# Natural Language Processing Specialization

## Course 2 – Natural Language Processing with Probabilistic Models

### Week 1 – Autocorrect

#### Objective of this Week:

Learn about autocorrect, minimum edit distance, and dynamic programming, then build your own spellchecker to correct misspelled words!

Lecture: **Autocorrect and Minimum Edit Distance**

**Week 2 – Part of Speech Tagging**

Part of Speech Tagging (POS) is the process of assigning a part of speech to a word. By doing so, you will learn the following:

Markov Chains

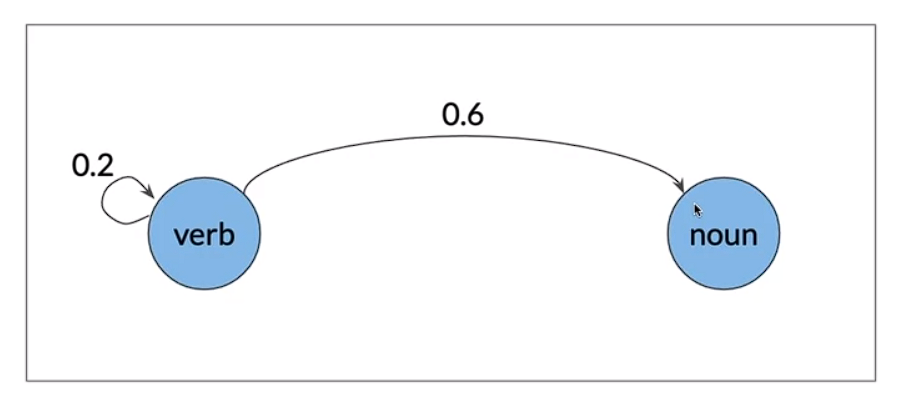
Hidden Markov Models

Viterbi algorithm

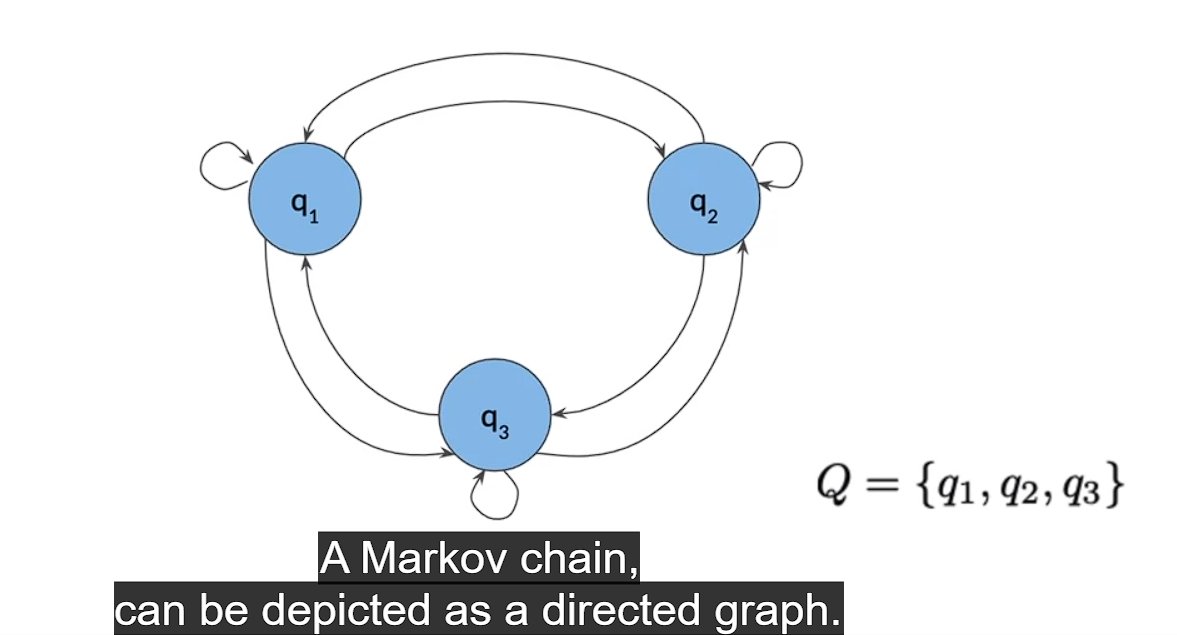
Applications of POS tagging:

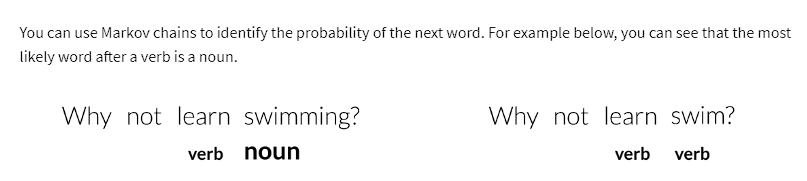
1. Identifying named entities
2. Speech recognition
3. Co-reference Resolution

#### Markov Chains



Markov Chains are a type of stochastic model that describes a sequence of possible events. To get the probability of each event, it needs only the states of the previous events. The word stochastic just means random or randomness.

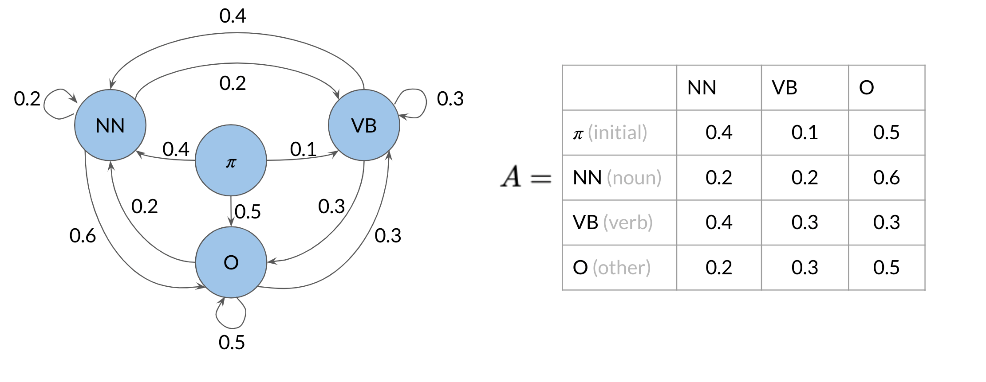




#### Markov Chains and POS Tags

Transition Probabilities tell you about the chances of going from one POS to another.

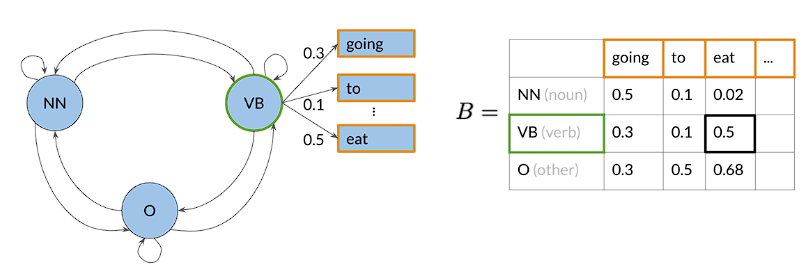
To help identify the parts of speech for every word, you need to build a transition matrix that gives you the probabilities from one state to another.



Markov property. Which states that the probability of the next event only depends on the current event. The Markov property helps keep the model simple. By saying, all you need to determine the next state is the current states. It doesn't need information from any of the previous states.

#### Hidden Markov Models

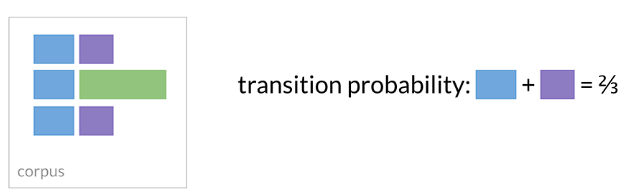
The transition probabilities allowed you to identify the transition probability from one POS to another. In hidden markov models you make use of emission probabilities that give you the probability to go from one state (POS tag) to a specific word.



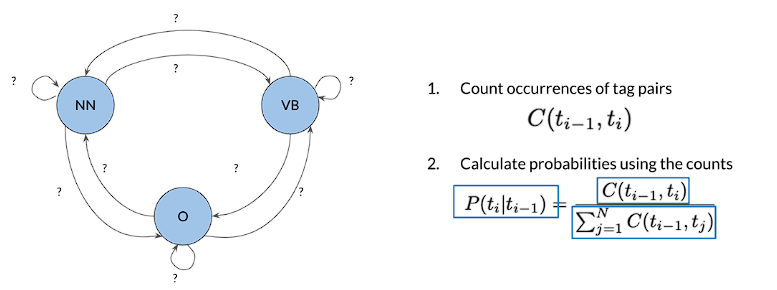
This emission matrix B, will be used with transition matrix A, to help you identify the part of speech of a word in a sentence. To populate the matrix B, you can just have a labelled dataset and compute the probabilities of going from a POS to each word in your vocabulary.

#### Calculating Probabilities

A visual representation of how to calculate the probabilities:

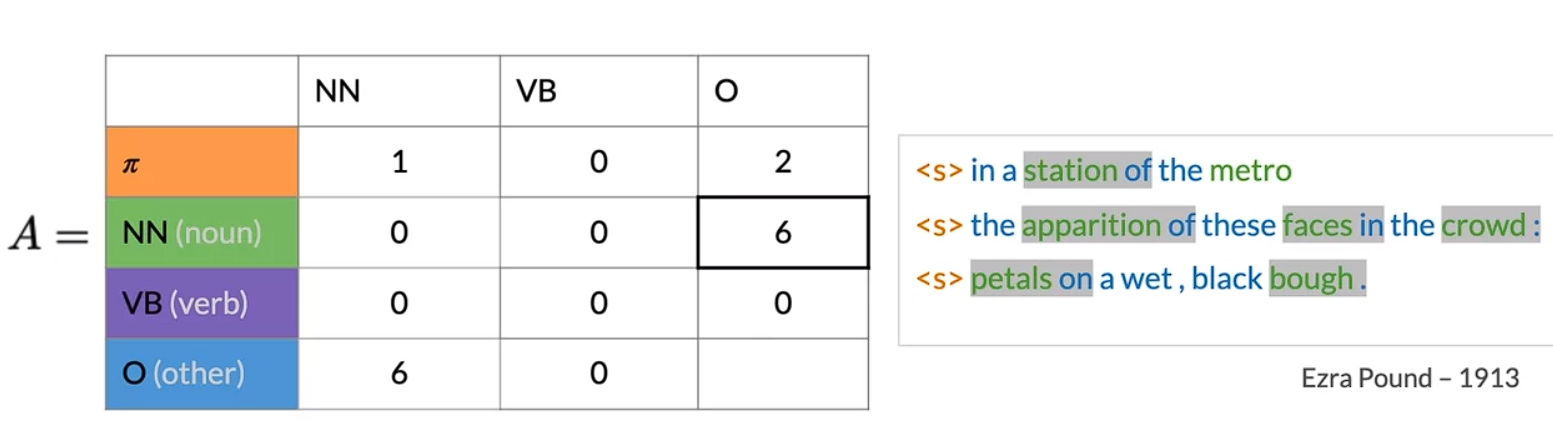


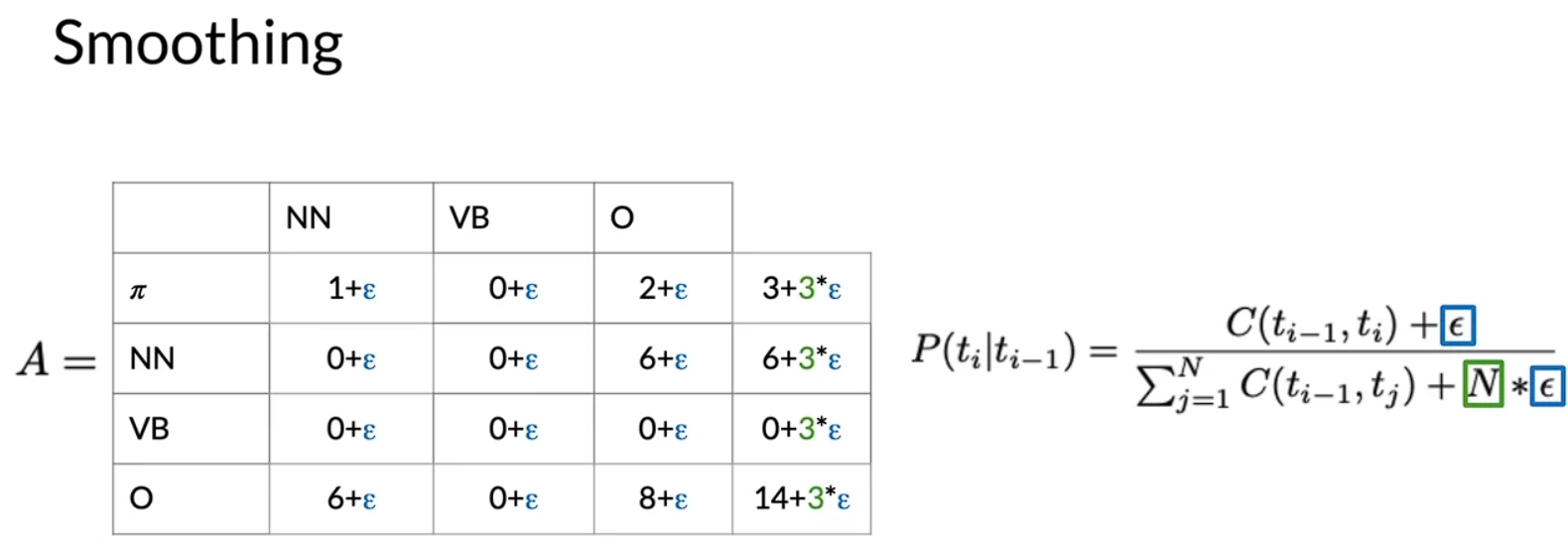
In the transition matrix we will count the number of times tag t(i-1), t(i) show up near each other and divide by the total number of times t(i-1) shows up (which is the same as the number of times it shows up followed by anything else).



Here, C (t(i-1), t(i)) is the count of times tag t(i-1) shows up before tag i.

#### Populating the Transition Matrix

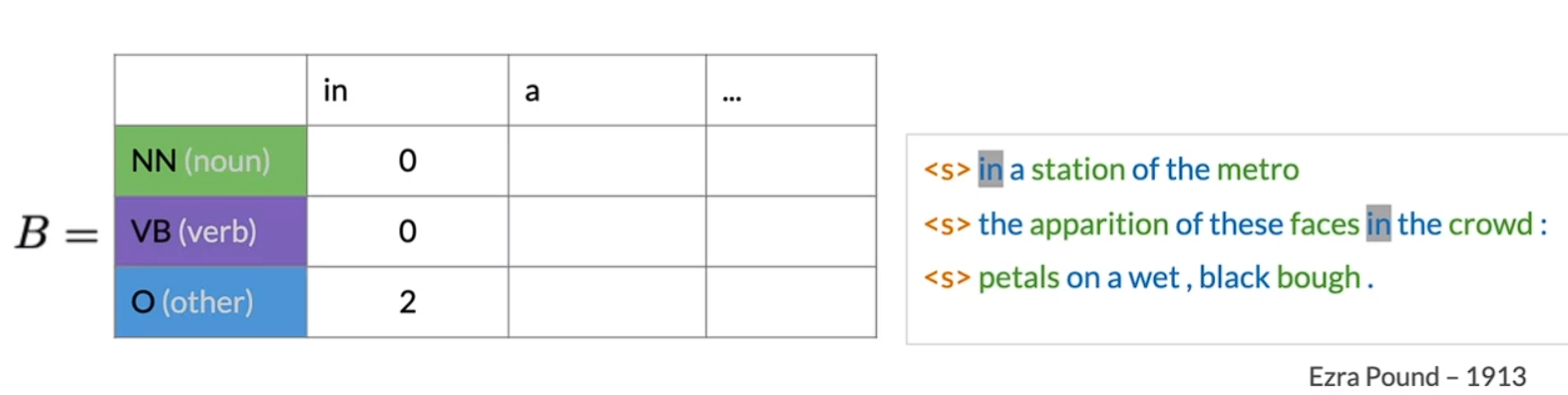




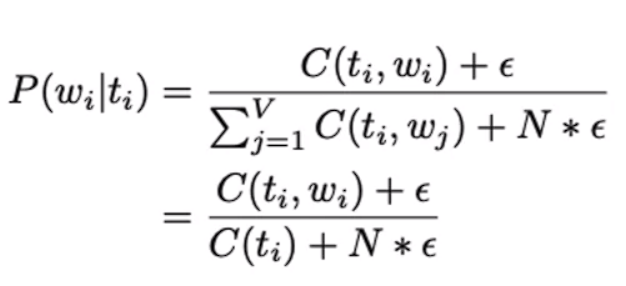
Note: **One more thing before you go, and a real-world example, you might not want to apply smoothing to the initial probabilities in the first row of the transition matrix. That's because if you apply smoothing to that row by adding a small value to possibly zeroed valued entries. You'll effectively allow a sentence to start with any parts of speech tag, including punctuation.**

#### Populating the Emission Matrix





Formula for Smoothing the Emission Matrix:

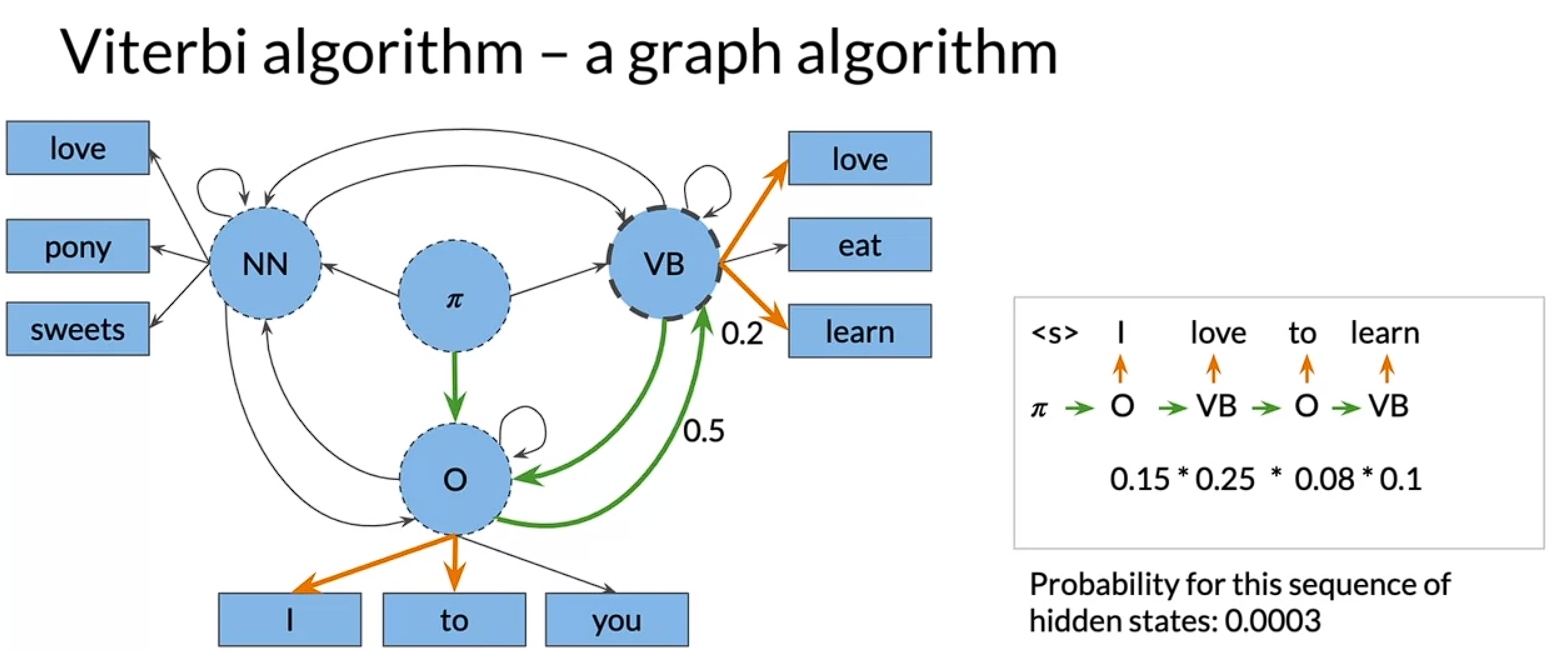


Where, C(t(i), w(i)), is the count associated with how many times the tag t(i) is associated with the word w(i). The epsilon above is the smoothing parameter.

Alphabetical list of part-of-speech tags used in the Penn Treebank Project:

|  |  |  |
| --- | --- | --- |
| No. | Tag | Description |
| 1. | CC | Coordinating conjunction |
| 2. | CD | Cardinal number |
| 3. | DT | Determiner |
| 4. | EX | Existential there |
| 5. | FW | Foreign word |
| 6. | IN | Preposition or subordinating conjunction |
| 7. | JJ | Adjective |
| 8. | JJR | Adjective, comparative |
| 9. | JJS | Adjective, superlative |
| 10. | LS | List item marker |
| 11. | MD | Modal |
| 12. | NN | Noun, singular or mass |
| 13. | NNS | Noun, plural |
| 14. | NNP | Proper noun, singular |
| 15. | NNPS | Proper noun, plural |
| 16. | PDT | Predeterminer |
| 17. | POS | Possessive ending |
| 18. | PRP | Personal pronoun |
| 19. | PRP$ | Possessive pronoun |
| 20. | RB | Adverb |
| 21. | RBR | Adverb, comparative |
| 22. | RBS | Adverb, superlative |
| 23. | RP | Particle |
| 24. | SYM | Symbol |
| 25. | TO | to |
| 26. | UH | Interjection |
| 27. | VB | Verb, base form |
| 28. | VBD | Verb, past tense |
| 29. | VBG | Verb, gerund or present participle |
| 30. | VBN | Verb, past participle |
| 31. | VBP | Verb, non-3rd person singular present |
| 32. | VBZ | Verb, 3rd person singular present |
| 33. | WDT | Wh-determiner |
| 34. | WP | Wh-pronoun |
| 35. | WP$ | Possessive wh-pronoun |
| 36. | WRB | Wh-adverb |

#### Viterbi Algorithm

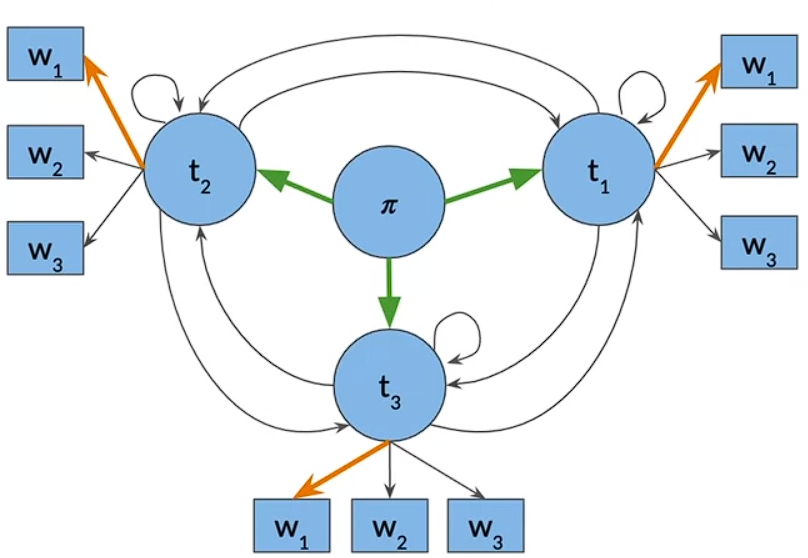


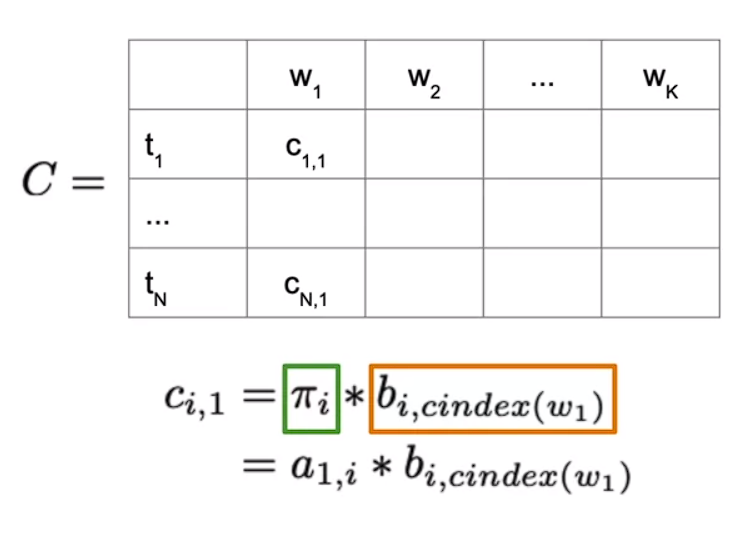
The Viterbi algorithm actually computes several such paths at the same time in order to find the most likely sequence of hidden states. It uses the matrix representation of the hidden Markov model.

Three steps:

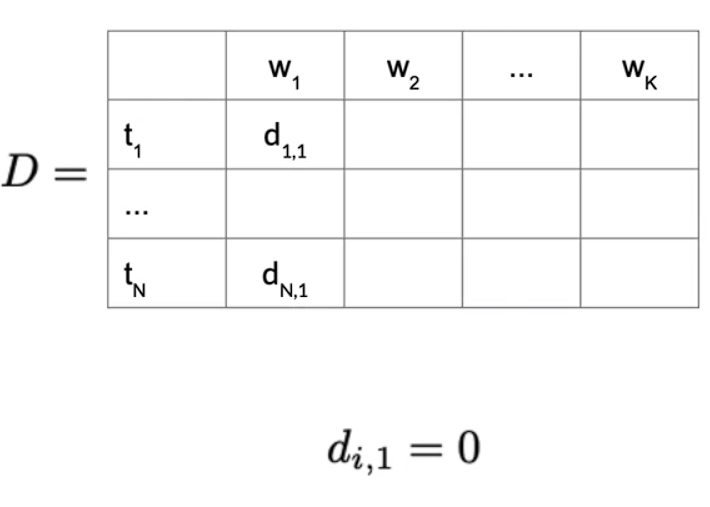
1. Initialization
2. Forward Pass
3. Backward Pass

#### Viterbi: Initialization



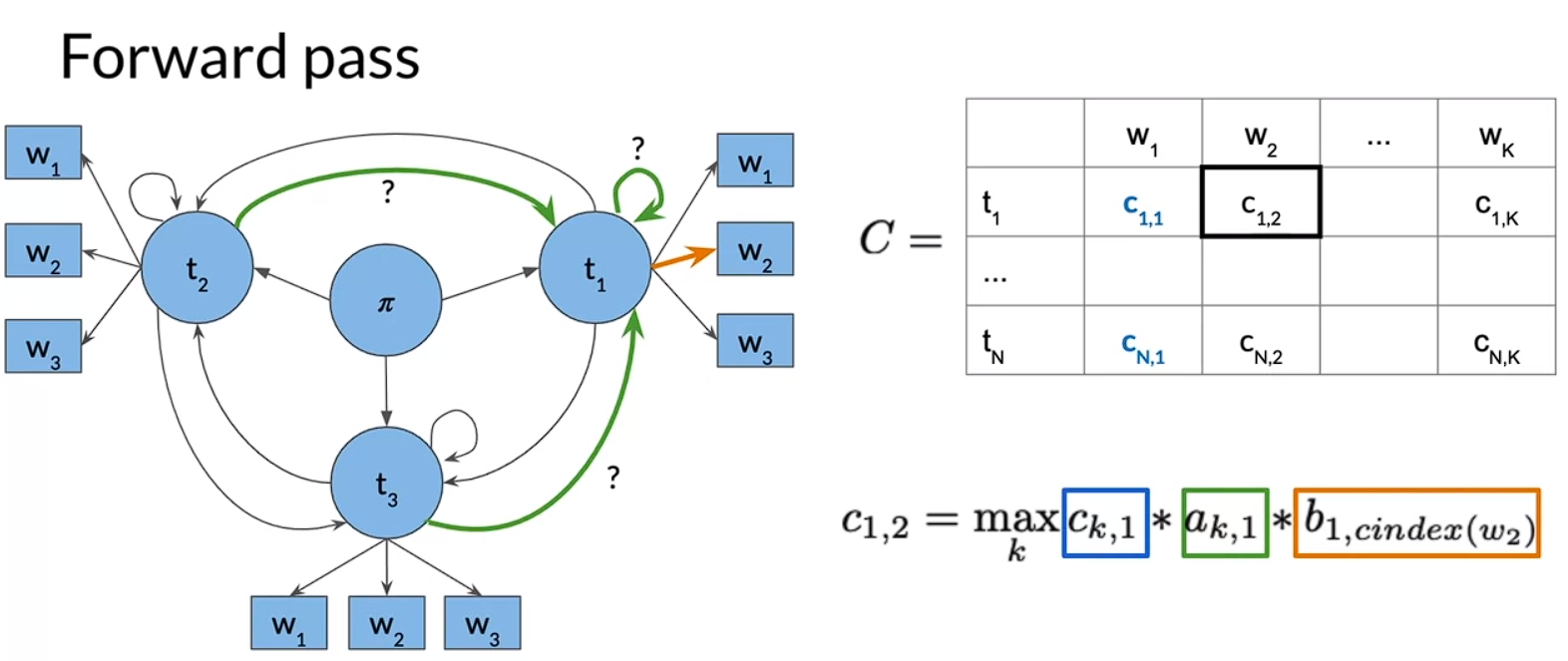


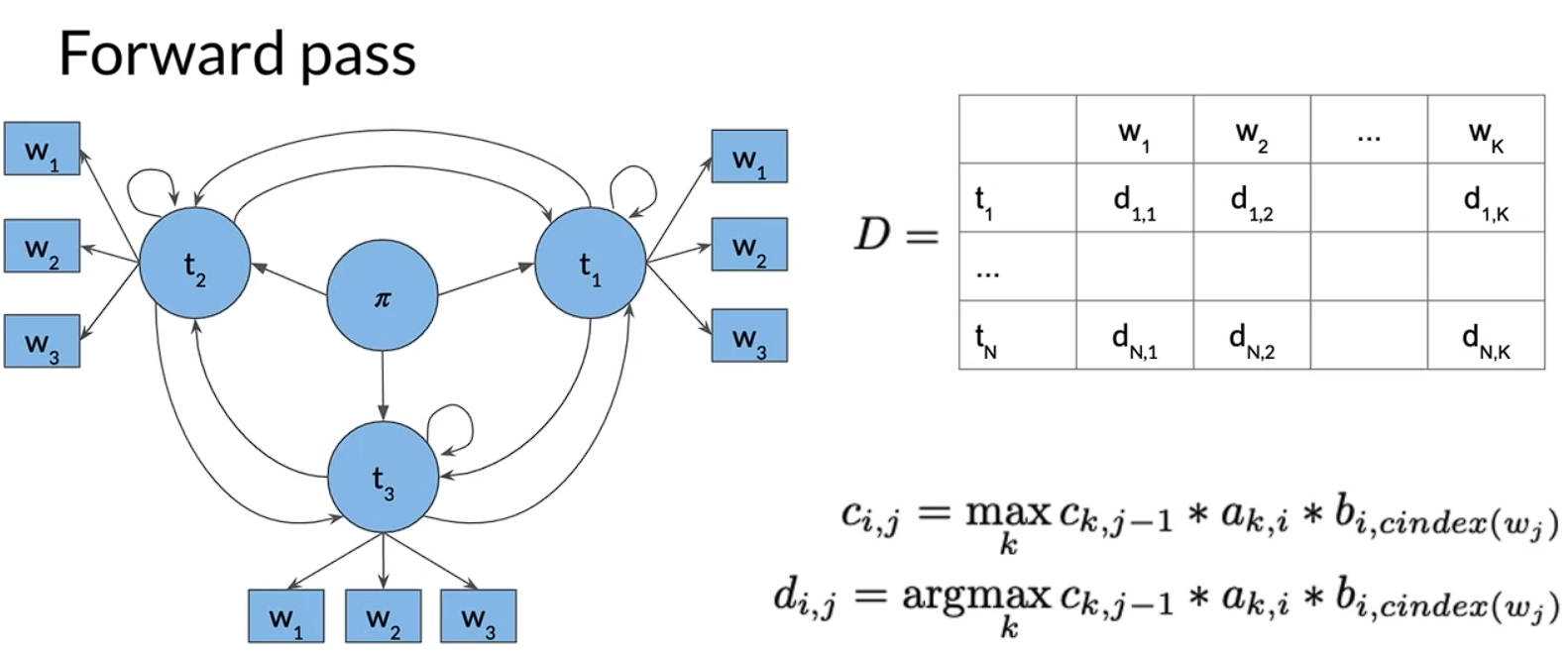
The first column of C represents probability of the transitions from the start states Pi in the graph to the first tag t\_i and word w\_1, meaning we're trying to go from tag 1 to the word w\_1. The first column entries, c\_1,1, are the products of the transition probabilities of the initial states and their respective emission probabilities.



In the D matrix, you store the labels that represent the different states you're traversing when finding the most likely sequence of parts of speech tags for the given sequence of words, w\_1, all the way to w\_k. In the first column, you simply set all entries to zero as there are no preceding parts of speech tags we have traversed.

#### Viterbi: Forward Pass





#### Viterbi: Backward Pass

Example: four tags and five words:

