

# Hidden Markov Model (HMM)

## Part-of-Speech Tagging

IIT – CS481 – Spring

2021.02.18

# Review

## Contents

- ▼ 1 What is Part-of-Speech tag?
  - 1.1 Tagset
  - 1.2 Explore NLTK tagged corpora
  - 1.3 POS tagging example
  - 1.4 POS tagging techniques
- ▼ 2 Automatic Tagging
  - 2.1 Rule-based tagger: Regular Expression Tagger
  - ▼ 2.2 Probabilistic tagging:
    - 2.2.1 Creating lookup table
    - 2.2.2 The Default Tagger
    - 2.2.3 The Lookup Tagger
  - ▼ 2.3 Evaluate Tagger performance
    - 2.3.1 Separate Training and Testing Data
  - ▼ 2.4 General N-Gram Tagging
    - 2.4.1 Combine several n-gram taggers
  - 2.5 Tagging Unknown Words
  - 2.6 Storing pre-trained Taggers

# POS tagging techniques

---

- Rule-based: Regular Expression Tagger

## ➤ **Probabilistic tagging:**

- Default Tagger
- N-gram Tagger
- **HMM tagger**

## ➤ **Transformation-based:**

- applies pre-defined rules as well as rules automatically induced during training
- Brill tagger

- Deep learning models:
  - Meta-BiLSTM

# Probability Tagging

---

- 1) Data:  
tagged corpus
- 2) Train a tagger:  
create a lookup table to store the most frequent tag for each word
- 3) Prediction: tag new sentences  
i.e., sentences not seen in the training data
- 4) Evaluation:  
train test split  
tags assigned by human expert as gold standard

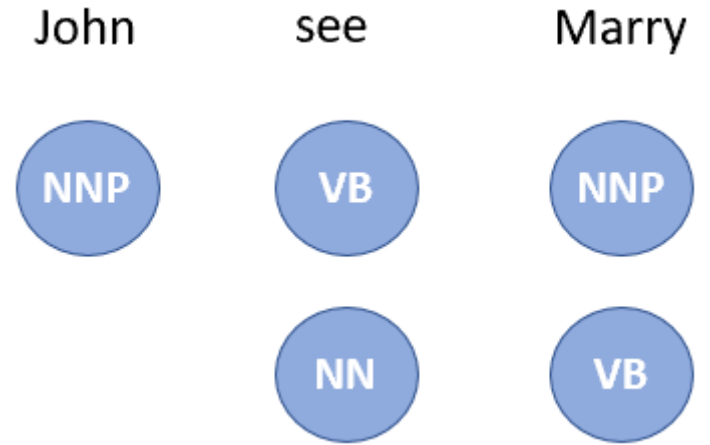
# HMM POS tagging

**Observation:** a sequence of words  $w_1^n$

**Goal:** assign a sequence of POS tags  $t_1^n$

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

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



# Hidden Markov Model

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

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


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

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

Drop denominator  $P(w_1^n)$

# HMM tagger: 2 assumptions

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

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


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

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

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

# HMM: 2 probabilities

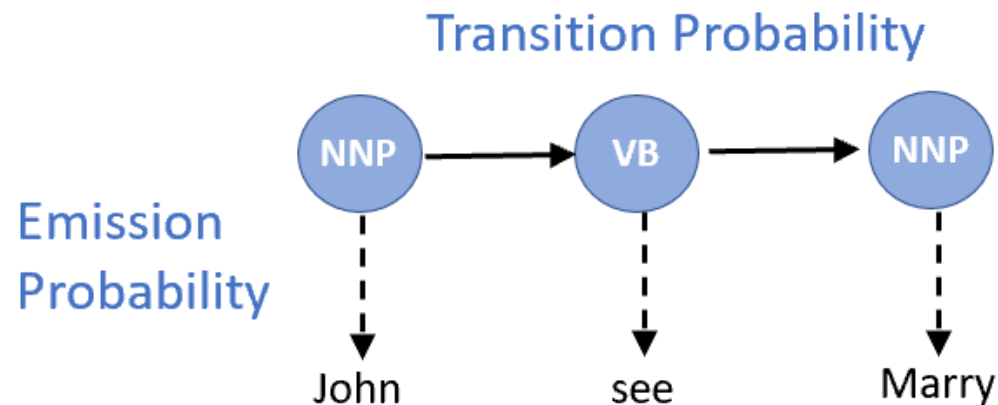
$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

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

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

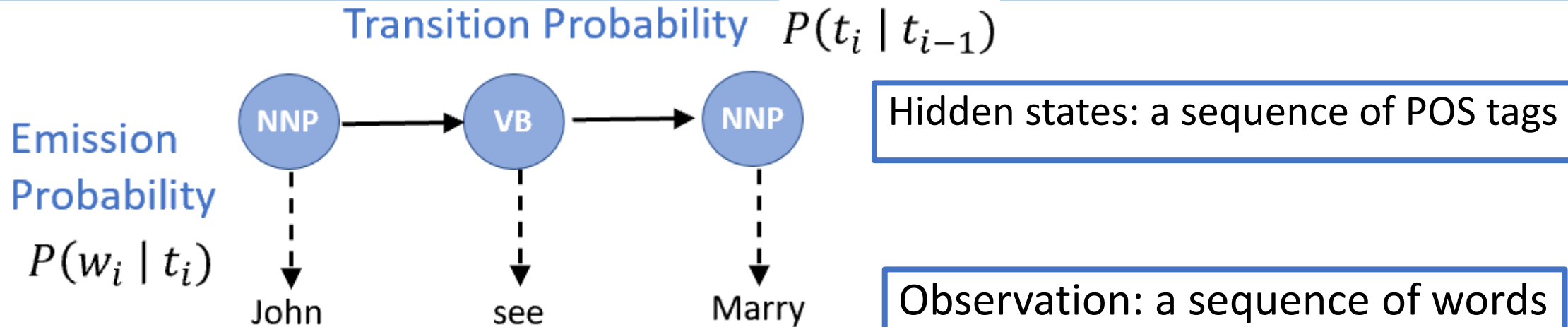
$$P(\text{see} | VB) = \frac{C(VB, \text{see})}{C(VB)} = 0.057$$

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





# Formalizing HMM tagger



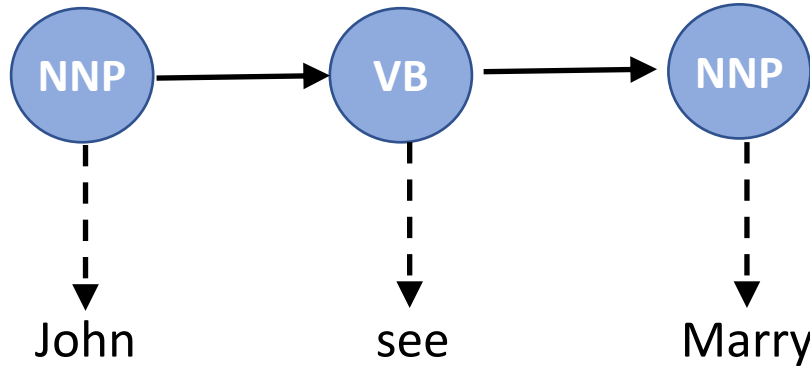
Emission Probability: given a particular tag, how likely is a word generated from this particular tag?

- 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



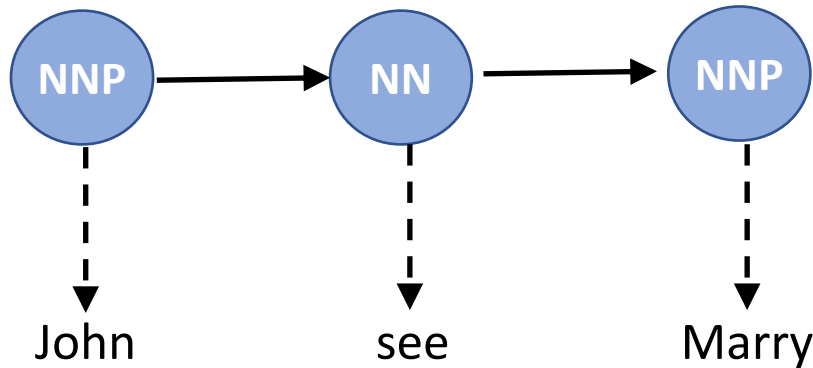
$$P(\textit{see}|\textit{VB}) = 0.057$$

$$P(\textit{VB}|\textit{NNP}) = 0.49$$

$$P(\textit{NNP}|\textit{VB}) = 0.27$$

$$P(\textit{NNP}) * P(\textit{John}|\textit{NNP}) *$$

$$P(\textit{VB}|\textit{NNP}) * P(\textit{see}|\textit{VB}) * P(\textit{NNP}|\textit{VB}) * P(\textit{Marry}|\textit{NNP}) = 0.0075411$$



$$P(\textit{see}|\textit{NN}) = 0.0012$$

$$P(\textit{NN}|\textit{NNP}) = 0.047$$

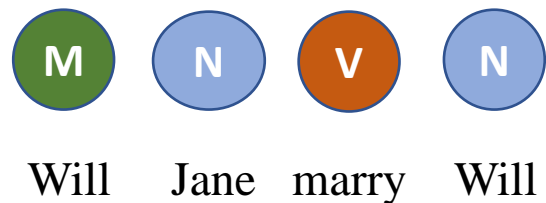
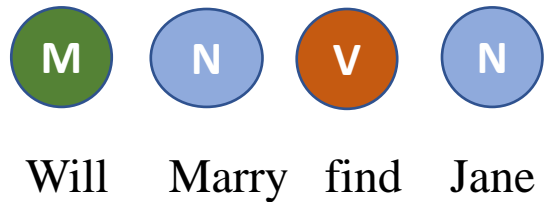
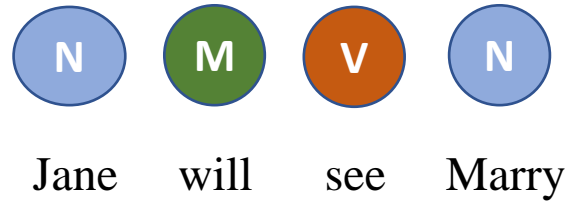
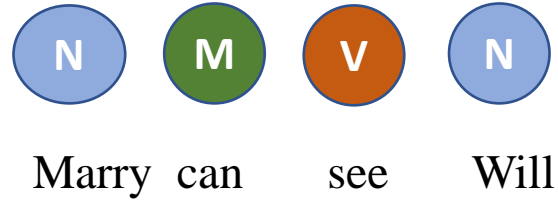
$$P(\textit{NNP}|\textit{NN}) = 0.001$$

$$P(\textit{NNP}) * P(\textit{John}|\textit{NNP}) *$$

$$P(\textit{NN}|\textit{NNP}) * P(\textit{see}|\textit{NN}) * P(\textit{NNP}|\textit{NN}) * P(\textit{Marry}|\textit{NNP}) = 0.0000000564$$

# Data

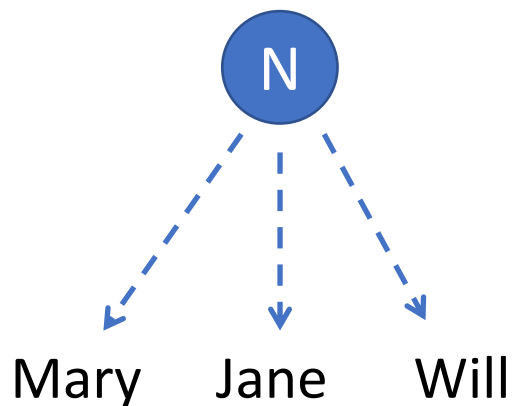
- Marry can see Will
- Jane will see Marry
- Will Marry find Jane?
- Will Jane marry Will?



|       | 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.

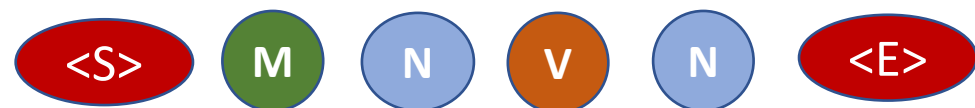
How likely is that: a Noun followed by a Modal?



Marry can see Will



Jane will see Marry



Will Marry find Jane



Will Jane marry Will

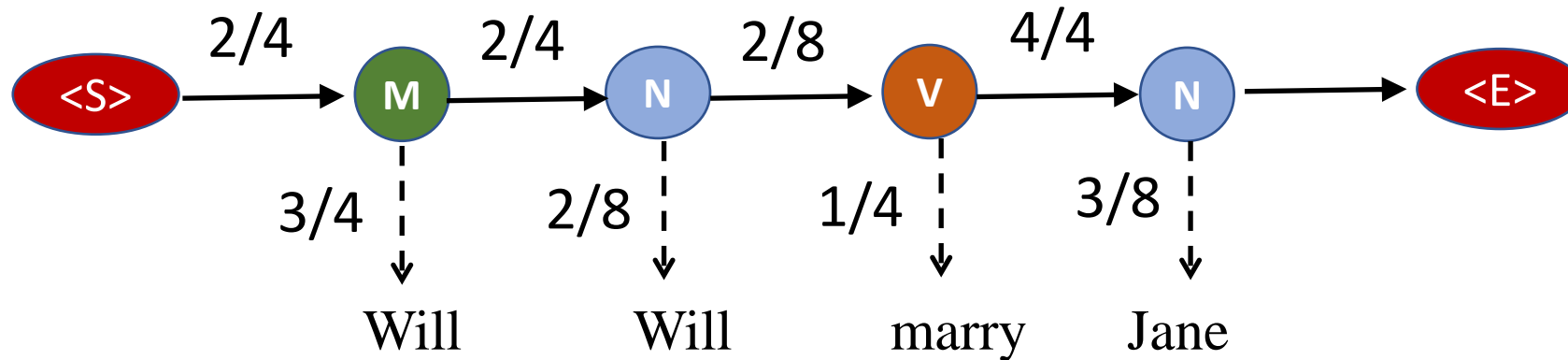
|     | N | V | M | <E> |
|-----|---|---|---|-----|
| <S> | 2 | 0 | 2 | 0   |
| N   | 0 | 2 | 2 | 4   |
| V   | 4 | 0 | 0 | 0   |
| M   | 2 | 2 | 0 | 0   |

|     | 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

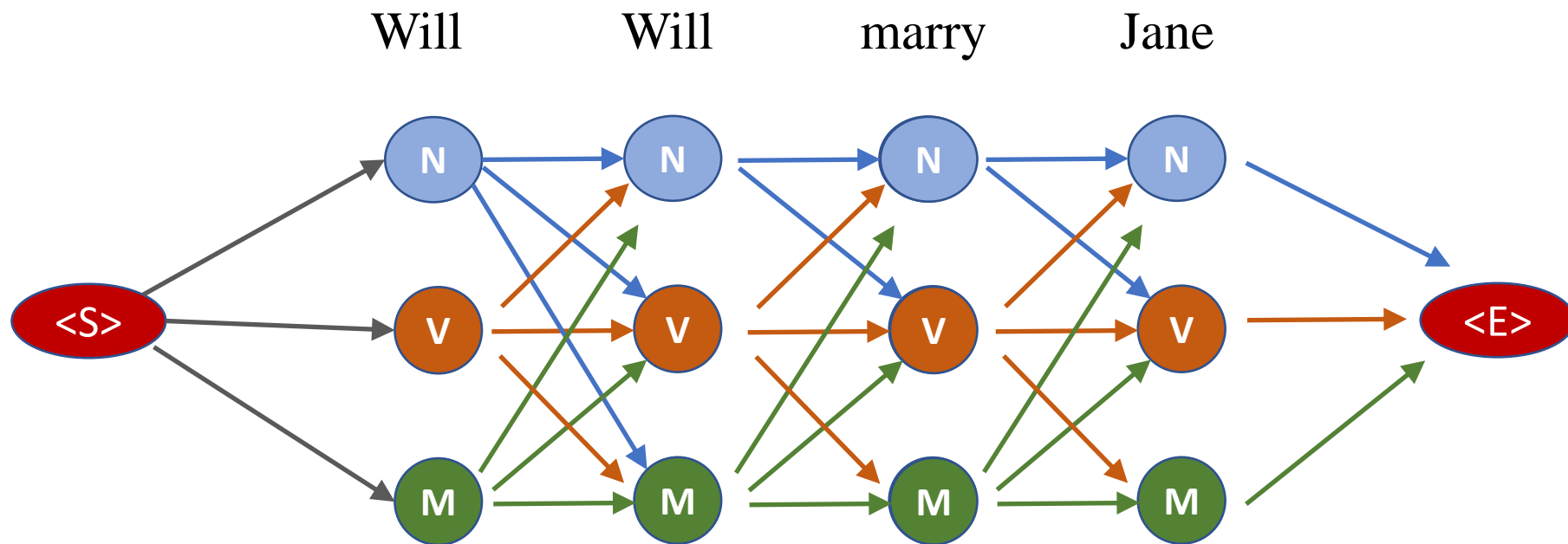
How does the HMM determine the appropriate sequence of tags for a new sentence

E.g., Will Will marry Jane?



$$\langle S \rangle \rightarrow M \rightarrow N \rightarrow V \rightarrow N \rightarrow \langle E \rangle = 2/4 * 3/4 * 2/4 * 2/8 * 2/8 * 1/4 * 4/4 * 3/8$$

# All possible tags



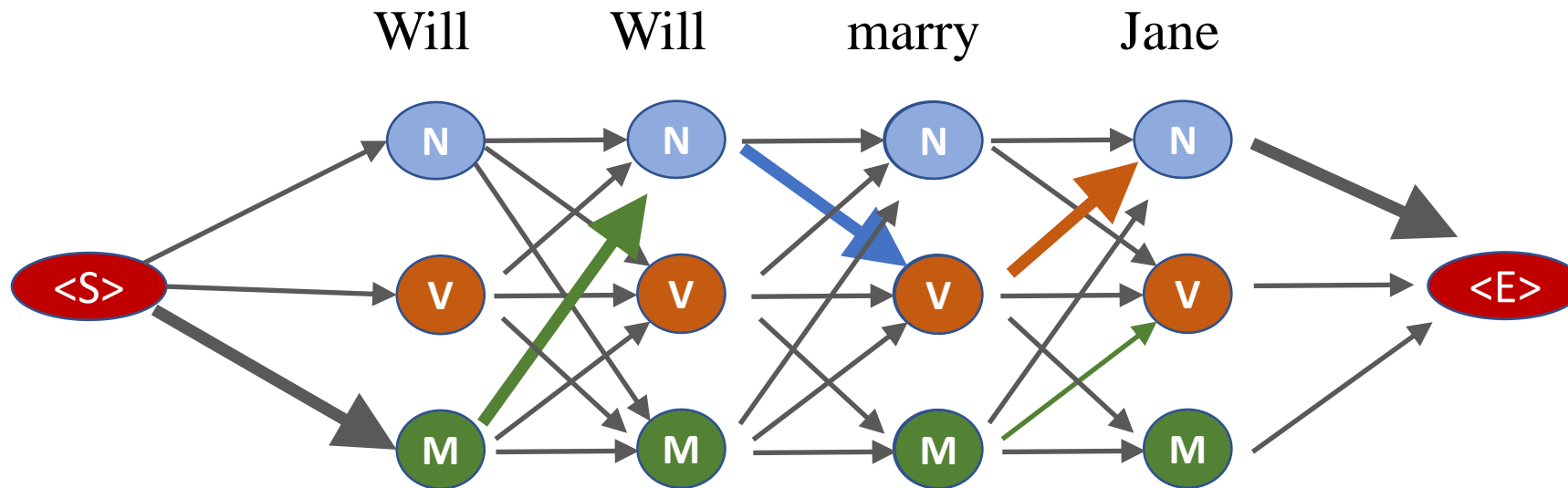
$$\langle S \rangle \rightarrow M \rightarrow N \rightarrow V \rightarrow N \rightarrow \langle E \rangle = 2/4 * 3/4 * 2/4 * 2/8 * 2/8 * 1/4 * 4/4 * 3/8$$

$$3 * 3 * 3 * 3 = 81 \text{ combinations}$$

# Optimizing HMM with Viterbi

Viterbi algorithm:

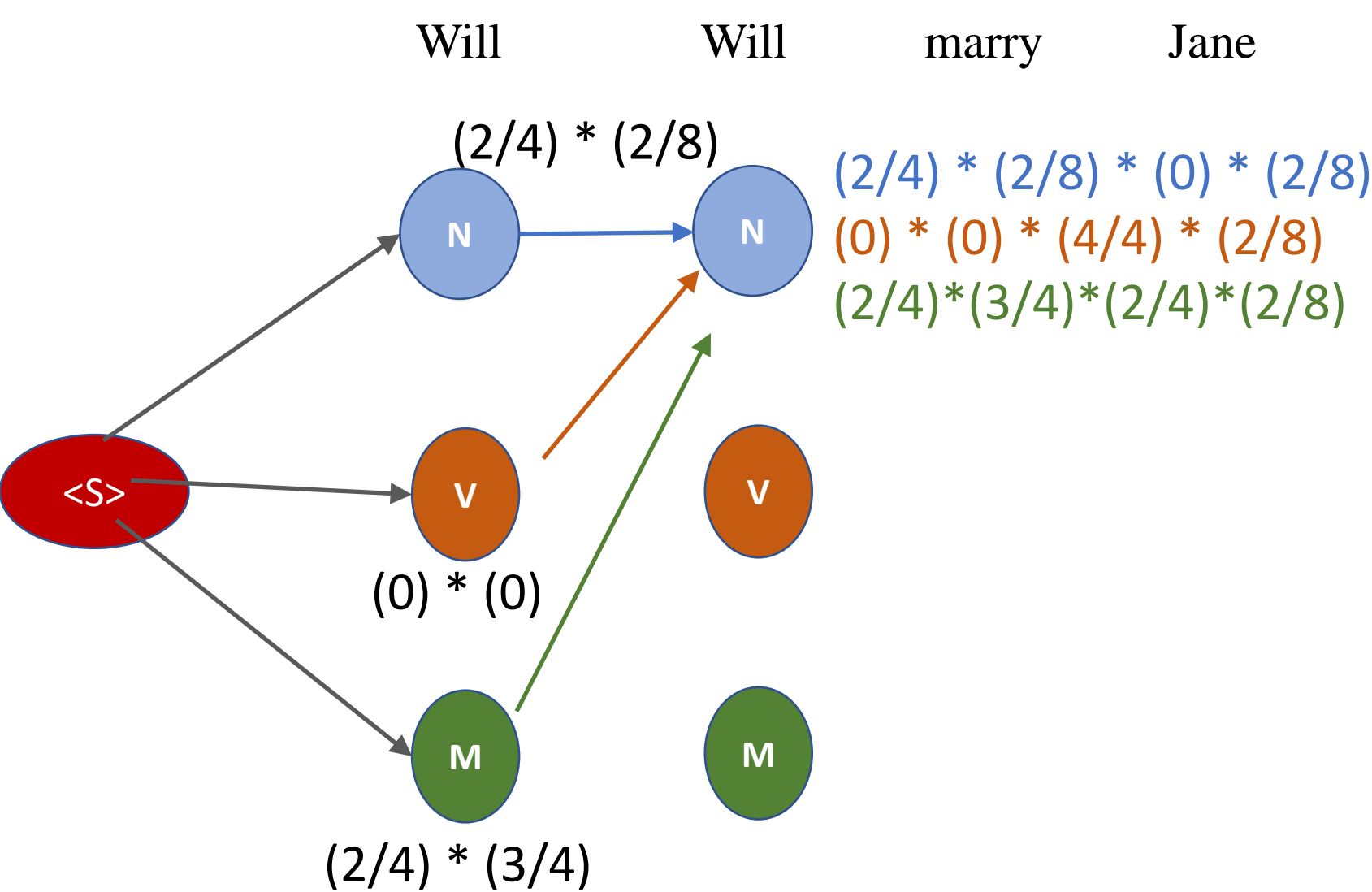
- dynamic programming
- **Input:** a single HMM and a sequence of observed words
- **Output:** the most probable hidden state / tag sequence, together with its probabilities
  - Viterbi path



$$\langle S \rangle \rightarrow M \rightarrow N \rightarrow V \rightarrow N \rightarrow \langle E \rangle = \frac{2}{4} * \frac{3}{4} * \frac{2}{4} * \frac{2}{8} * \frac{2}{8} * \frac{1}{4} * \frac{4}{4} * \frac{3}{8}$$



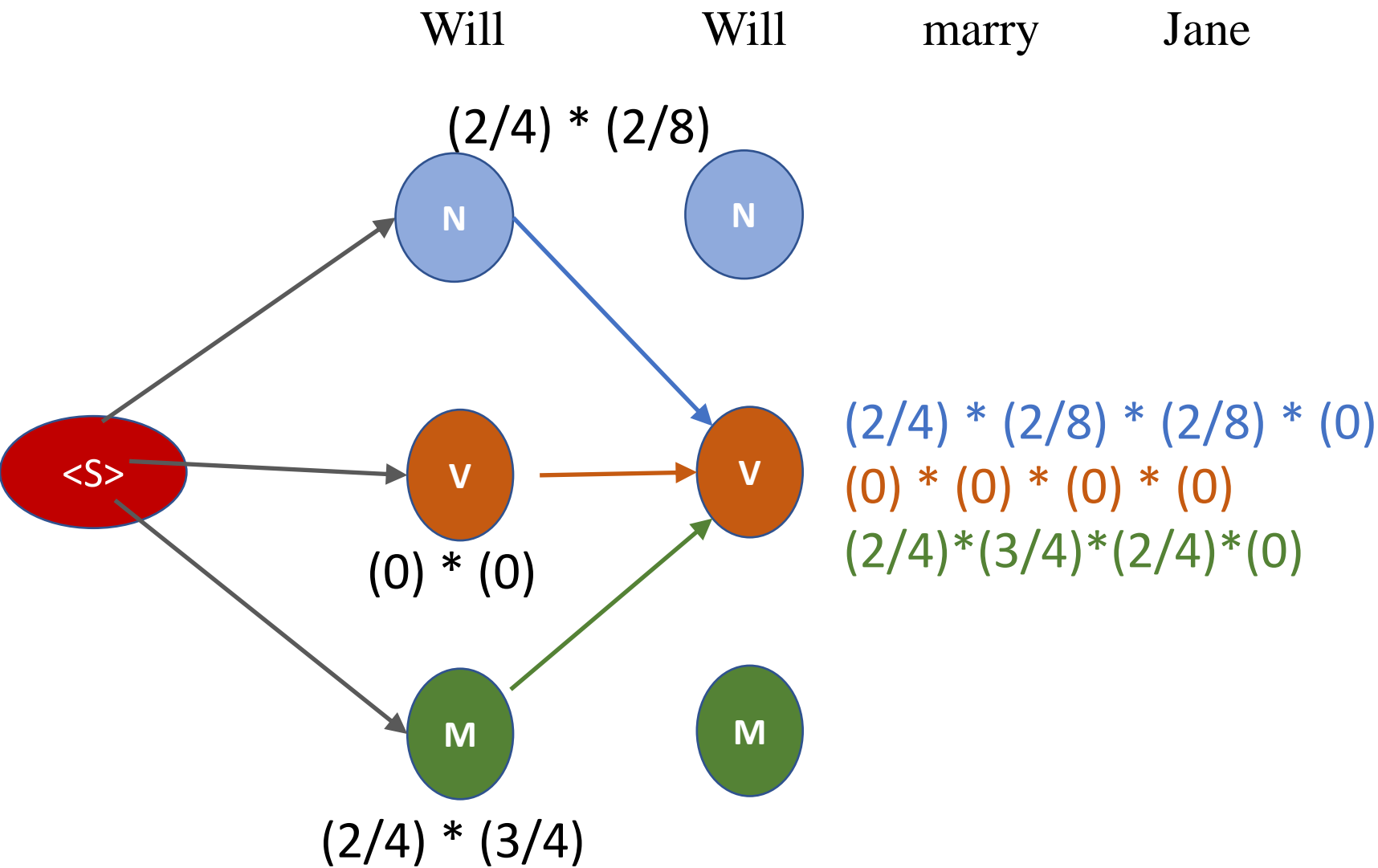
For each point, record the incoming edge that gives the highest probability



|     | 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     |

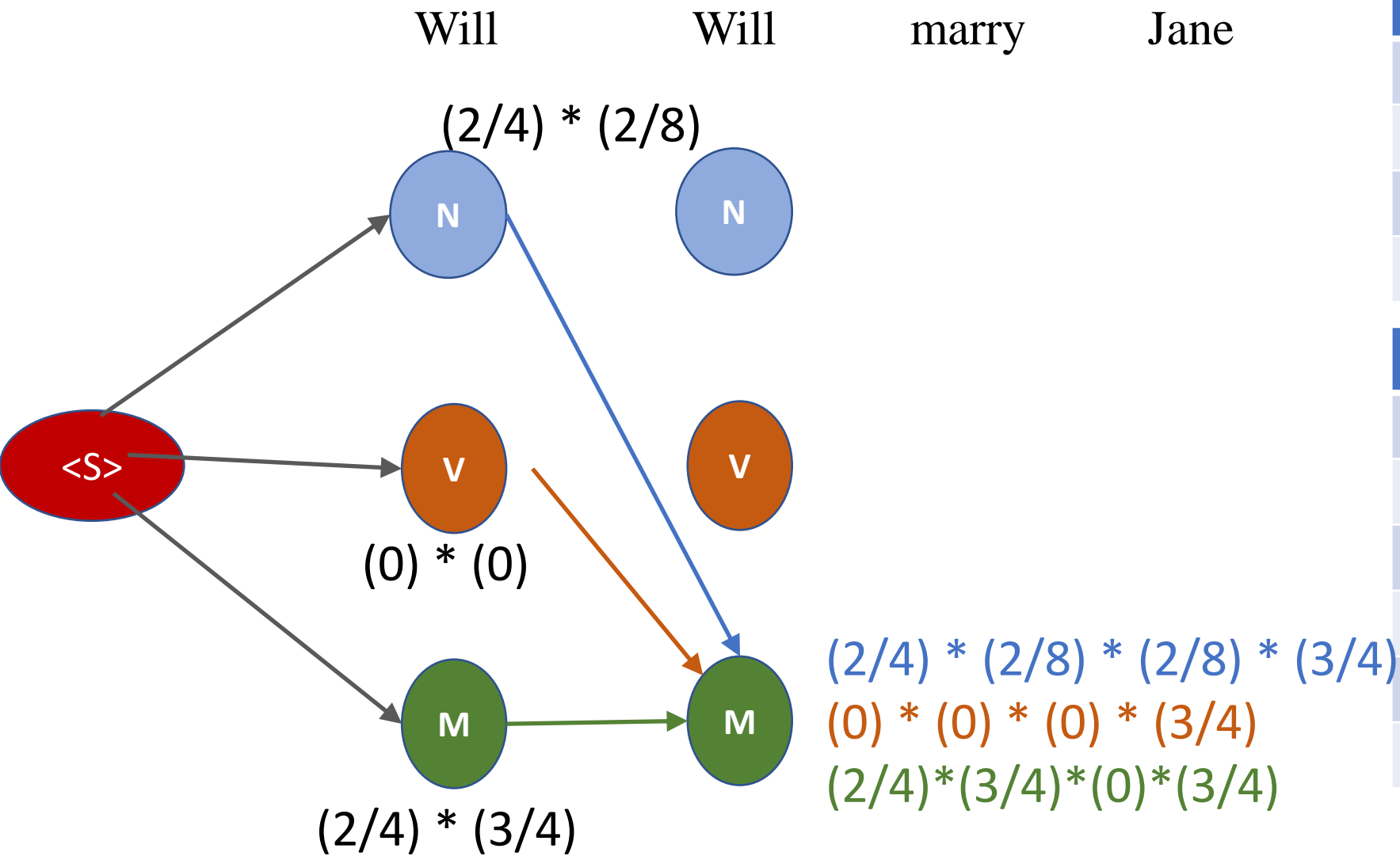
For each point, record the incoming edge that gives the highest probability



|     | 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     |

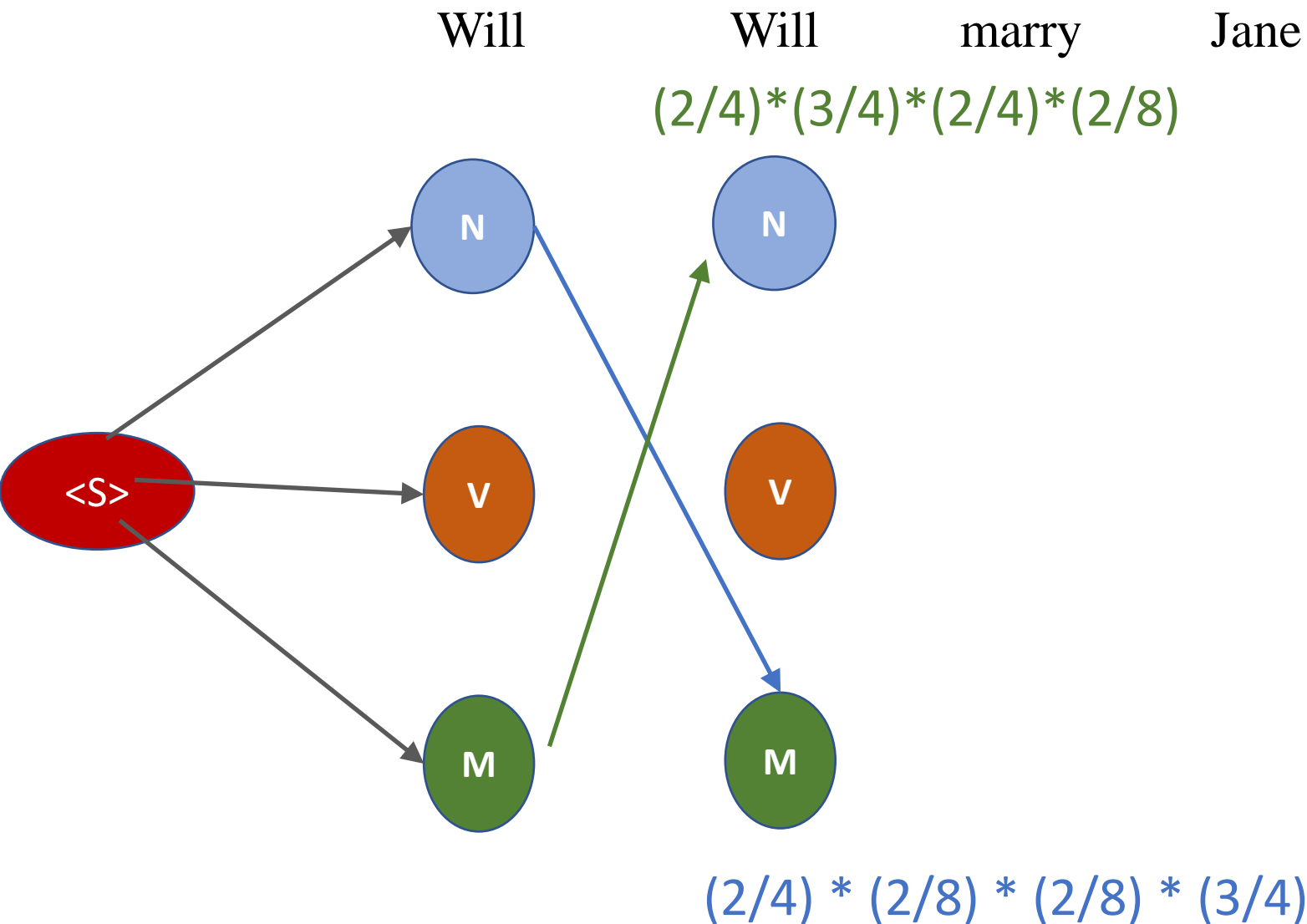
For each point, record the incoming edge that gives the highest probability



|     | 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     |

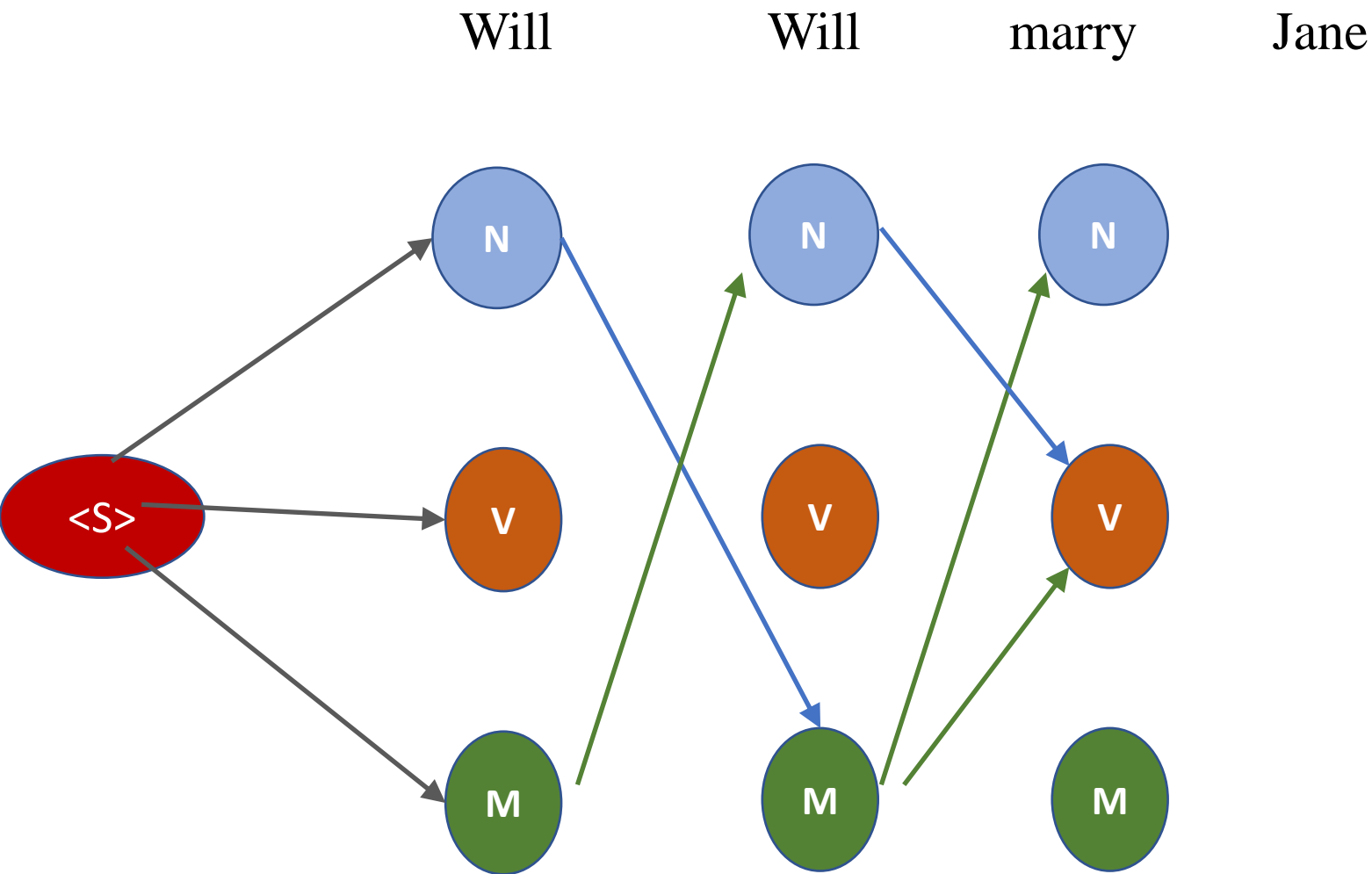
For each point, record the incoming edge that gives the highest probability



|     | 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     |

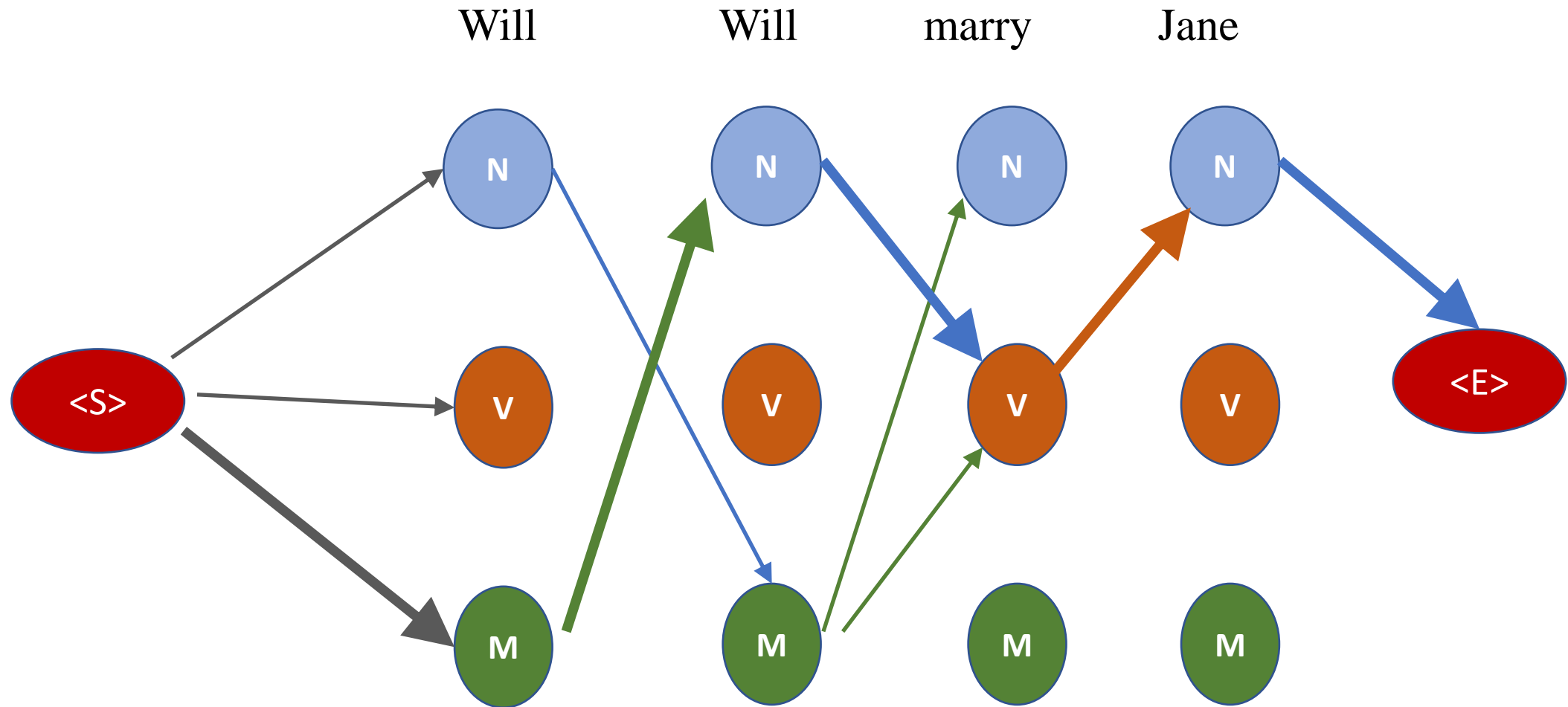
For each point, record the incoming edge that gives the highest probability



|     | 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



$$\langle S \rangle \rightarrow M \rightarrow N \rightarrow V \rightarrow N \rightarrow \langle E \rangle = \frac{2}{4} * \frac{3}{4} * \frac{2}{4} * \frac{2}{8} * \frac{2}{8} * \frac{1}{4} * \frac{4}{4} * \frac{3}{8}$$

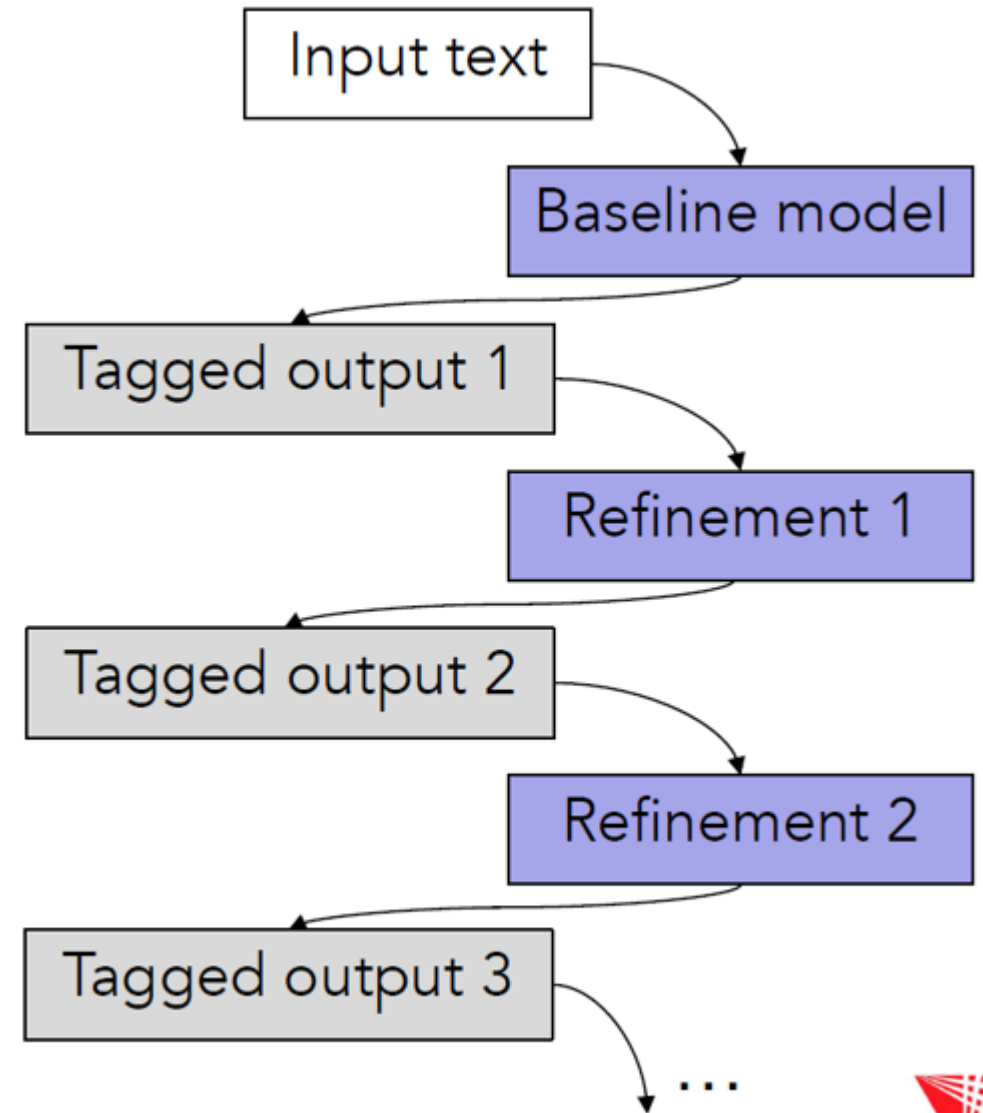
# POS tagging Techniques

---

- Rule-based: Regular Expression Tagger
- Probabilistic tagging:
  - Default Tagger
  - N-gram Tagger
  - HMM tagger
- **Transformation-based:**
  - applies pre-defined rules as well as rules automatically induced during training
  - 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



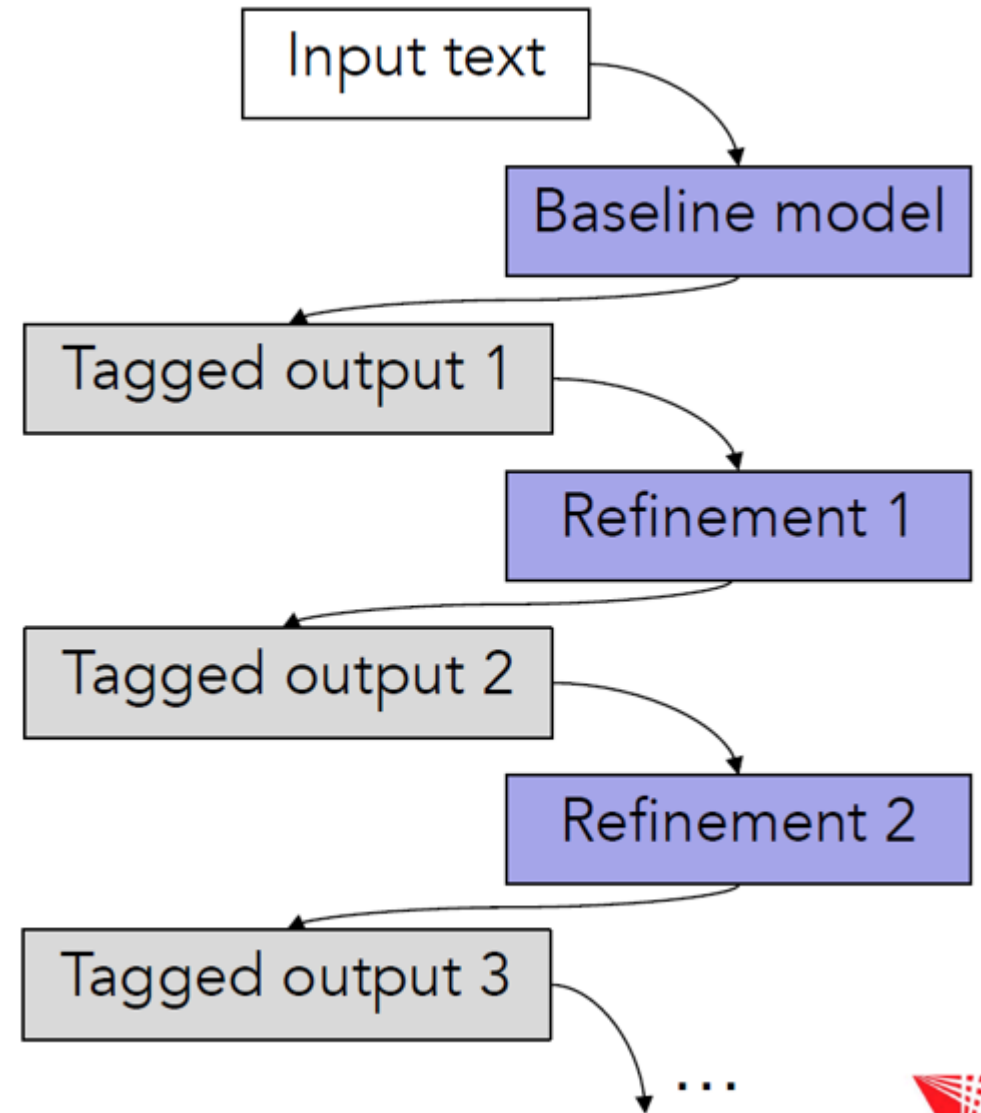


# Painting example as analogy

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  | VCN | 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