NAMED ENTITY RECOGNITION (NER)

Gabriella Pasi
gabriella.pasi@unimib.it

Named Entity Recognition (NER)

A very important sub-task of Information Extraction, aimed at finding and classifying names in texts, for example:

The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation

Named Entity Recognition

- *Named entity recognition*
  - identify words that refer to *names* of interest in a particular application
  - e.g., people, companies, locations, product names

- *Text is mapped to the identified names*
  - Task: what is the most likely mapping

Named Entity Recognition

- Example :

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Organization, Location, Person

Linguistically difficult problem

- NER involves ***identification*** of *proper names* in texts, and ***classification*** into a set of predefined categories of interest.
- Three universally accepted categories: **person**, **location** and **organisation.**
- Other common tasks: recognition of **date/time expressions**, **measures** (percent, money, weight etc), email addresses etc.
- Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc.

Applications of NER

- Yellow pages with local search capabilities
- Monitoring trends and sentiment in textual social media
- Interactions between genes and cells in biology and genetics

Problems in NER task definition

- Category definitions are intuitively quite clear, but there are many grey areas.

- Many of these grey areas are caused by **metonymy**.
  - Organisation vs. Location : “**England** won the World Cup” vs. “The World Cup took place in **England**”.
  - Company vs. Artefact: “shares in **MTV**” vs. “watching **MTV**”
  - Location vs. Organisation: “she met him at **Heathrow**” vs. “the **Heathrow** authorities”

Named Entity Recognition

1) Rule-based

- Uses *lexicons* (lists of words and phrases) that categorize names
    - e.g., locations, peoples' names, organizations, etc.

- Rules also used to verify or find new entity names
    - e.g., "<number> <word> street" for addresses
    - "<street address>, <city>" or "in <city>" to verify city names
    - "<street address>, <city>, <state>" to find new cities
    - "<title> <name>" to find new names

- Rules either developed manually by trial and error or by using machine learning techniques

- Language dependent

Named Entity Recognition

- 2) *Statistical machine learning*
  - uses a probabilistic model of the words in and around an entity
  - probabilities estimated using *training data* (manually annotated text)
  - Hidden Markov Model (HMM) is one approach (other approaches Conditional Random Fields, Support Vector Machines…)

Hidden Markov model for NER

- Resolve ambiguity in a word using *context*
  - e.g., “marathon” is a location or a sporting event, “boston marathon” is a specific sporting event

- Model context using a *generative* model of the sequence of words

HMM for Extraction

- The HMM is based on augmenting the Markov chain

- *Markov Model (Markov Chain)* describes a process as a sequence of states (random variables) with transitions between them.
  - each transition has a probability associated with it
  - next state depends only on current state and transition probabilities

Markov Assumption: $P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_i - 1)$

HMM for Extraction

- The HMM is based on augmenting the Markov chain

- *Markov Model (Markov Chain)*

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{N} \pi_i = 1$

Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2021. All rights reserved. Draft of December 29, 2021.

HMM for Extraction

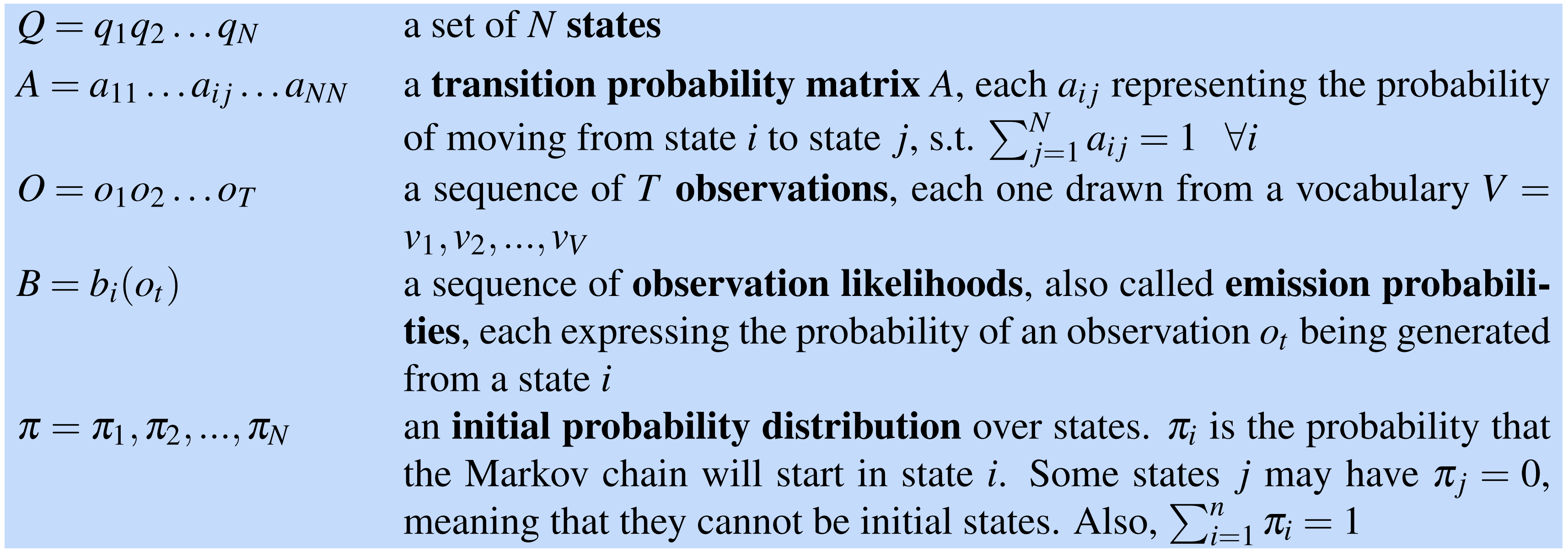
- The HMM is based on augmenting the Markov chain

Markov chain useful to compute a probability for a sequence of ***observable events***.

Often events we are interested in are hidden: we do not observe them directly. For example, in a text we do not normally observe part-of-speech tags; we see words and must infer the tags from the word sequence.

We call the tags ***hidden*** because they are not observed.

HMM for Extraction



$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

As with a first-order Markov chain, the probability of a particular state depends only on the previous state.

The probability of an output observation *oi* depends only on the state that produced the observation *qi* and not on any other states or any other observations.

HMM for Extraction

- To recognize named entities, find sequence of “labels” that give highest probability for the sentence
  - only the outputs (words) are visible or observed
  - states are “hidden”
  - E.g
    - Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long time collector of tropical fish.
    - <start><name><not-an-entity><location><not-an-entity><end>

- *Viterbi* algorithm implements a Markov tagging process and can be used for recognition

Named Entity Recognition

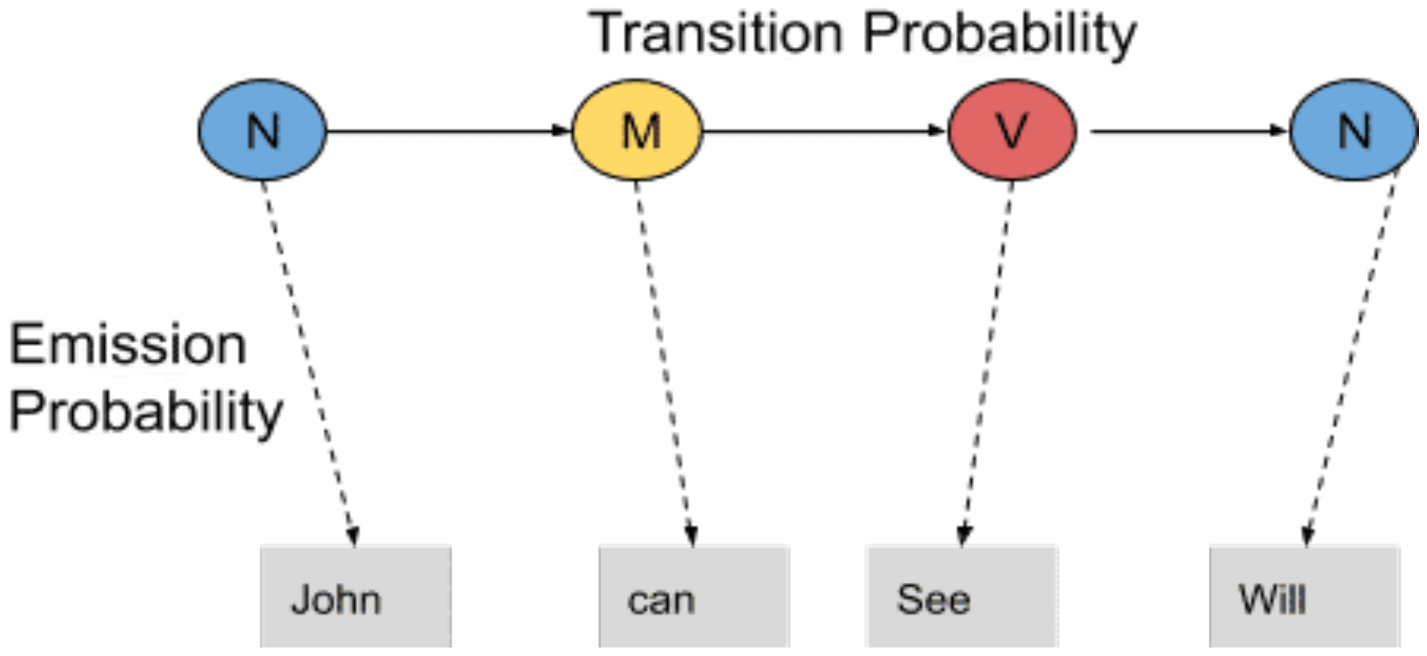
- Accurate recognition requires about 1M words of training data (1,500 news stories)
  - may be more expensive than developing rules for some applications

- Both rule-based and statistical can achieve about 90% effectiveness for categories such as names, locations, organizations
  - others, such as product name, can be much worse

Open source sw for POS and NER:
https://nlp.stanford.edu/software/

CORENLP: https://corenlp.run/

HMM-EXAMPLE



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

Transition probability

- The transition probability is the likelihood of a particular sequence for example, how likely is that a noun is followed by a model and a model by a verb and a verb by a noun. This probability is known as Transition probability. It should be high for a particular sequence to be correct.

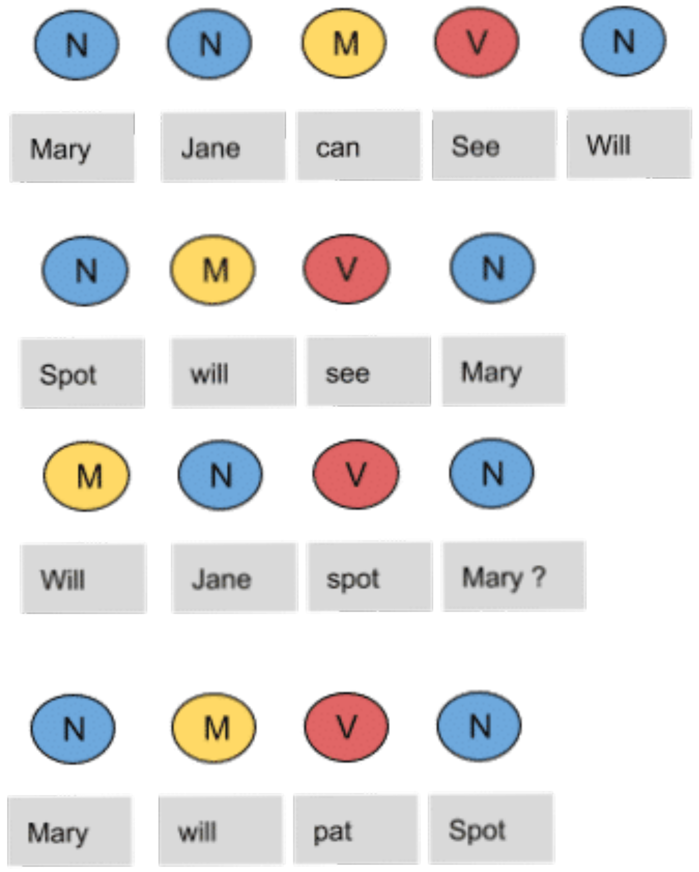
Emission Probability

- Ted will spot Will
- Now, what is the probability that the word Ted is a noun, will is a model, spot is a verb and Will is a noun. These sets of probabilities are Emission probabilities and should be high for our tagging to be likely.

Training Data

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

Example


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

count

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

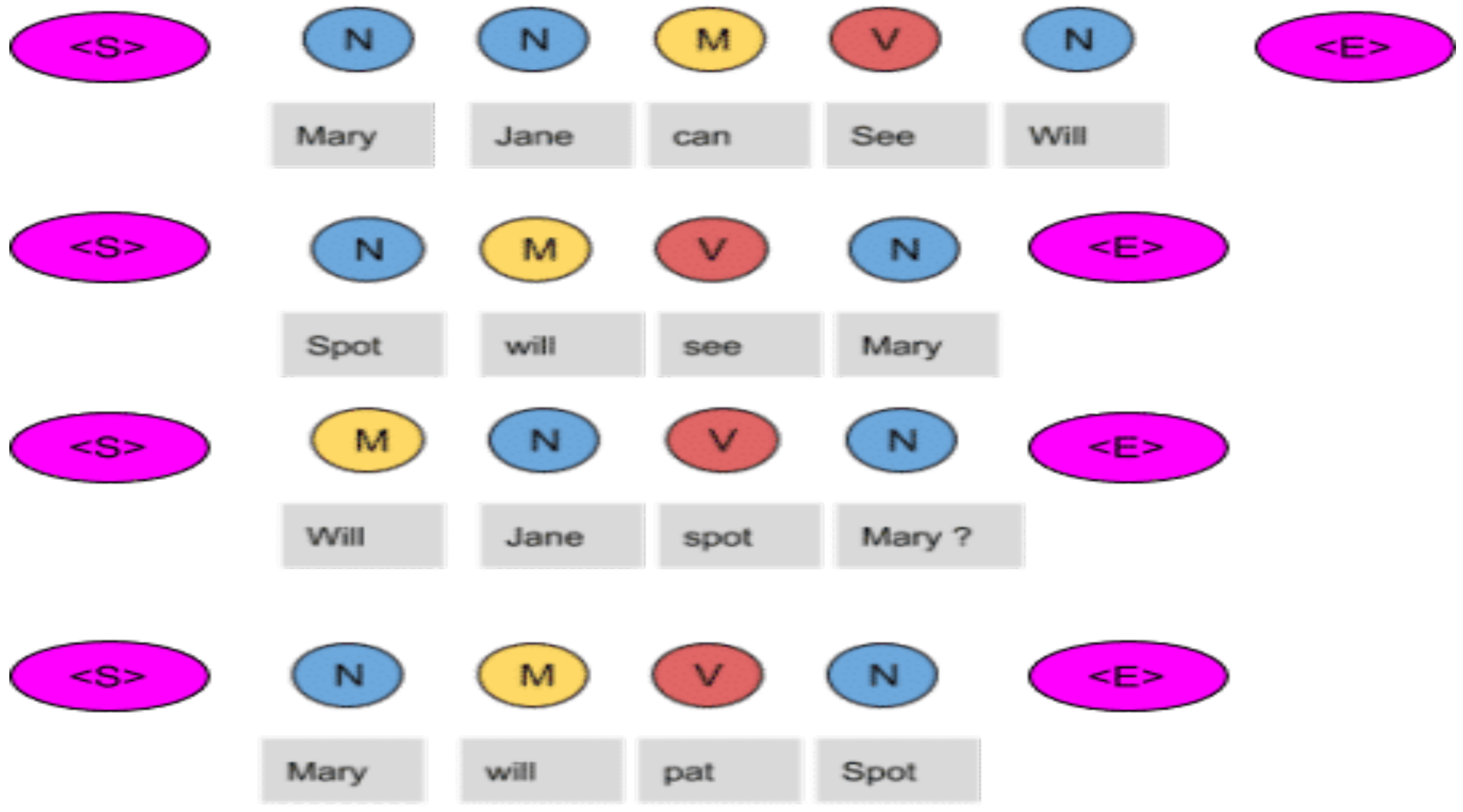
Probability

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

Emission probability

- The probability that Mary is Noun = 4/9
- The probability that Mary is Model = 0
- The probability that Will is Noun = 1/9
- The probability that Will is Model = 3/4

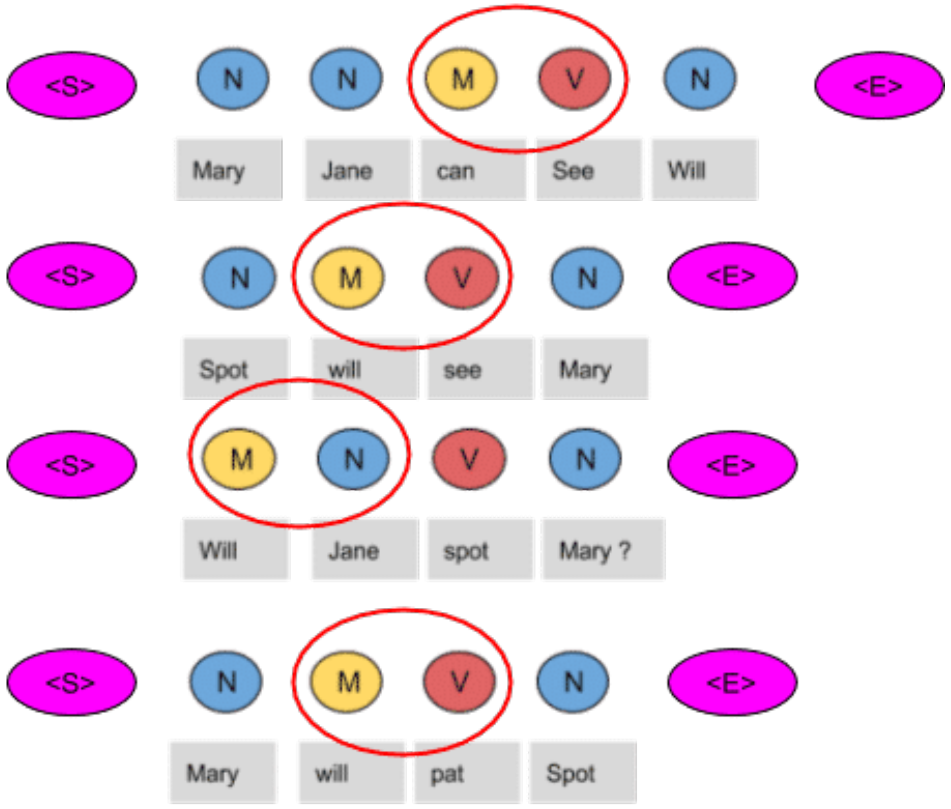
Start and end tag



COUNT

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

In the above figure, we can see that the <S> tag is followed by the N tag three times, thus the first entry is 3.The model tag follows the <S> just once, thus the second entry is 1. In a similar manner, the rest of the table is filled.



<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 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

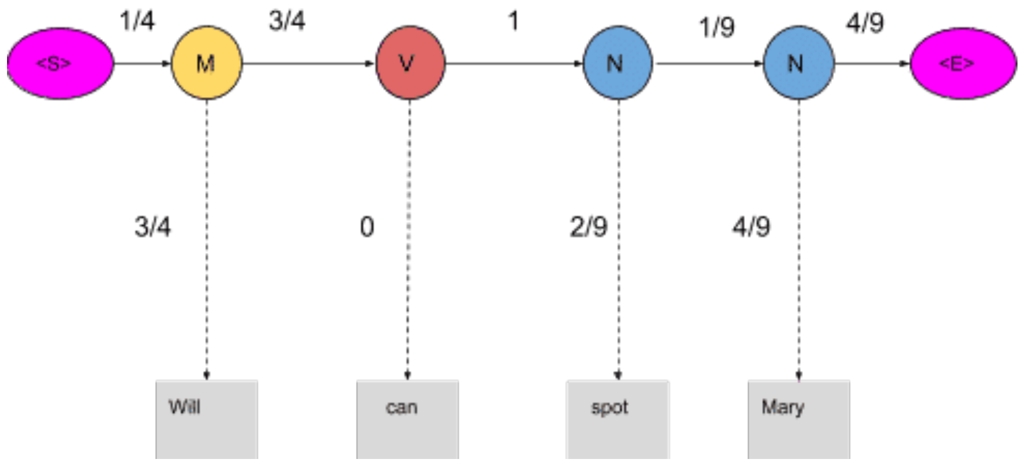
HMM

- how does the HMM determine the appropriate sequence of tags for a particular sentence from the above tables? Let us find it out.

Test data

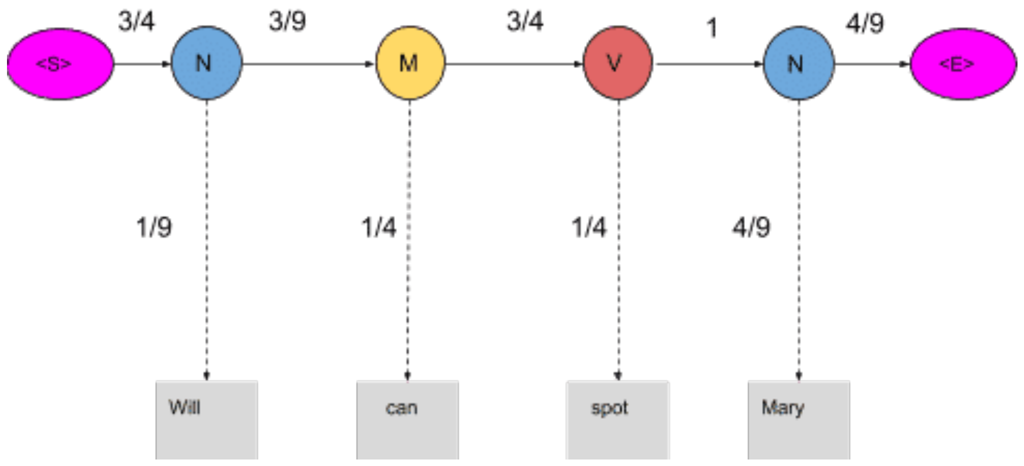
- Take a new sentence and tag them with wrong tags. Let the sentence, ‘ Will can spot Mary’ be tagged as-
- Will as a model
- Can as a verb
- Spot as a noun
- Mary as a noun

Test example



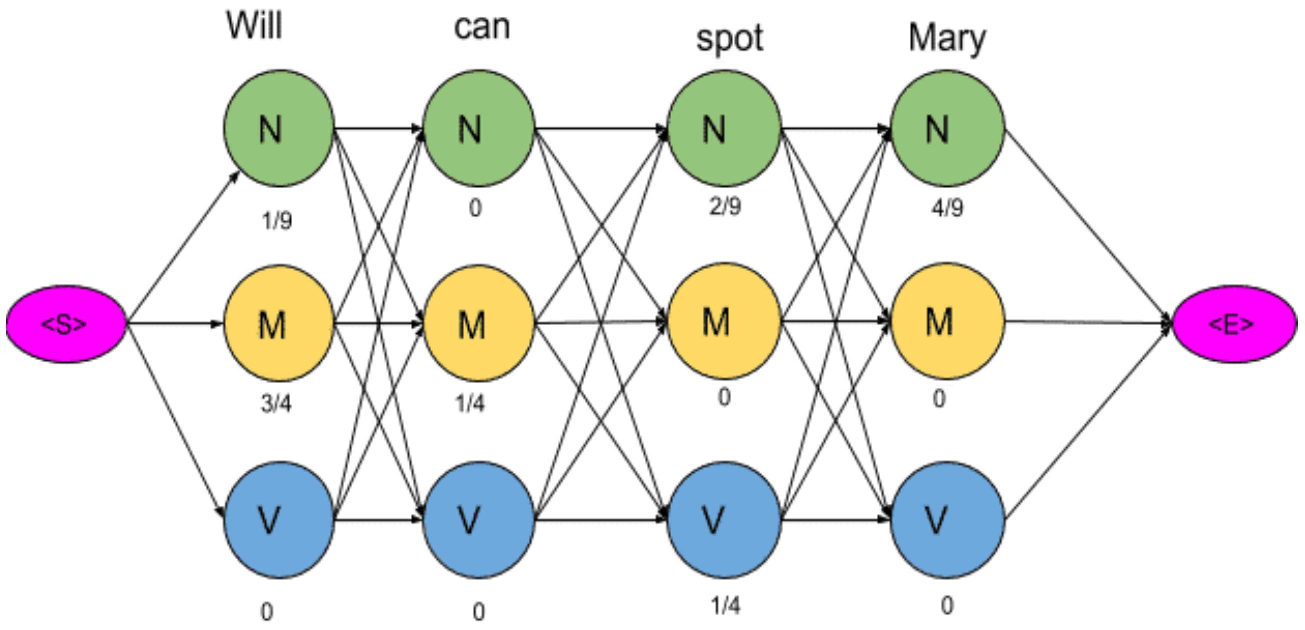
1/4*3/4*3/4*0*1*2/9*1/9*4/9*4/9=0

Test example

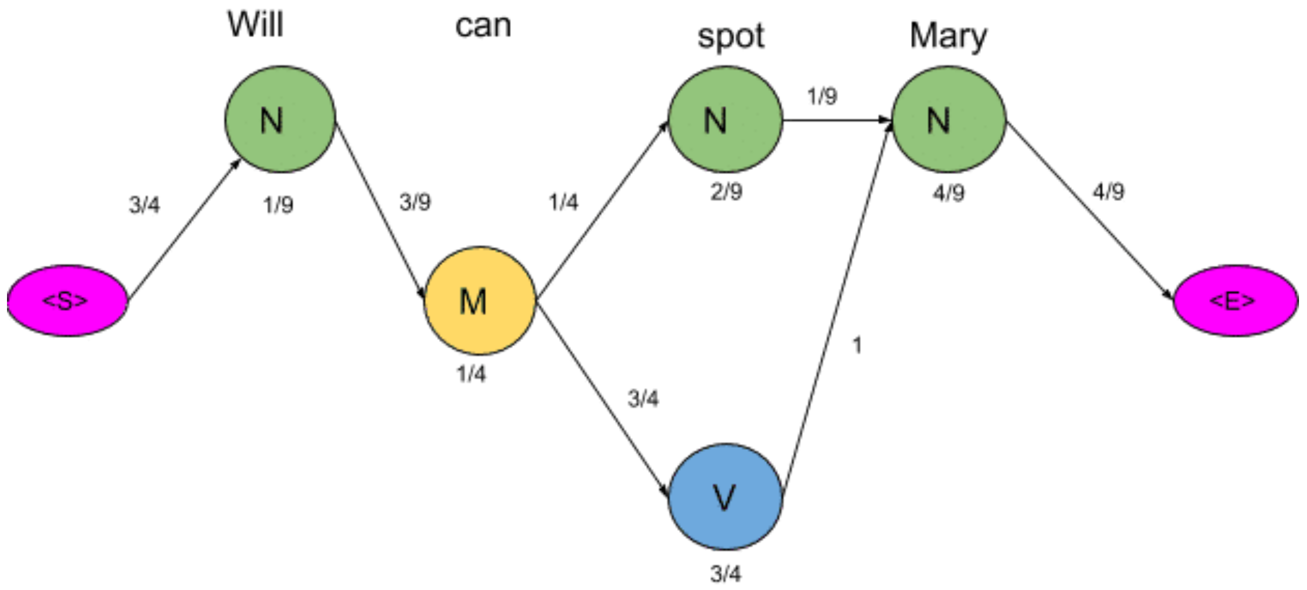


3/4*1/9*3/9*1/4*3/4*1/4*1*4/9*4/9=0.000 25720164

Solution



Test example



The next step is to delete all the vertices and edges with probability zero, also the vertices which do not lead to the endpoint are removed.

Test example

- 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**