Hidden Markov Model (HMM) Part-of-Speech Tagging

IIT – CS481 – Spring

2021.02.18

ILLINOIS INSTITUTE OF TECHNOLOGY

Review

Contents

POS tagging techniques

- Rule-based: Regular Expression Tagger

➢ **Probabilistic tagging:**
  - Default Tagger
  - N-gram Tagger
  - **HMM tagger**

➢ **Transformation-based:**
  - pre-defined rules
  - automatically induced rule
  - Brill tagger

- Deep learning models:
  - Meta-BiLSTM

Probability Tagging

1) Data:
   tagged corpus

2) Train a tagger:
   create a lookup table to store the word and tag information

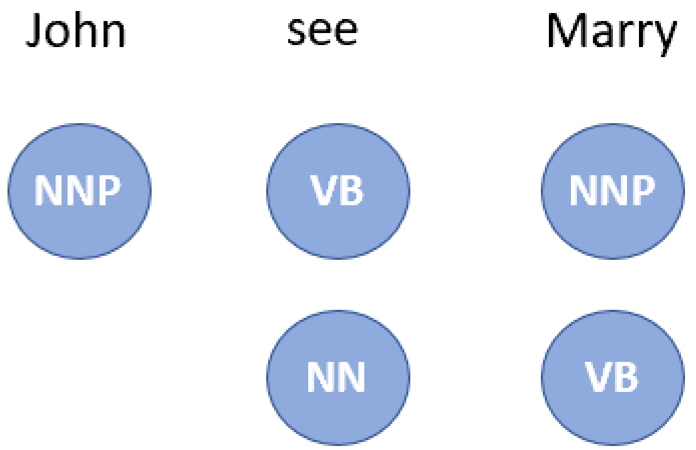
3) Prediction: tag new sentences
   i.e., tag sentences not seen in the training data

4) Evaluation:
   train test split
   tags assigned by human expert as gold standard

HMM POS tagging

Observation: a sequence of words w_1^n

Goal: assign a sequence of POS tags t_1^n



$$\boxed{\hat{t}_1^n} = \underset{t_1^n}{\boxed{\operatorname{argmax}}} P(t_1^n | w_1^n)$$

- argmax: the x that maximize f(x)
- Hat ^ notation: our estimate of the correct tag sequence

Hidden Markov Model

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{argmax} \frac{P(w_1^n \mid t_1^n)\, P(t_1^n)}{P(w_1^n)}$$

Drop denominator $P(w_1^n)$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \; \overbrace{P(t_1^n)}^{\text{prior}}$$

HMM tagger: 2 assumptions

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

The probability of a word appearing only depends on its own POS tag

Bigram assumption: a tag appearing only depends on the previous tag, rather than the entire tag sequence

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \qquad P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{argmax}\, P(t_1^n|w_1^n) \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

HMM: 2 probabilities

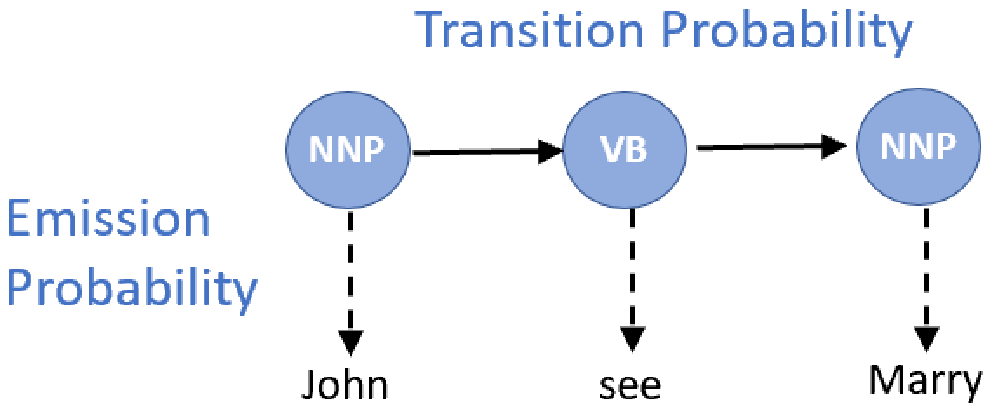
$$\hat{t}_1^n = \underset{t_1^n}{argmax}\, P(t_1^n | w_1^n) \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(t_i \mid t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(see|VB) = \frac{C(VB, see)}{C(VB)} = 0.057$$

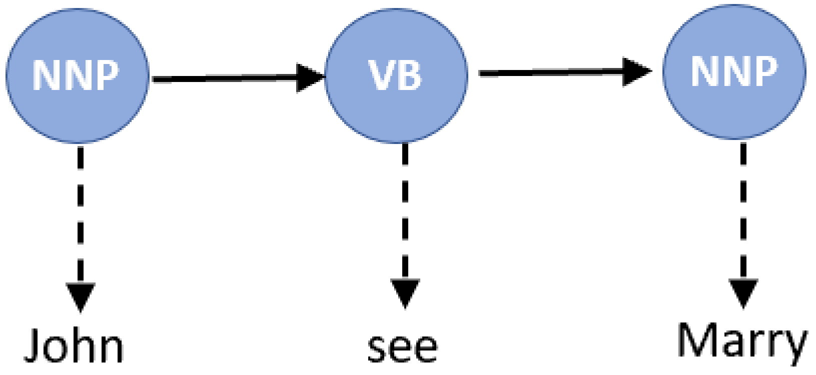
$$P(VB|NNP) = \frac{C(NNP, VB)}{C(NNP)} = 0.49$$



Formalizing HMM tagger

Transition Probability $P(t_i \mid t_{i-1})$

Emission Probability $P(w_i \mid t_i)$



Hidden states: a sequence of POS tags

Observation: a sequence of words

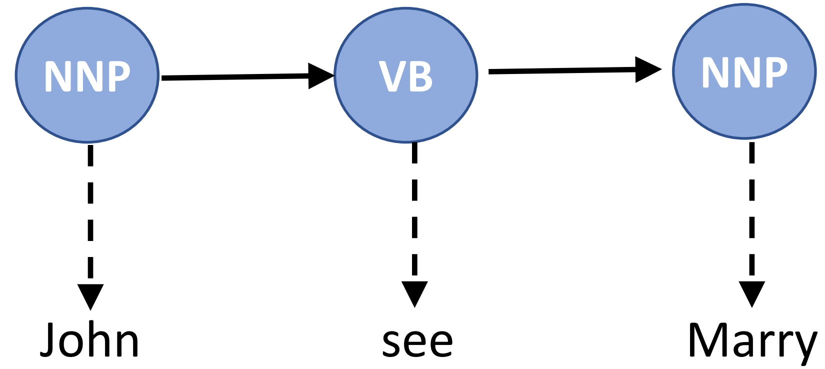
Emission Probability: given a particular tag, how likely is a word generated from this particular tag?

- what is the probability that:
  - John is a NNP
  - see is a VB
  - Marry is a NNP

Transition Probability: the probability of a POS tag followed by another tag

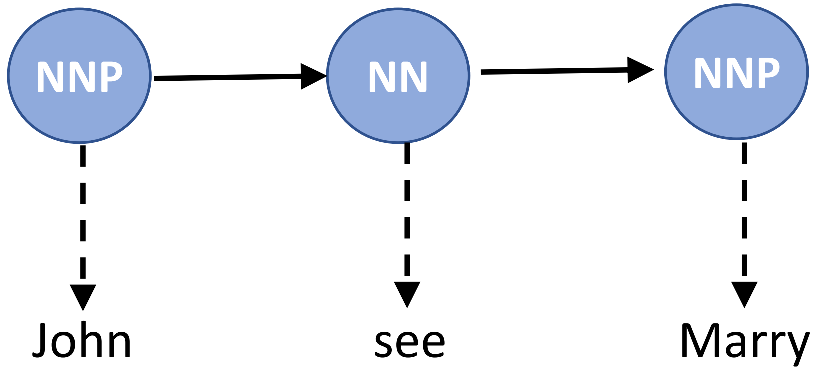
- how likely is that:
  - a NNP is followed by a VB
  - a VB is followed by a NNP

Computing the most likely sequence



$P(see|VB)$ = 0.057
$P(NNP|VB)$ = 0.27
$P(VB|NNP)$ = 0.49

$P(NNP) * P(John|NNP) *$
$P(VB|NNP) * P(see|VB) * P(NNP|VB) * P(Marry|NNP)$ = 0.0075411

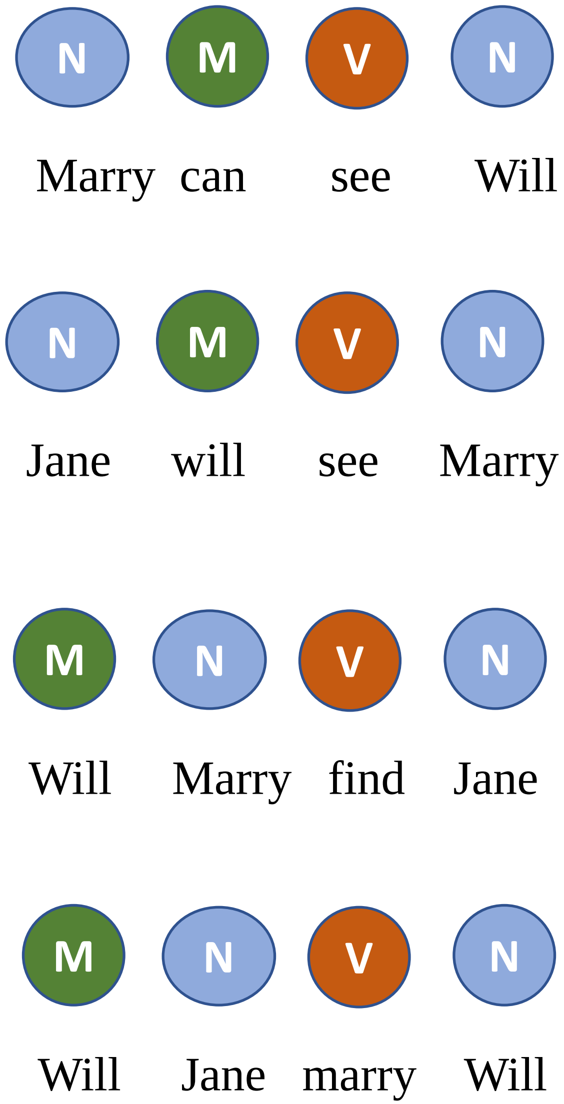


$P(see|NN)$ = 0.0012
$P(NN|NNP)$ = 0.047
$P(NNP|NN)$ = 0.001

$P(NNP) * P(John|NNP) *$
$P(NN|NNP) * P(see|NN) * P(NNP|NN) * P(Marry|NNP)$ = 0.000000564

Data

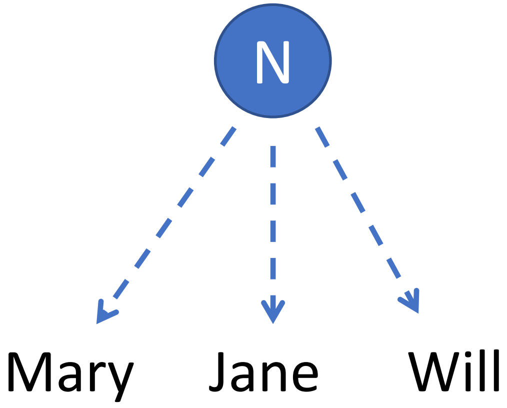
- Marry can see Will
- Jane will see Marry
- Will Marry find Jane?
- Will Jane marry Will?



	Noun	Verb	Model
marry	3	1	0
jane	3	0	0
will	2	0	3
can	0	0	1
see	0	2	0
find	0	1	0

Train a tagger: learn emission probability

Given a particular tag, how likely is a word generated from this particular tag?

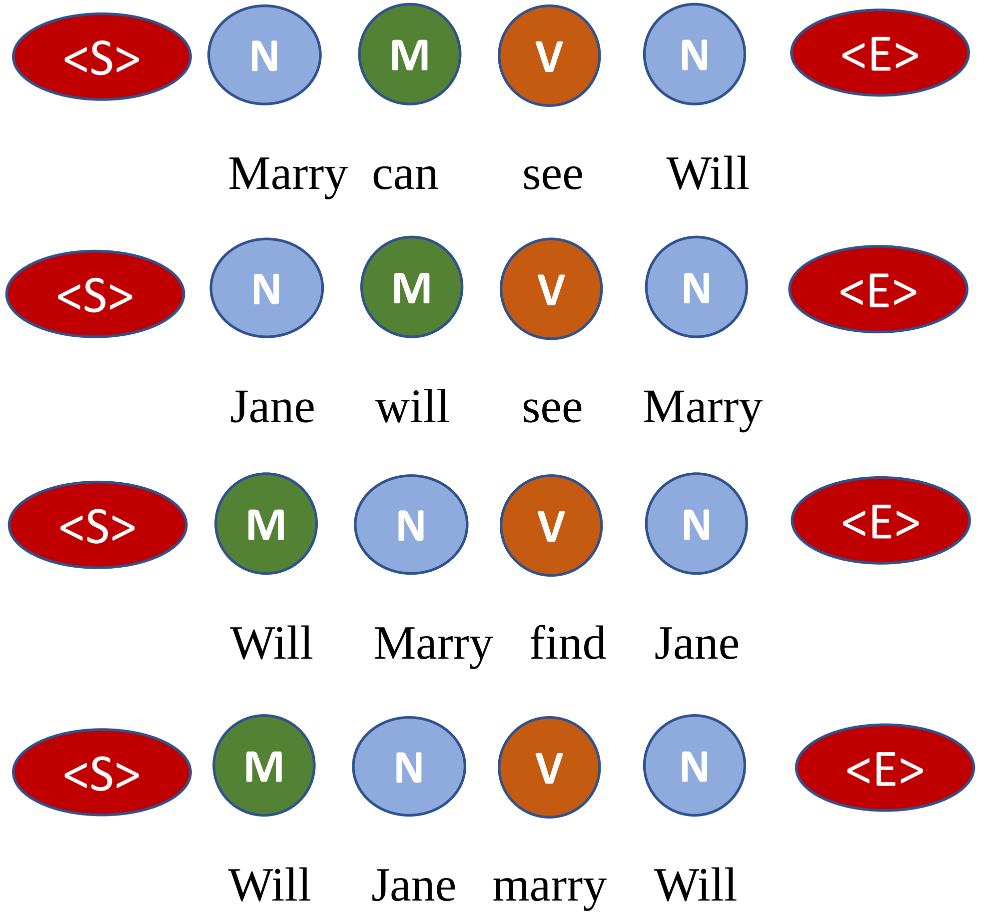


	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

Train a tagger: learn transition Probability

The probability of a POS tag followed by another tag

How likely is that: a Noun followed by a Modal?



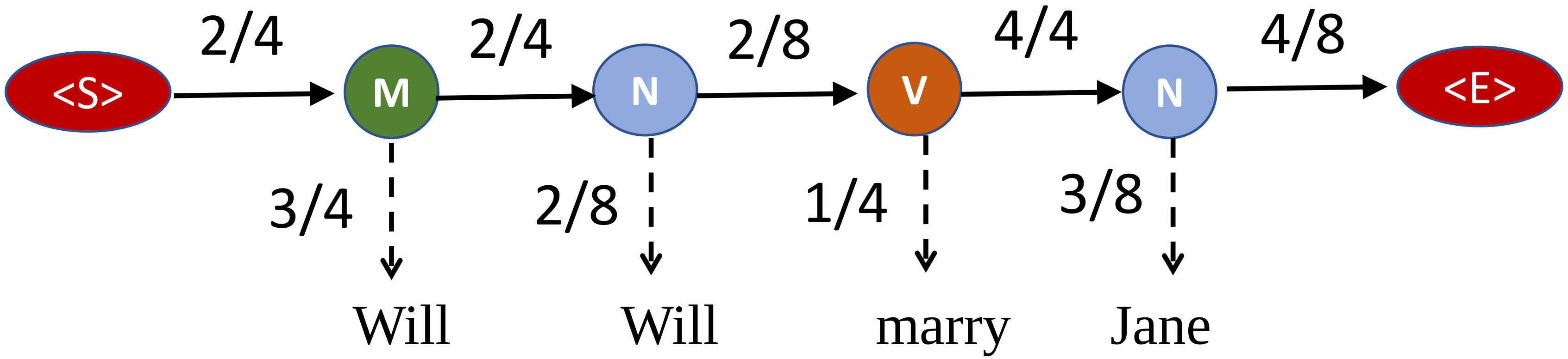
	N	V	M	<E>
<S>	2	0	2	0
N	0	2	2	4
V	4	0	0	0
M	2	2	0	0

	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

Prediction: tag new sentences

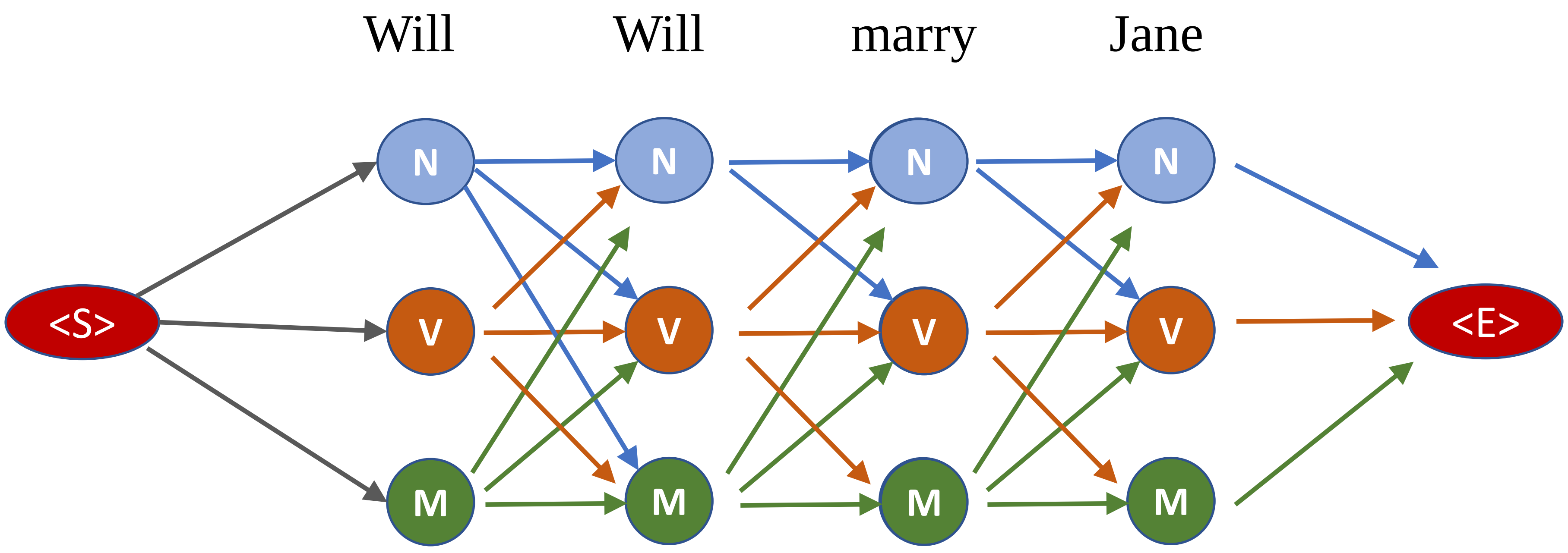
How does the HMM determine the appropriate sequence of tags for a new sentence?

E.g., “Will Will marry Jane?”



<S> -> M -> N -> V -> N -> <E> = 2/4*3/4*2/4*2/8*2/8*1/4*4/4*3/8*4/8

All possible tags



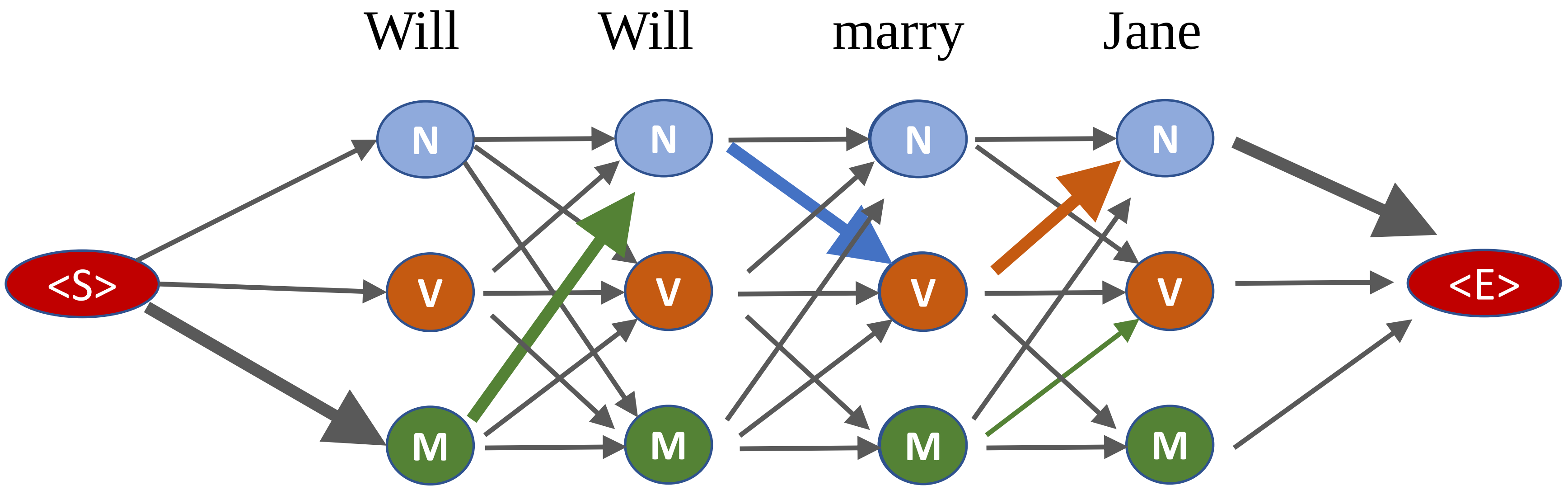
<S> -> M -> N -> V -> N -> <E> = 2/4*3/4*2/4*2/8*2/8*1/4*4/4*3/8

3*3*3*3 = 81 combinations

Optimizing HMM with Viterbi

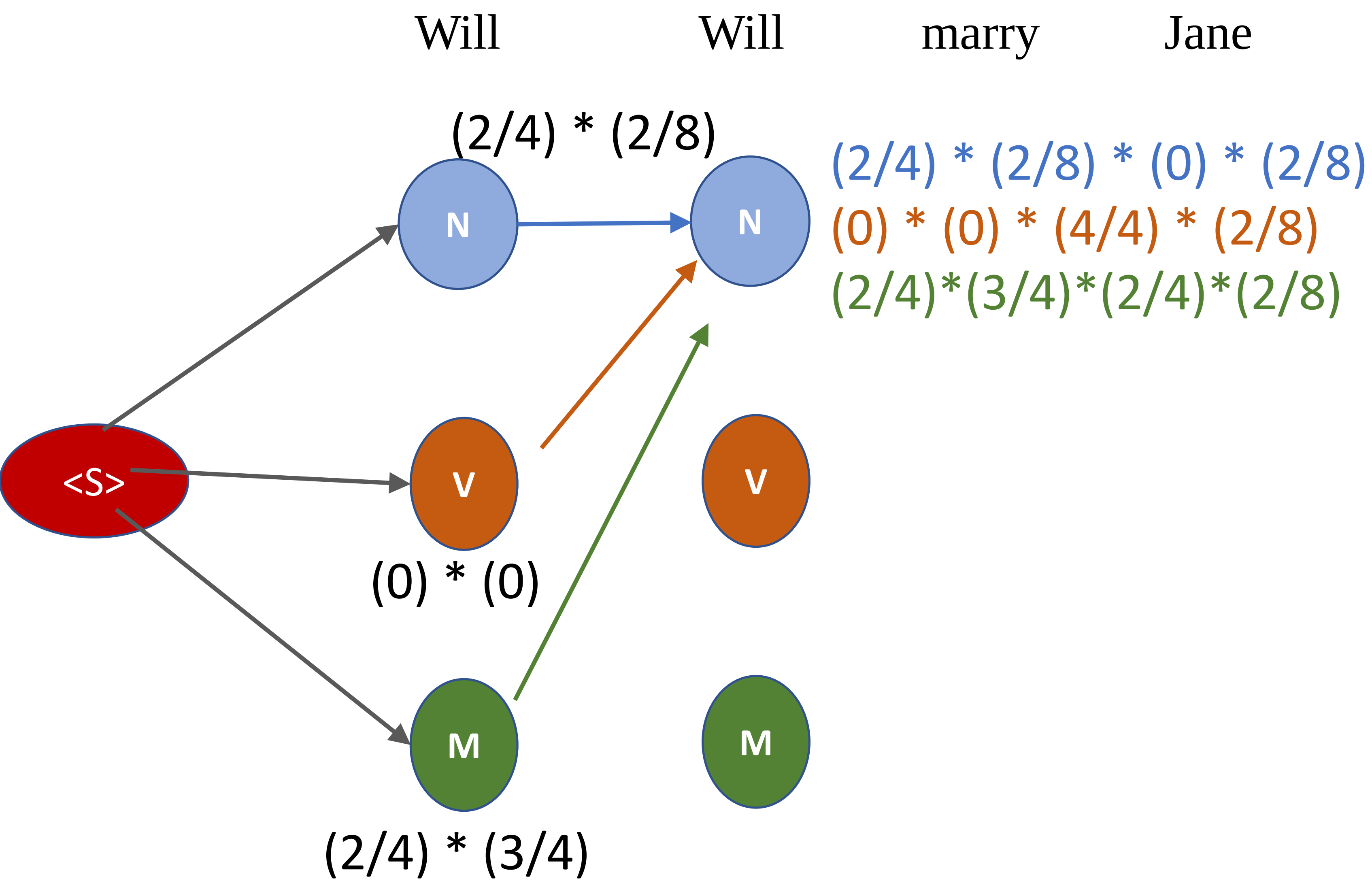
Viterbi algorithm:

- dynamic programming
- **Input**: a single HMM and a sequence of observed words
- **Output**: the most probable hidden state / tag sequence, together with its probabilities
    - Viterbi path



<S> -> M -> N -> V -> N -> <E> = 2/4*3/4*2/4*2/8*2/8*1/4*4/4*3/8

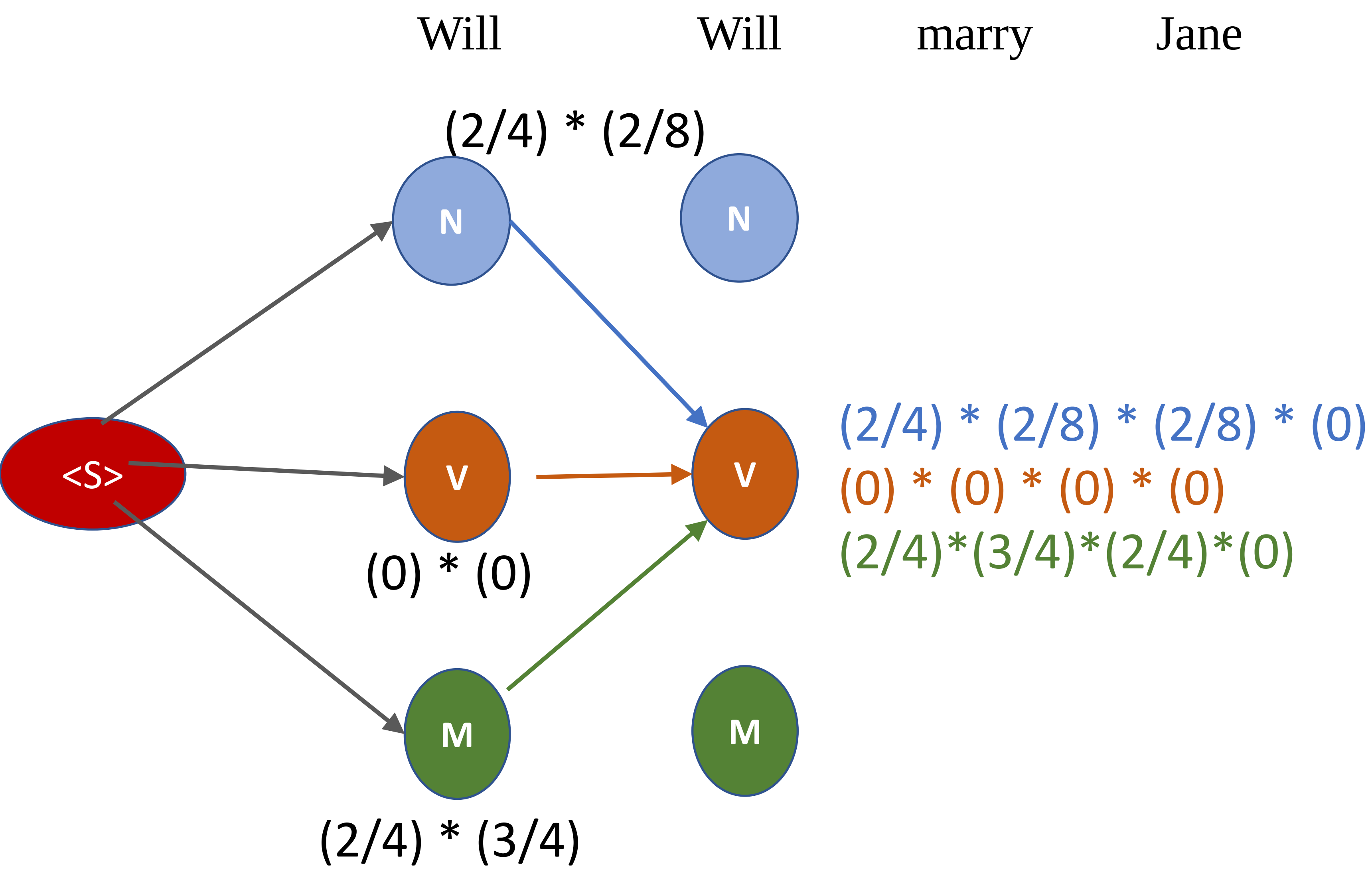
For each point, record the incoming edge that gives the highest probability



	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

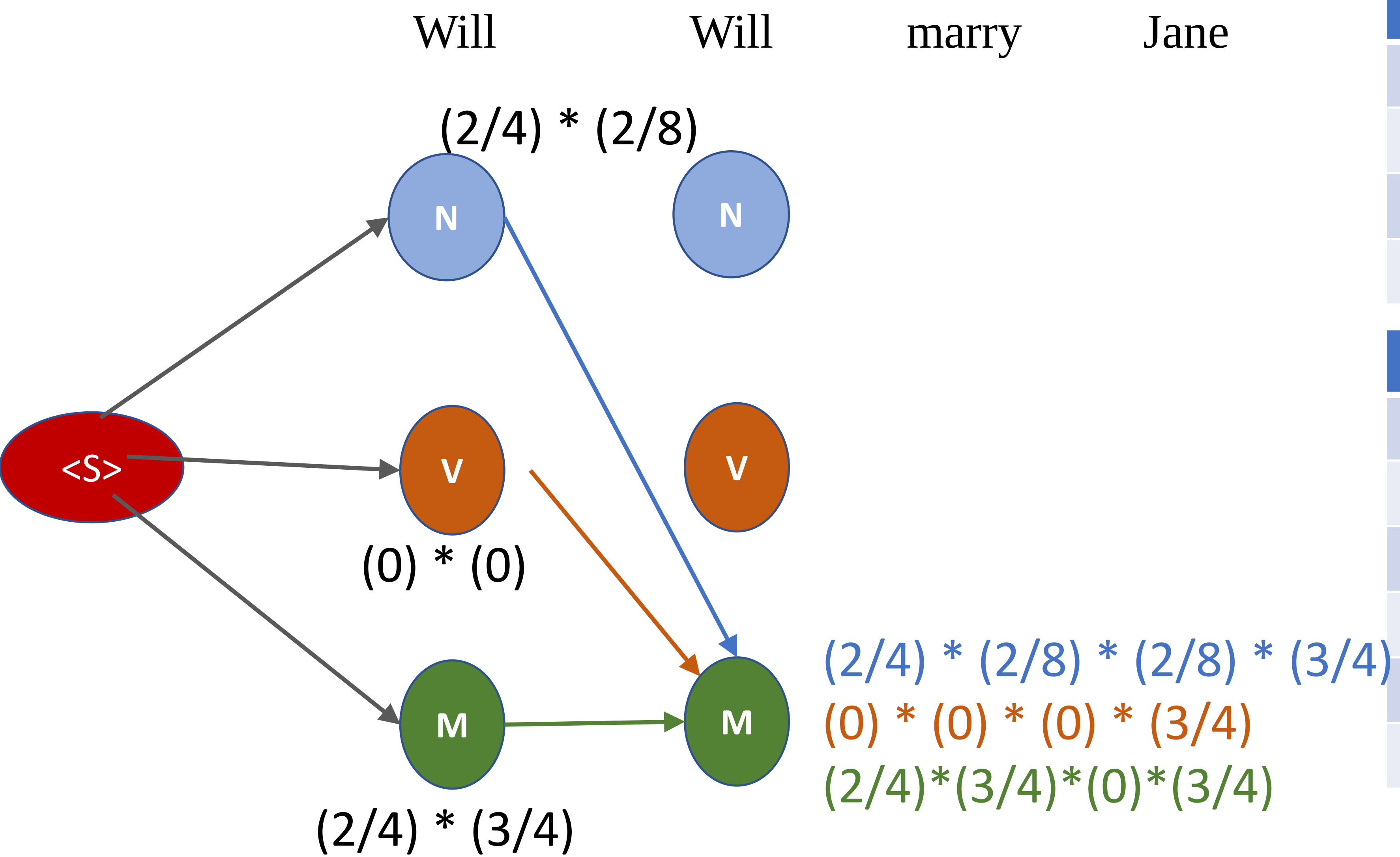
For each point, record the incoming edge that gives the highest probability



	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

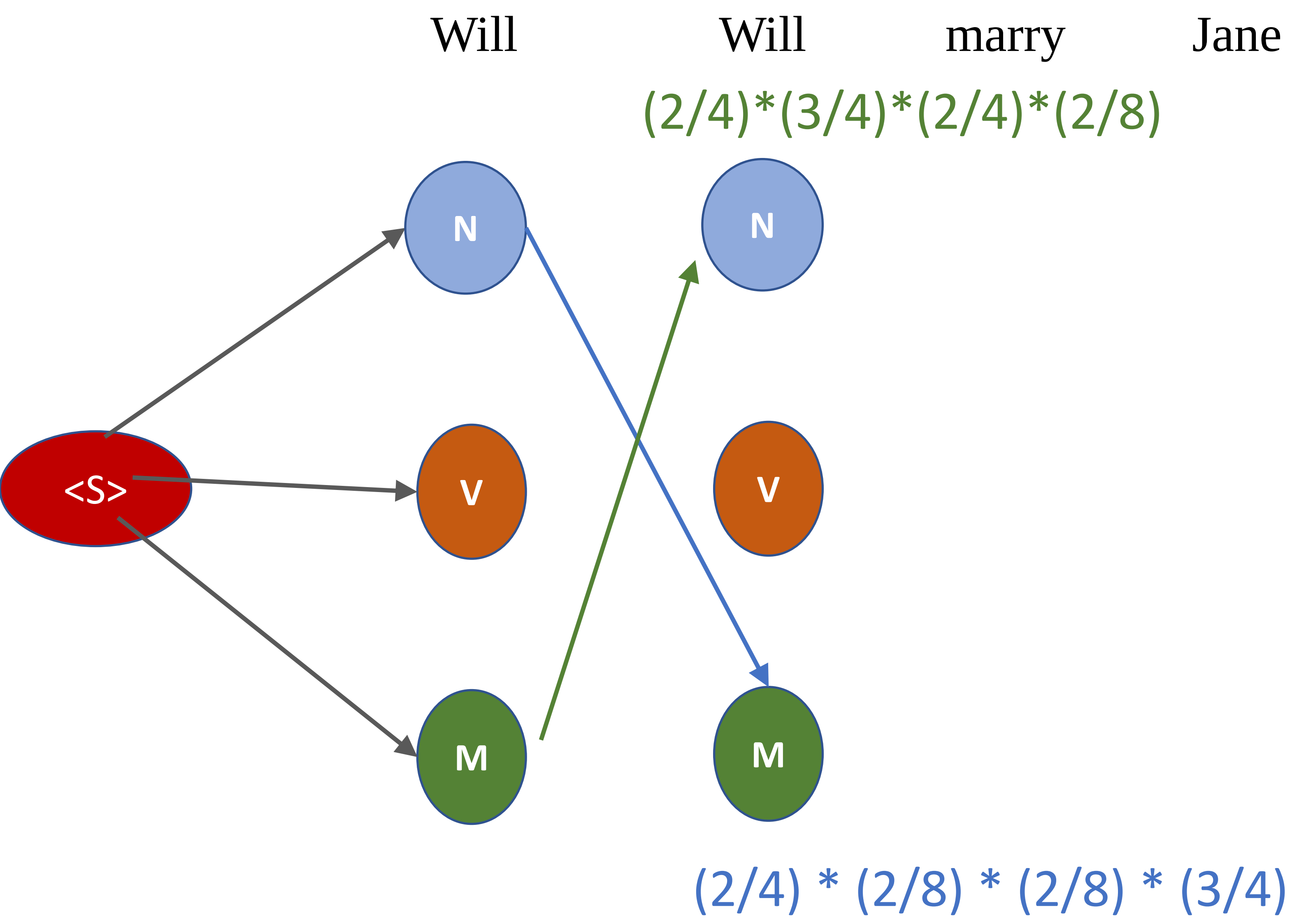
For each point, record the incoming edge that gives the highest probability



	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

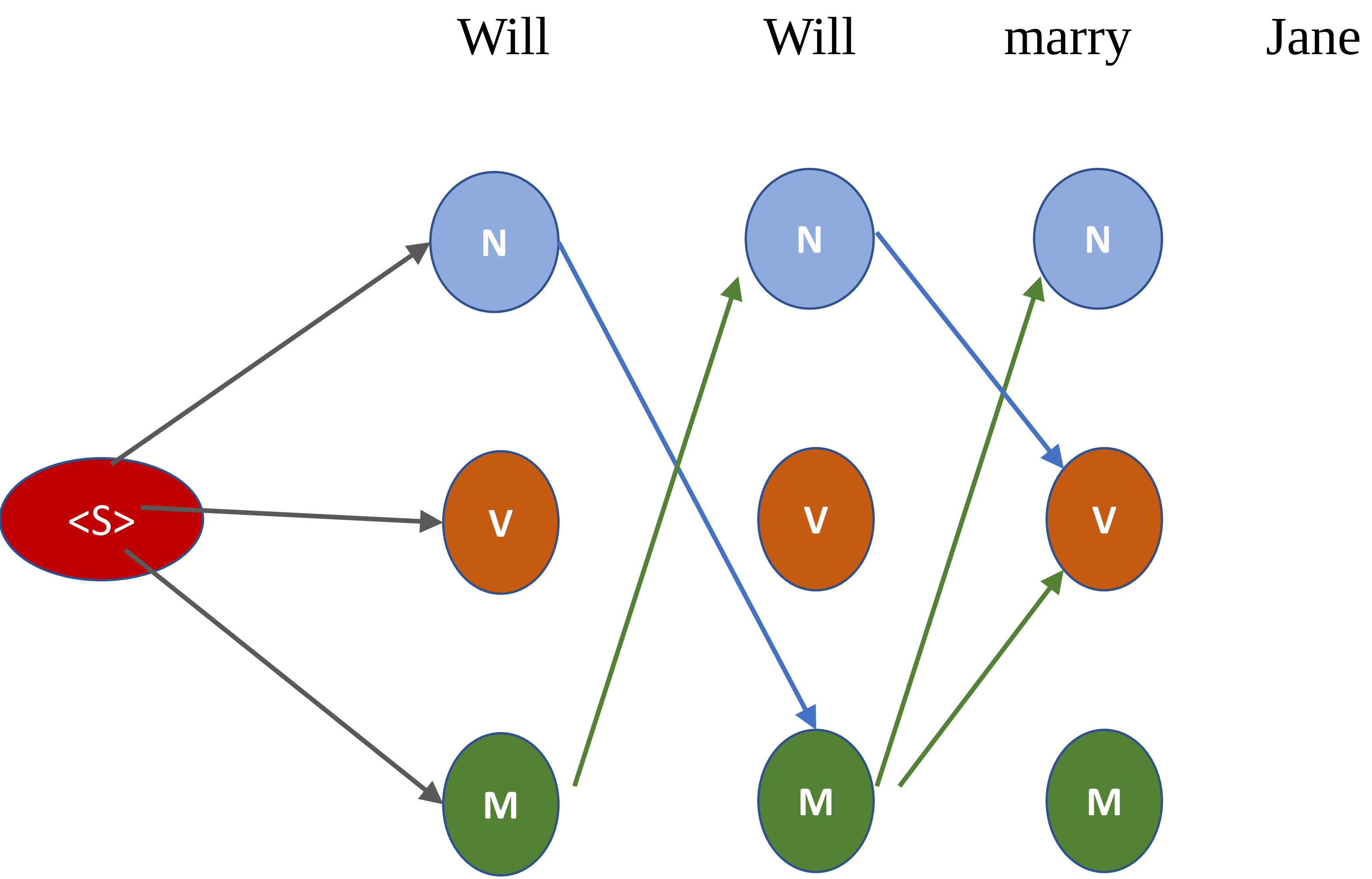
For each point, record the incoming edge that gives the highest probability



	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

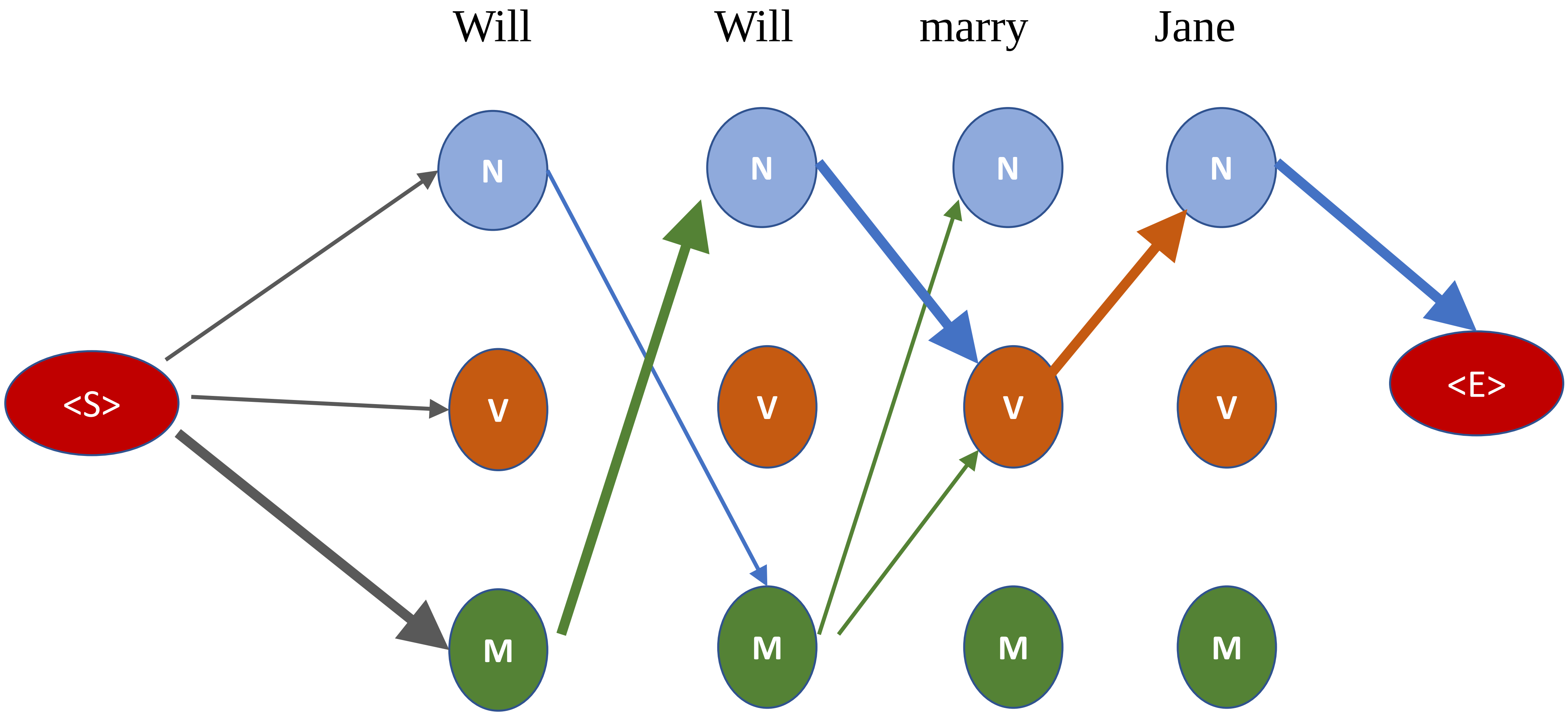
For each point, record the incoming edge that gives the highest probability



	N	V	M	<E>
<S>	2/4	0	2/4	0
N	0	2/8	2/8	4/8
V	4/4	0	0	0
M	2/4	2/4	0	0

	Noun	Verb	Model
marry	3/8	1/4	0
jane	3/8	0	0
will	2/8	0	3/4
can	0	0	1/4
see	0	2/4	0
find	0	1/4	0

For each point, record the incoming edge that gives the highest probability



<S> -> M -> N -> V -> N -> <E> = 2/4*3/4*2/4*2/8*2/8*1/4*4/4*3/8

POS tagging Techniques

- Rule-based: Regular Expression Tagger

- Probabilistic tagging:
  - Default Tagger
  - N-gram Tagger
  - HMM tagger

➢ **Transformation-based:**
  - pre-defined rules
  - automatically induced rule
  - Brill tagger

- Deep learning models:
  - Meta-BiLSTM

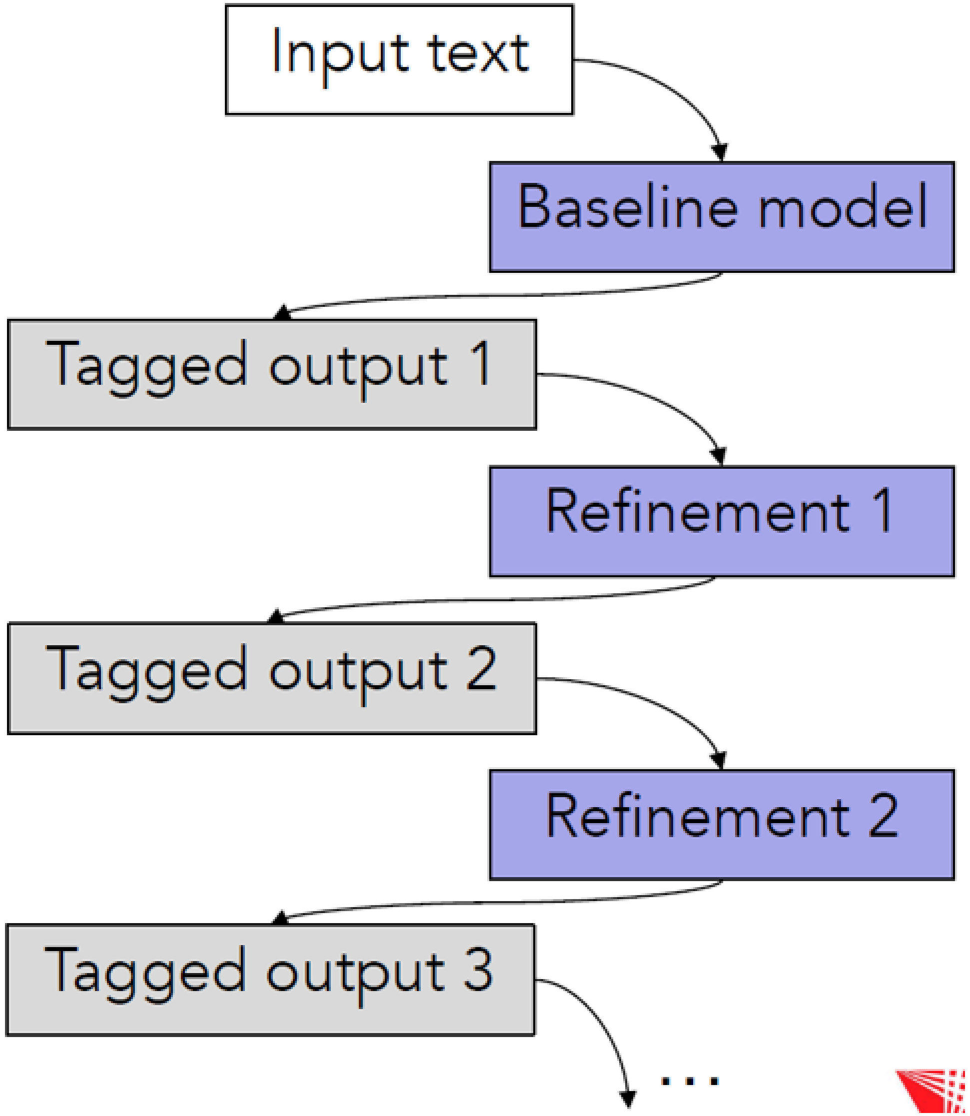
Transformation-based learning (Brill Tagger)

- A pre-tagged training corpus
- Supervised machine learning technique

1) Label every word with its most likely tag;

2) Identify the most common errors;
3) Induce a new rule to fix the errors;

4) Repeat (2) and (3) until reaches the stopping criterion

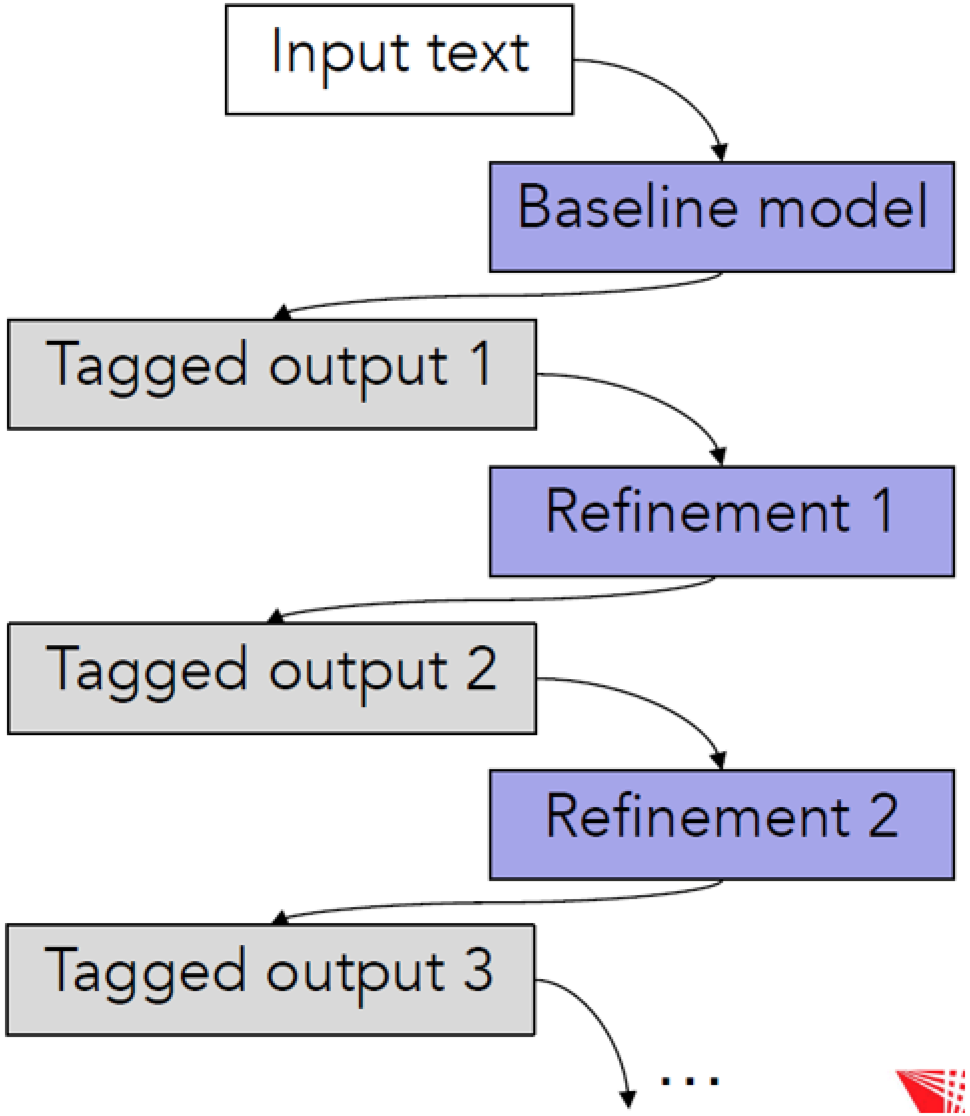


Painting example as analogy

A white house with green trim against a blue sky

1) Big brush, paint the entire canvas blue;
2) Medium brush, color the white house;
3) Small brush, trim on the gables;

Start from a broad layer, gradually corrects smaller and smaller layers.



Rule-templates for Brill Tagger

Change tag a to tag b when:

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

	Change tags			
#	**From**	**To**	**Condition**	**Example**
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

Brill Tagger: tradeoff

Pro:

- Simple ways of incorporating both local context features and top-down information about consistency of tag sequences
- Good accuracy, especially on unknown words

Con:

- Fairly slow to learn and slow at prediction time