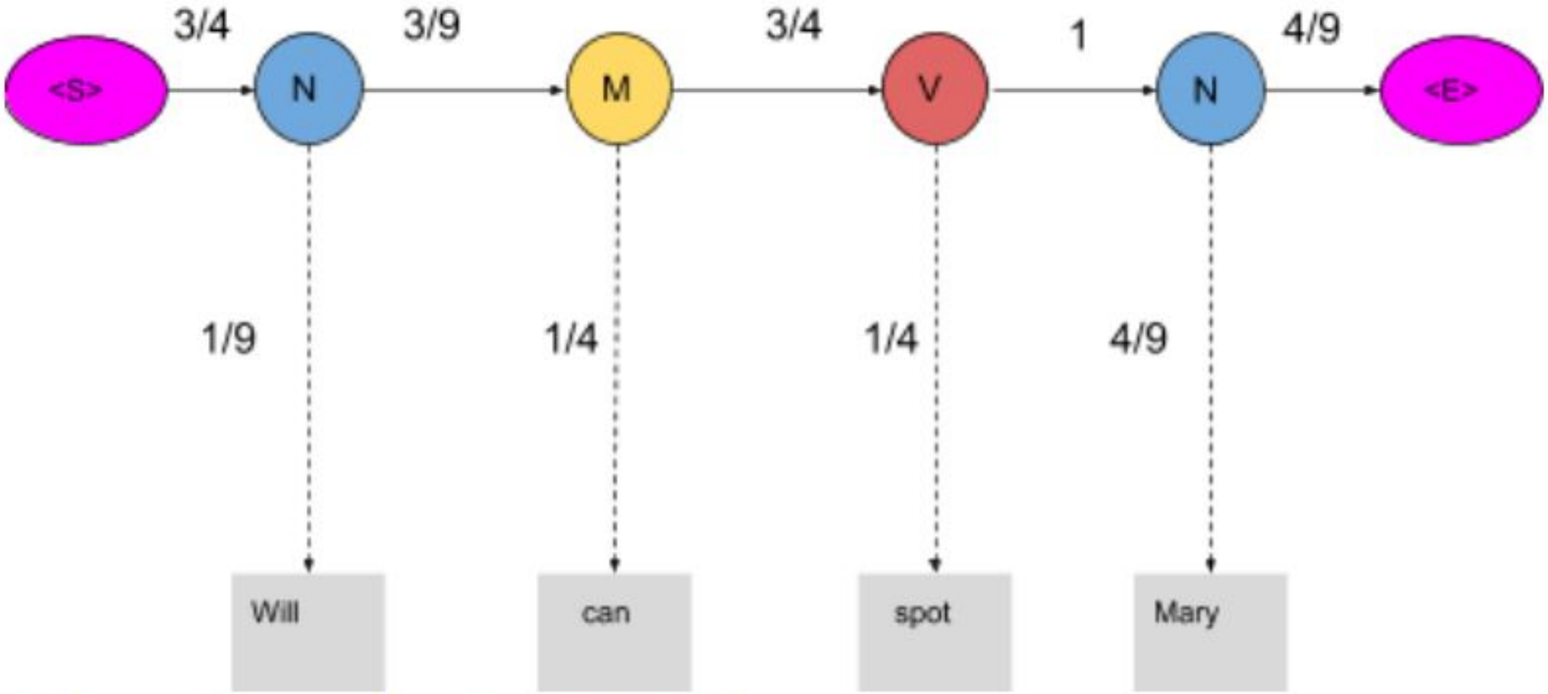
Can as a verb

Spot as a noun

Mary as a noun



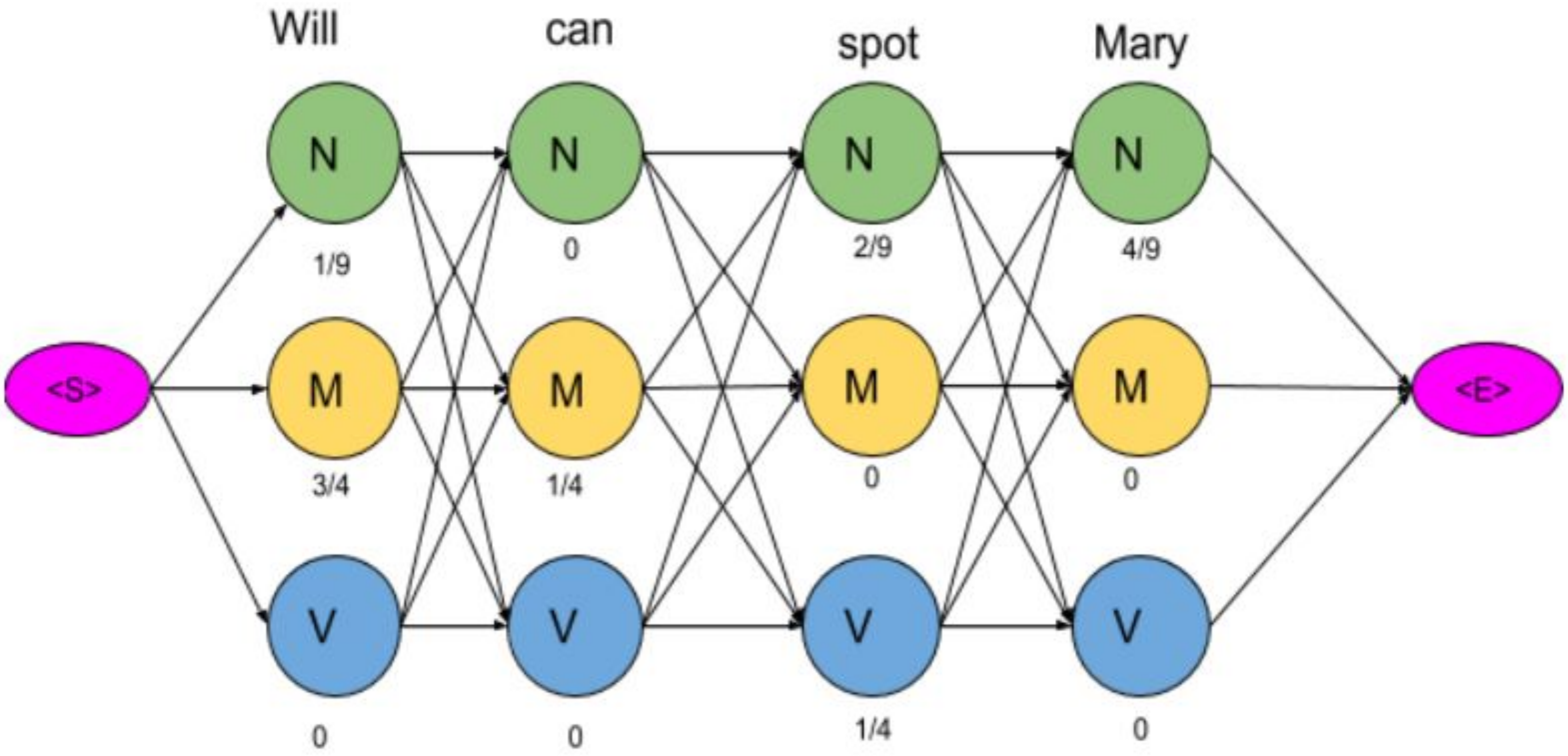
When these words are correctly tagged, we get a probability greater than zero as shown below

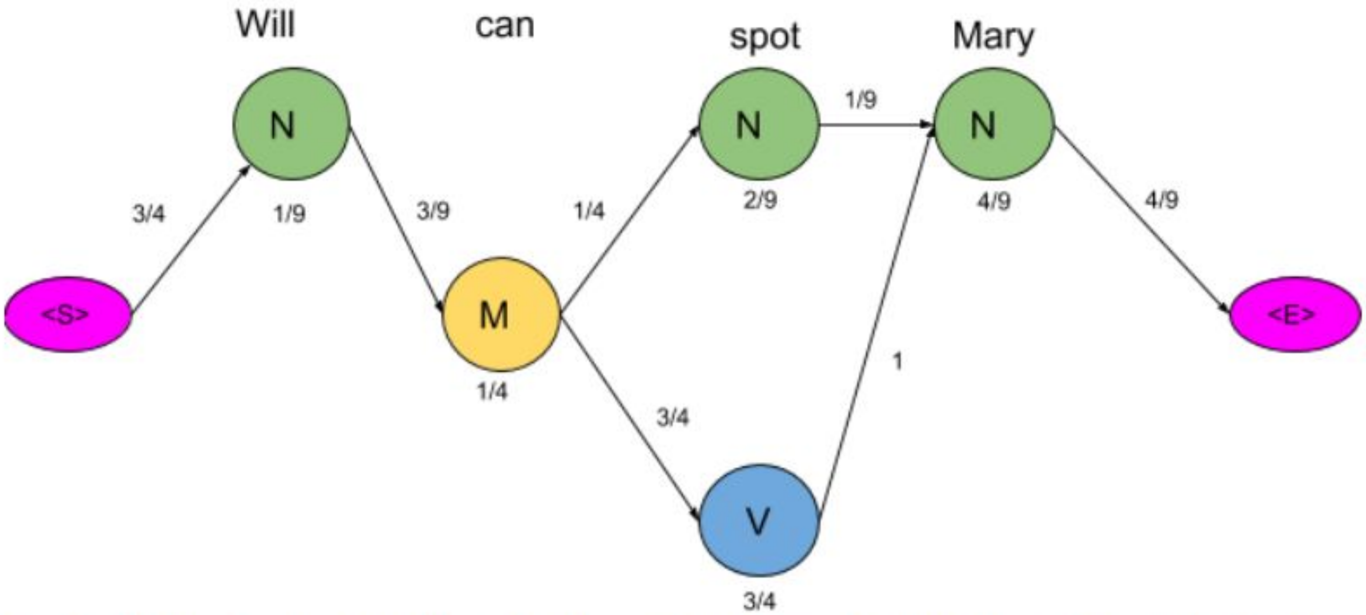


Calculating the product of these terms we get,

3/4*1/9*3/9*1/4*3/4*1/4*1*4/9*4/9=0.00025720164

Hidden Markov Model





Now there are only two paths that lead to the end, let us calculate the probability associated with each path.

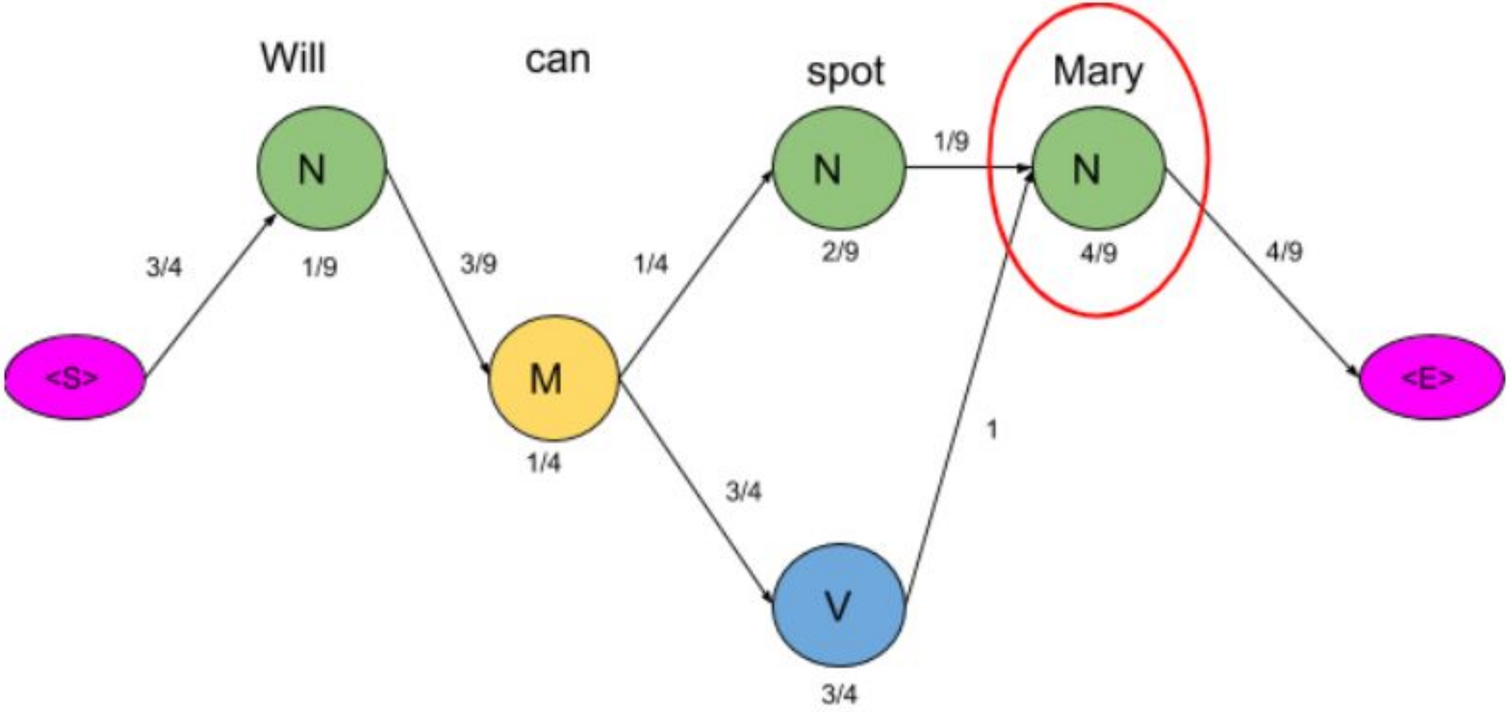
<S>→N→M→N→N→<E> =**3/4*1/9*3/9*1/4*1/4*2/9*1/9*4/9*4/9=0.00000846754**

<S>→N→M→N→V→<E>=**3/4*1/9*3/9*1/4*3/4*1/4*1*4/9*4/9=0.00025720164**

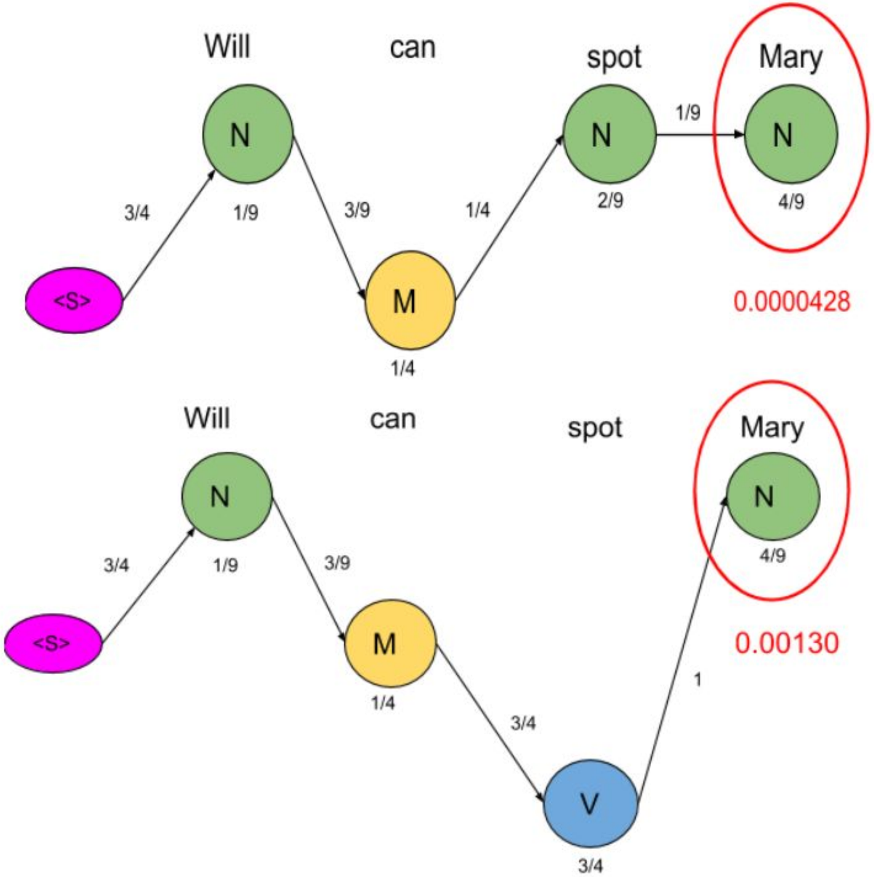
OPTIMIZING HMM WITH VITERBI ALGORITHM

> “
>
> The Viterbi algorithm is a dynamic programming algorithm 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 (HMM).

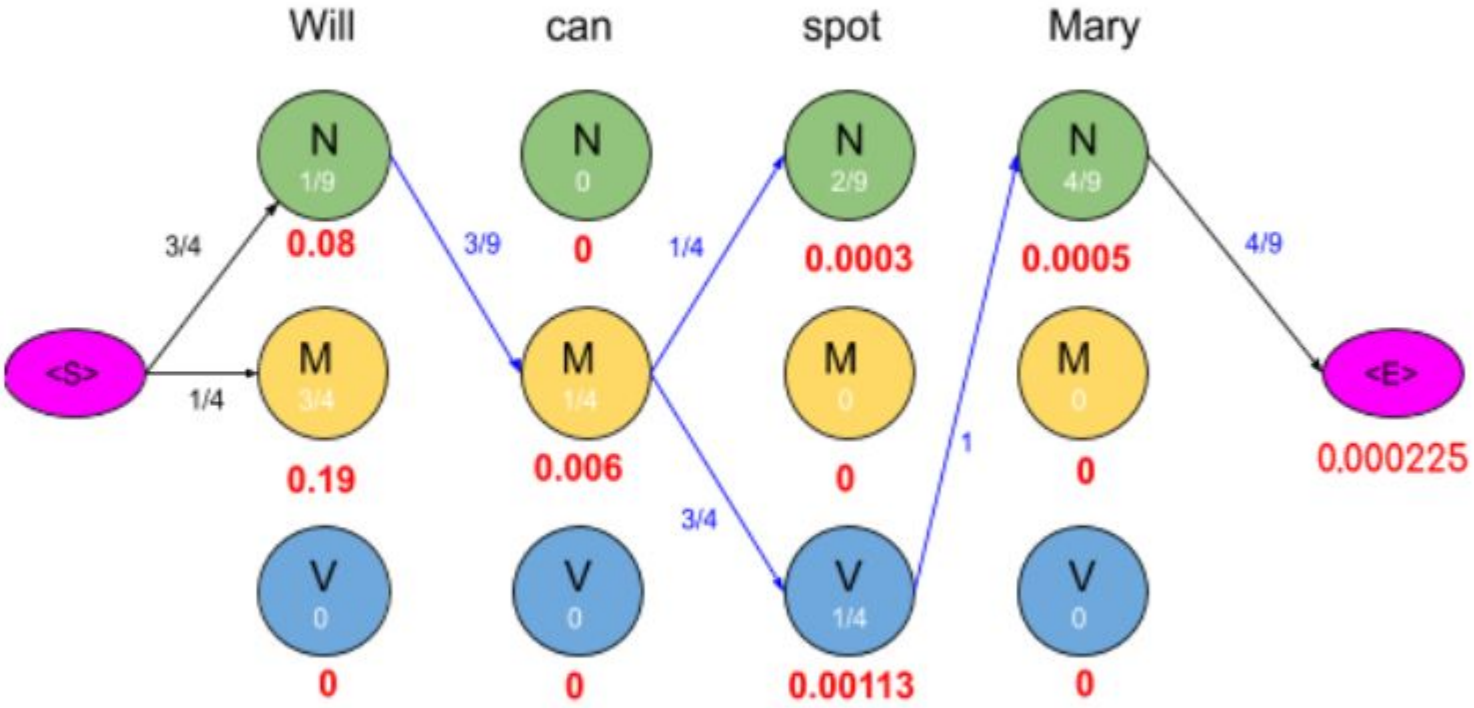
OPTIMIZING HMM WITH VITERBI ALGORITHM



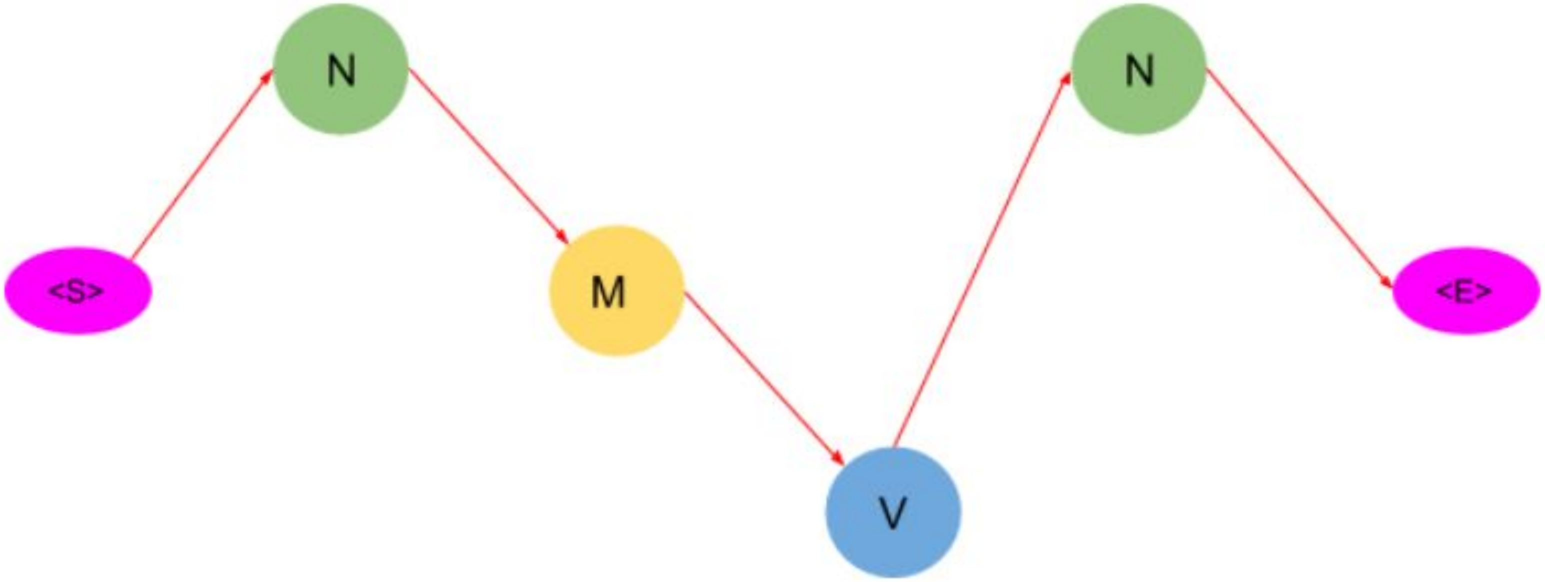
OPTIMIZING HMM WITH VITERBI ALGORITHM



OPTIMIZING HMM WITH VITERBI ALGORITHM



OPTIMIZING HMM WITH VITERBI ALGORITHM



function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*

for each state s **from** 1 **to** N **do** ; initialization step

$\quad viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$

$\quad backpointer[s,1] \leftarrow 0$

for each time step t **from** 2 **to** T **do** ; recursion step

$\quad$**for** each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname*{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$$; termination step

$$bestpathpointer \leftarrow \operatorname*{argmax}_{s=1}^{N} viterbi[s,T]$$; termination step

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

return *bestpath*, *bestpathprob*

NER
Named Entity Recognition

Named Entity Recognition (NER)

- The task: find and classify names in text, for example:

  Last night , Paris Hilton wowed in a sequin gown .

  PER PER

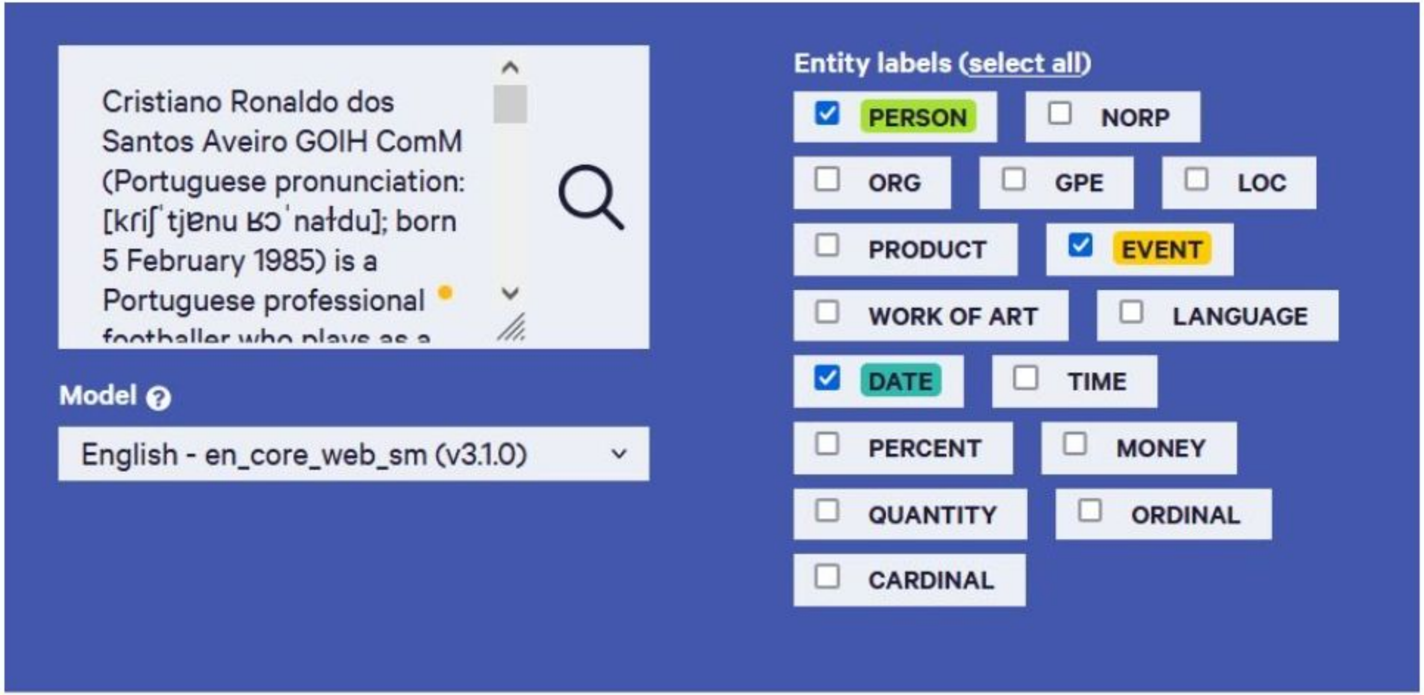
  Samuel Quinn was arrested in the Hilton Hotel in Paris in April 1989 .

  PER PER LOC LOC LOC DATE DATE

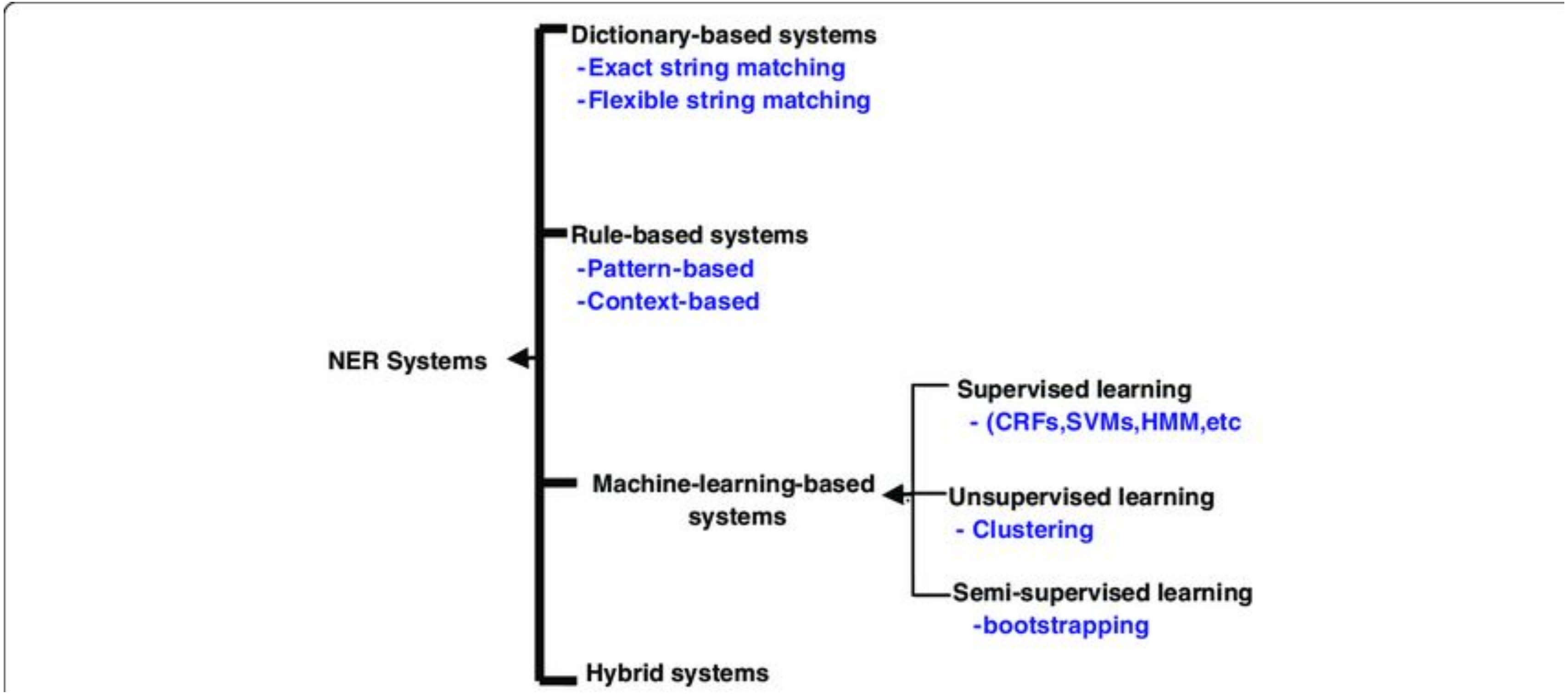
- Possible uses:
  - Tracking mentions of particular entities in documents
  - For question answering, answers are usually named entities
- Often followed by Named Entity Linking/Canonicalization into Knowledge Base

In simpler words, if your task is to find out ‘where’, ‘what’, ‘who’, ‘when’ from a sentence, NER is the solution you should opt for.

HTTPS://DEMOS.EXPLOSION.AI/DISPLACY-ENT



NER Methods



Dictionary-based systems
-Exact string matching
-Flexible string matching
Rule-based systems
-Pattern-based
-Context-based
NER Systems
Machine-learning-based systems
Supervised learning
- (CRFs,SVMs,HMM,etc
Unsupervised learning
- Clustering
Semi-supervised learning
-bootstrapping
Hybrid systems

Common Named Entity

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Golden Gate Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

Problems With NER

Name	Possible Categories
Washington	Person, Location, Political Entity, Organization, Vehicle
Downing St.	Location, Organization
IRA	Person, Organization, Monetary Instrument
Louis Vuitton	Person, Organization, Commercial Product

Below are some sentences using 'Washington' as different Named Entities.

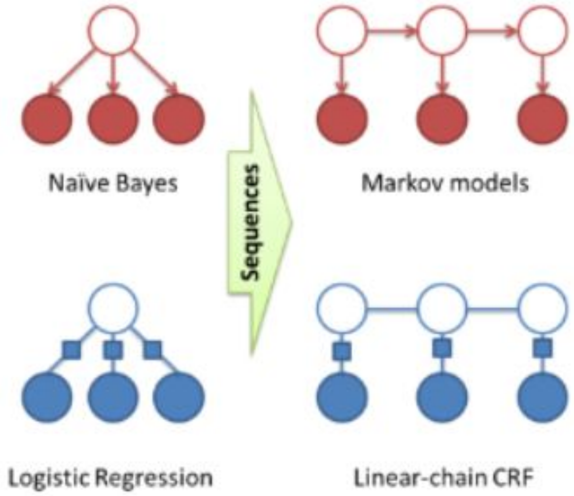
[$_{PER}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{ORG}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{LOC}$ Washington] for what may well be his last state visit.
In June, [$_{GPE}$ Washington] passed a primary seatbelt law.
The [$_{VEH}$ Washington] had proved to be a leaky ship, every passage I made...

Linear Chain Conditional Random Fields

CRF is amongst the most prominent approach used for NER.

A linear chain CRF confers to a labeler in which tag assignment(for present word, denoted as y_i) depends only on the tag of just one previous word(denoted by y_{i-1}).

	Generative *(Joint Probability)*	Discriminant *(Conditional Probability)*
Single Class	Naive Bayes	ME, Logistic Regression
Sequence	HMM	MEMM, CRF



Feature Function

embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
presence of hyphen

Feature Function

1. w_i: i_th word of a sentence

2. Embeddings refer to the numerical(vector) representation of a word. More can be explored here

3. Gazetteer: It is a list of places' names (India, Agra, etc) with their geographical & political information. It has millions of entries.

4. Word shape: It is a notation in which letters of a word are denoted in the following way:

Small letters: ‘x’
Capital letters: ‘X’
Digits: ‘d’
Punctuations & other symbols are untouched

Hence, if I get the word ‘Delhi%123%DD’, using Word shape, it can be transformed into ‘Xxxxx%ddd%XX’

Feature Function

5. Short word shape: Similar notation to Word shape with a slight change. Here, we would be removing consecutive similar type letters. ‘Delhi%123%DD’= ‘Xx%d%X’.

Every Feature Function intakes the below parameters:
Index of current word=’i’
Label of current word=’y_i’
Label of previous word=’y_i-1’
Sentence=’x’
Consider, ‘Ram is cool’
with Named Entity Labels as [PER O O] where we have
Ram: PER, is:O, cool:O

Feature Function

Consider a **Feature function(F_j(x, y, y-1, i))** with the definition:

The i-th word in ‘x’ is capitalized return 1 else 0

If i=2 (considering indexing from 1 & not 0), hence we are calculating the feature for ‘is’, the above feature function is demonstrated below:

F_j(‘Ram is cool’, ‘O’, ‘PER’, 2): return 0 as ‘is’ isn’t capitalized.

The suffix ‘j’ refers to the j^{th} feature function where j goes from 1 →total feature functions

Linear Chain Conditional Random Fields

pθ (y|x) refers to the probability of calculating a Label sequence(y) given a word sequence(x).

CRF:

$$p_\theta(y|x) = \frac{exp(\sum_j w_j F_j(x,y))}{\sum_{y'} exp(\sum_j w_j F_j(x,y'))}$$

, where $F_j(x,y) = \sum_{i=1}^{L} f_j(y_{i-1}, y_i, x, i)$

Length of sequence x

Sum over all feature function

Weight for given feature function

$$P(\overline{y}|\overline{x};w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \overline{x}, i))}$$

Feature Functions

Sum over all possible label sequence

Feature function can access all of observation

CRF

The outer summation goes from i=1 to i=length of sentence 'L'. Hence we are summating the value of any feature function for all words of the sentence

if we have a sentence 'Ram is cool', the outer summation will add values of the output of the j^{th} feature function for all 3 words of the sentence

CRF

The inner summation goes from j=1 to the total number of feature functions.

It is doing something like this

$W_1 * \Sigma feature\_function_1 + W_2 * \Sigma feature\_function_2 \ldots\ldots$

W□ refers to weights assigned to a feature_function□.

CRF

The denominator is referred to as a Normalizing constant.

To calculate the P([PER, PER, LOC] | ‘Ram is cool’)=

Numerator=exp (Σ□ w□ Σ $_i$ F□(‘Ram is cool’,’PER PER LOC’))

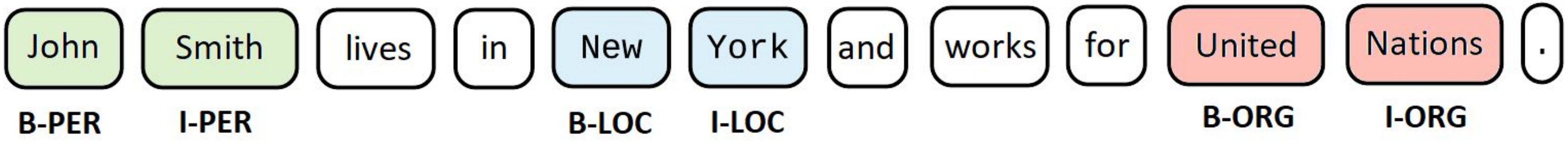
Denominator=exp (Σ□ w□ Σ $_i$ F□(‘Ram is cool’,’O O O’))+exp (Σ□ w□ Σ $_i$ F□(‘Ram is cool’,’ VEH ORG O’))+exp (Σ□ w□ Σ $_i$ F□(‘Ram is cool’,’PER ORG ORG’))….

Are We Done?

$$\frac{\partial \log p_\theta(y|x)}{\partial w_j} = F_j(x,y) - \sum_{y'} F_j(x,y')p_\theta(y'|x)$$

$$w_j \leftarrow w_j + \alpha \left(F_j(x,y) - \sum_{y'} F_j(x,y')p_\theta(y'|x) \right)$$

IOB tagging


John
Smith
lives
in
New
York
and
works
for
United
Nations
.
B-PER
I-PER
B-LOC
I-LOC
B-ORG
I-ORG

Lab Tasks

https://github.com/susanli2016/NLP-with-Python/blob/master/NER_NLTK_Spacy.ipynb

https://towardsdatascience.com/named-entity-recognition-and-classification-with-scikit-learn-f05372f07ba2

References

https://www.mygreatlearning.com/blog/pos-tagging/

https://medium.com/data-science-in-your-pocket/pos-tagging-using-hidden-markov-models-hmm-viterbi-algorithm-in-nlp-mathematics-explained-d43ca89347c4

https://medium.com/data-science-in-your-pocket/named-entity-recognition-ner-using-conditional-random-fields-in-nlp-3660df22e95c