DD1418
5a: Part-of-speech tagging

Johan Boye, KTH

Recall: Word classes

Words can be divided into classes depending on their use in the language.

- Noun, verb, adjective, adverb, preposition, pronoun, conjunction, interjection, determiner, etc.
- These classes are often called *parts-of-speech* (or *POS tags*).

The *part-of-speech tagging* problem is to assign a POS tag to each word in a text.

Ambiguities

Some words can belong to more than one class, e.g. *like*:

VERB	"*I like her.*"
NOUN	"*He got a like on Facebook.*"
ADJ	"*The portrait is very like.*"
ADV	"*This is, like, crazy!*"
PREP	"*It looks like an accident*"
CONJ	"*He acted like he was all alone*"

Ambiguities

Swedish also has this kind of words, e.g. *var*:

Den gamle mannen visste inte var han var.

Ambiguities

Swedish also has this kind of words, e.g. *var*:

Den	gamle	mannen	visste	inte	var	han	var
DET	ADJ	NOUN	VERB	ADV	ADV	PRON	VERB

Part-of-speech tagging

Retrieve all possible tags from a dictionary, then decide which ones are the most likely, e.g. :

I	*like*	*plays*	*about*	*Bolliwogs*
PRON	VB/NOUN/ADJ/ ADV/PREP/CONJ	VB/NOUN	ADV/PREP/ADJ	?

$$\Downarrow$$

PRON	VB	NOUN	PREP	NOUN

POS tagging: A probabilistic view

Given a sequence of words $w_1 \dots w_n$, we want to find the most probable sequence of tags $t_1 \dots t_n$.

$$\arg\max_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n)$$

What information can we use to do POS tagging?

The word itself *man* is often a noun, more rarely an interjection, even more rarely a verb.

The preceding tags Some combinations are common (noun verb), others are rare (determiner determiner).

Digression: Bayes' theorem

Bayes' theorem expresses how beliefs should be updated in the light of new evidence.

Likelihood

Prior

$$\boxed{P(S|T) = \frac{P(T|S)P(S)}{P(T)}}$$

Posterior

Marginal

Rewrite and simplify

Rewrite using Bayes' theorem:

$$\arg\max_{t_1 \ldots t_n} P(t_1 \ldots t_n | w_1 \ldots w_n) =$$

$$= \arg\max_{t_1 \ldots t_n} \frac{P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)} =$$

$$= \arg\max_{t_1 \ldots t_n} P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)$$

Rewrite and simplify

$$\arg\max_{t_1 \ldots t_n} P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)$$

Markov assumptions: Prior (context model)

- $P(t_1 \ldots t_n) = \prod_{i=1}^{n} P(t_i | t_{i-1})$ (in the bigram case)
- $P(t_1 \ldots t_n) = \prod_{i=1}^{n} P(t_i | t_{i-k+1} \ldots t_{i-1})$ (in the k-gram case)

Likelihood (lexical model)

- $P(w_1 \ldots w_n | t_1 \ldots t_n) = \prod_{i=1}^{n} P(w_i | t_i)$

Maximum likelihood estimation of parameters

Lexical model:

$$P(w_i|t_i) = \frac{c(w_i, t_i)}{c(t_i)} = \frac{\text{\# of times } w_i \text{ has been tagged as } t_i}{\text{\# of tokens tagged as } t_i}$$

Context model (bigram case):

$$P(t_i|t_{i-1}) = \frac{c(t_{i-1} t_i)}{c(t_{i-1})} = \frac{\text{\# of times a } t_{i-1} \text{ is followed by a } t_i}{\text{\# of tokens tagged as } t_{i-1}}$$

Decoding problem

Find the most probable tagging (bigram case):

$$\arg\max_{t_1 \ldots t_n} \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)$$

k-gram case:

$$\arg\max_{t_1 \ldots t_n} \prod_{i=1}^{n} P(t_i|t_{i-k+1} \ldots t_{i-1})P(w_i|t_i)$$

DD1418
5b: Hidden Markov Models

Johan Boye, KTH

POS tagging: A probabilistic view

Given a sequence of words $w_1 \dots w_n$, we want to find the most probable sequence of POS tags $t_1 \dots t_n$.

$$\arg\max_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n)$$

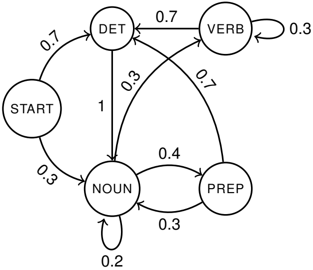
The POS tags $t_1 \dots t_n$ are the **hidden states**, and the words $w_1 \dots w_n$ are the **observations**.

Hidden Markov Models

A Hidden Markov Model (HMM) consists of:

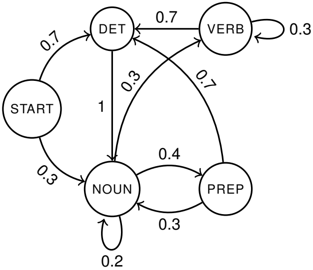
- a set of (hidden) states $\{q_1, \ldots, q_m\}$
- the transition probability matrix A
- a set of observations $\{o_1, \ldots, o_k\}$.
- the observation probability matrix B

Example: A first-order Markov chain for POS tags



$$P(\text{START NOUN VERB DET NOUN}) = 0.3 \times 0.3 \times 0.7 \times 1$$

Example



$$
\begin{array}{lll}
P(\mathit{flies}|N) = 10^{-6} & P(\mathit{flower}|N) = 10^{-5} & P(\mathit{like}|N) = 10^{-7} \\
P(\mathit{flies}|V) = 10^{-5} & P(\mathit{flower}|V) = 10^{-8} & P(\mathit{like}|\mathit{Prep}) = 10^{-5} \\
P(a|N) = 10^{-8} & P(a|DET) = 0.36 & P(\mathit{like}|V) = 10^{-4}
\end{array}
$$

Find the POS tags for **flies like a flower**.

Viterbi algorithm

We want to find the most probable sequence of hidden states.

A naive algorithm that enumerates all the possible sequences and computes their probability would have exponential complexity.

However, the **Viterbi** algorithm finds the solution quicker (quadratic complexity), by storing intermediate results.

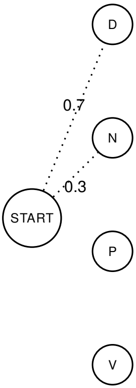
This is an example of **dynamic programming** (just as the algorithm in assignment 1).

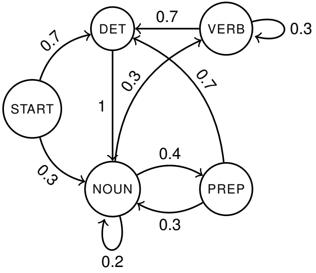
The Viterbi matrix

$\mathtt{v}(\mathtt{t}, \mathtt{k}) =$ the probability of being in state q_k after having seen the first t observations, and passing through the most probable state sequence $q_{i_1} \ldots q_{i_{t-1}}$.

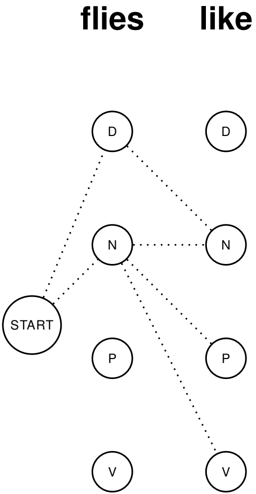
Viterbi trellis

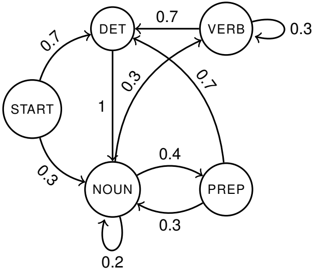
flies



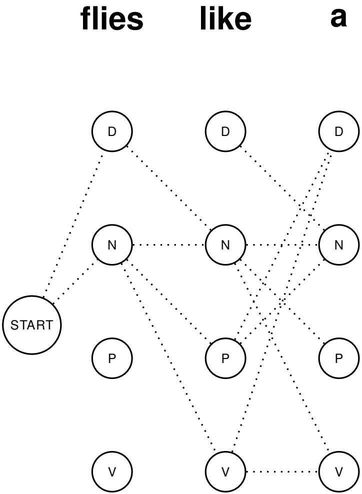


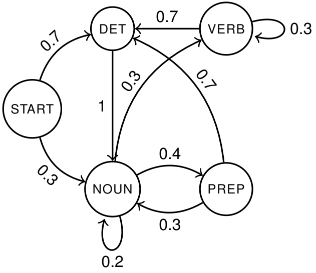
Viterbi trellis



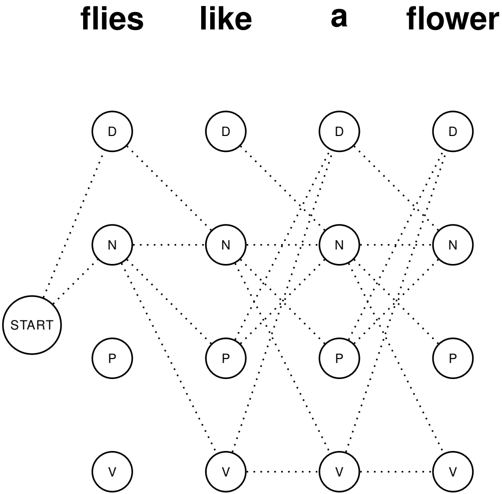


Viterbi trellis





Viterbi trellis



Viterbi algorithm

Bigram case:

$$\mathtt{v}(0,\mathtt{k}) = \mathtt{p}(\mathtt{q_k}|\mathtt{START})\mathtt{p}(\mathtt{o_0}|\mathtt{q_k})$$

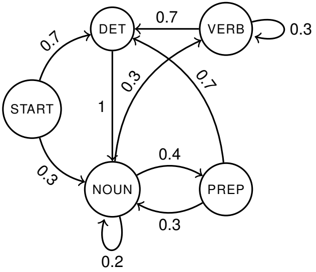
$$\mathtt{v}(\mathtt{t},\mathtt{k}) = \max_{\mathtt{i}}[\,\mathtt{v}(\mathtt{t}-1,\mathtt{i})\,\mathtt{p}(\mathtt{q_k}|\mathtt{q_i})\,\mathtt{p}(\mathtt{o_t}|\mathtt{q_k})\,]$$

$$\mathtt{backptr}(\mathtt{t},\mathtt{k}) = \arg\max_{\mathtt{i}}\,[\,\mathtt{v}(\mathtt{t}-1,\mathtt{i})\,\mathtt{p}(\mathtt{q_k}|\mathtt{q_i})\,\mathtt{p}(\mathtt{o_t}|\mathtt{q_k})\,]$$

DD1418:
5c: Viterbi decoding

Johan Boye, KTH

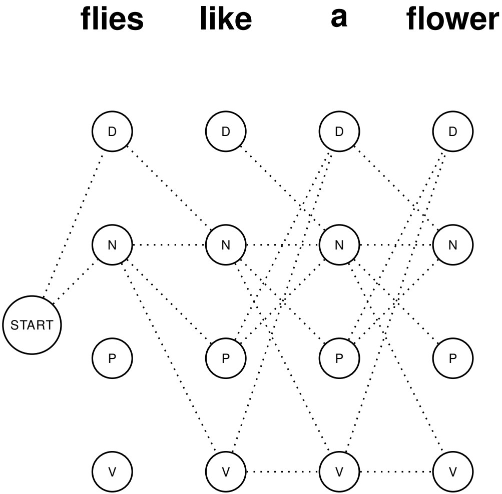
Example



$$P(flies|N) = 10^{-6} \quad P(flower|N) = 10^{-5} \quad P(like|N) = 10^{-7}$$
$$P(flies|V) = 10^{-5} \quad P(flower|V) = 10^{-8} \quad P(like|Prep) = 10^{-5}$$
$$P(a|N) = 10^{-8} \quad P(a|DET) = 0.36 \quad P(like|V) = 10^{-4}$$

Find the POS tags for **flies like a flower.**

Viterbi trellis



Viterbi algorithm

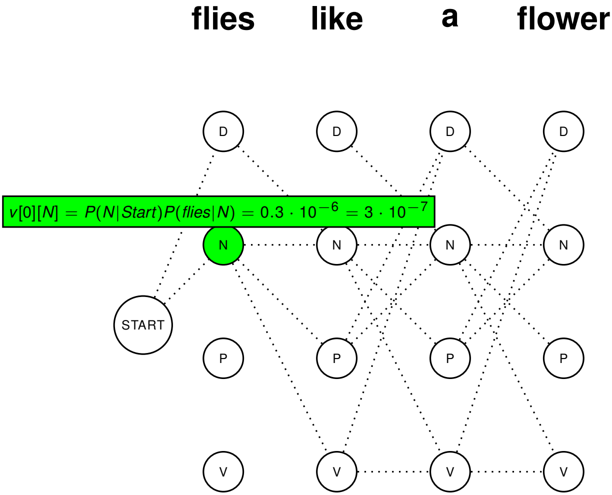
Bigram case:

$$\mathtt{v}(0,\mathtt{k}) = \mathtt{p}(\mathtt{q}_\mathtt{k}|\mathtt{START})\mathtt{p}(\mathtt{o}_0|\mathtt{q}_\mathtt{k})$$

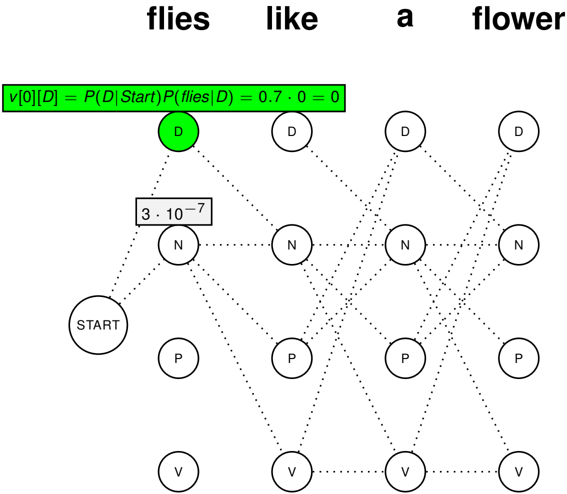
$$\mathtt{v}(\mathtt{t},\mathtt{k}) = \max_{\mathtt{i}}[\,\mathtt{v}(\mathtt{t}-1,\mathtt{i})\,\mathtt{p}(\mathtt{q}_\mathtt{k}|\mathtt{q}_\mathtt{i})\,\mathtt{p}(\mathtt{o}_\mathtt{t}|\mathtt{q}_\mathtt{k})\,]$$

$$\mathtt{backptr}(\mathtt{t},\mathtt{k}) = \arg\max_{\mathtt{i}}\,[\,\mathtt{v}(\mathtt{t}-1,\mathtt{i})\,\mathtt{p}(\mathtt{q}_\mathtt{k}|\mathtt{q}_\mathtt{i})\,\mathtt{p}(\mathtt{o}_\mathtt{t}|\mathtt{q}_\mathtt{k})\,]$$

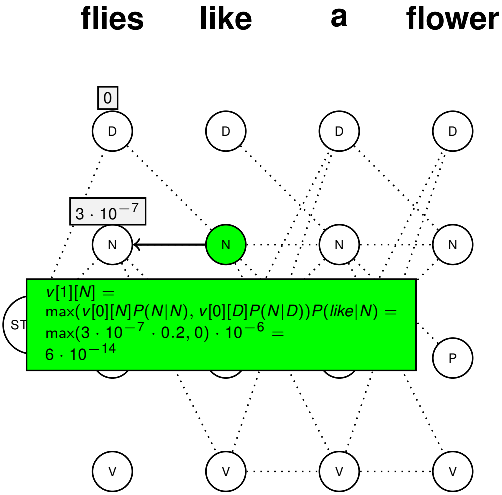
Viterbi decoding



Viterbi decoding

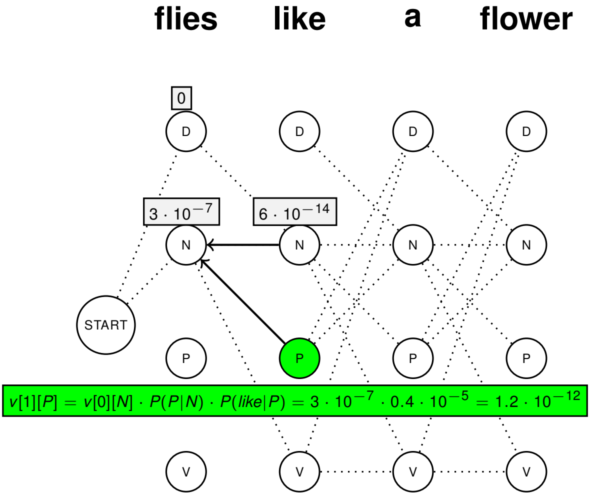


Viterbi decoding

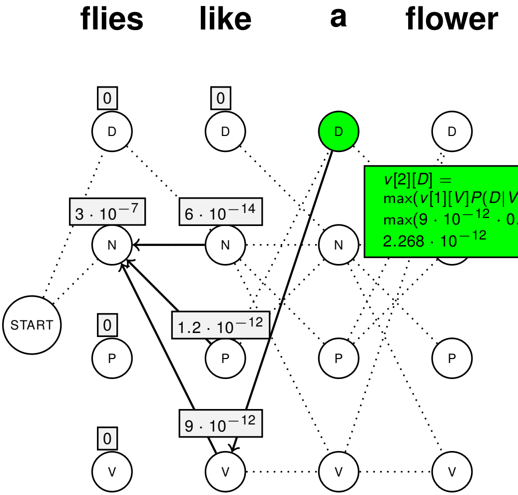


$v[1][N] =$
$\max(v[0][N]P(N|N), v[0][D]P(N|D))P(like|N) =$
$\max(3 \cdot 10^{-7} \cdot 0.2, 0) \cdot 10^{-6} =$
$6 \cdot 10^{-14}$

Viterbi decoding

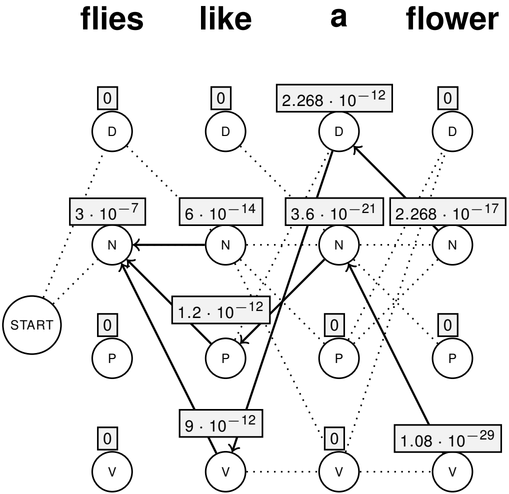


Viterbi decoding

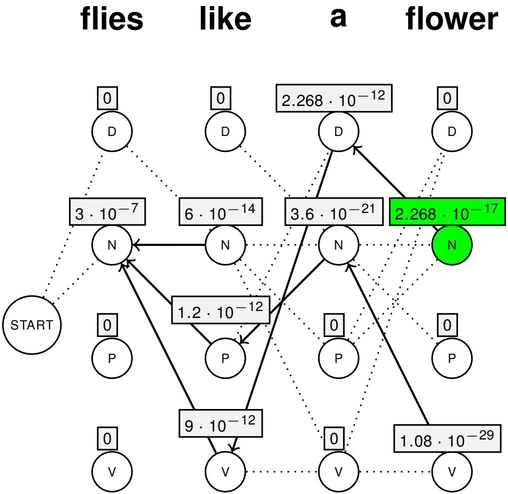


$v[2][D] =$
$\max(v[1][V]P(D|V), v[1][P]P(D|P))P(a|D) =$
$\max(9 \cdot 10^{-12} \cdot 0.7, 1.2 \cdot 10^{-12} \cdot 0.7) \cdot 0.36 =$
$2.268 \cdot 10^{-12}$

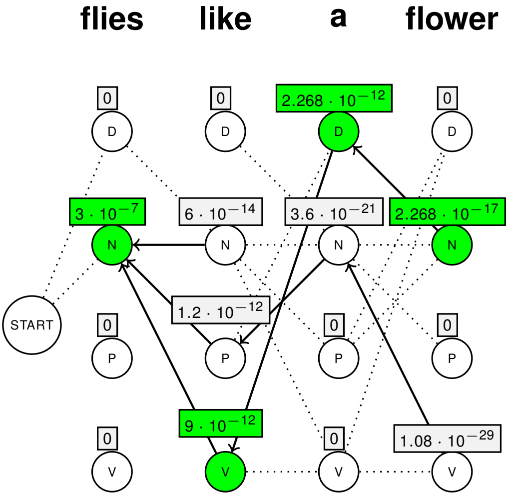
Viterbi decoding



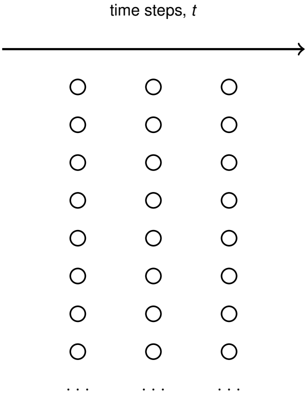
Viterbi decoding



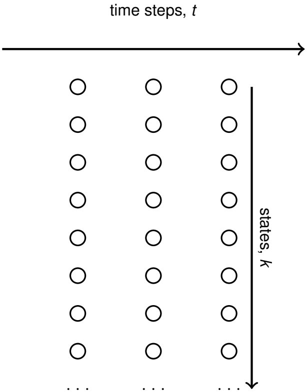
Viterbi decoding



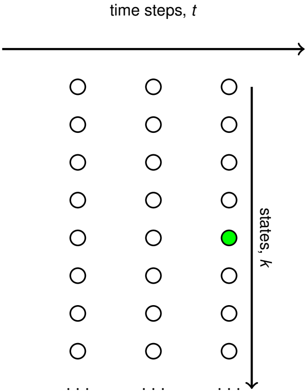
The Viterbi loop, bigram case



The Viterbi loop, bigram case

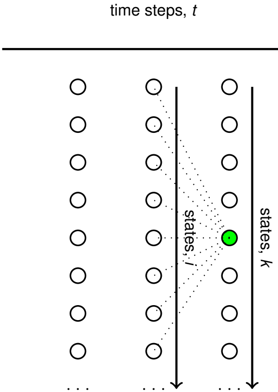


The Viterbi loop, bigram case



$\mathrm{v[t][k]} =$

The Viterbi loop, bigram case



v[t][k] =

Viterbi decoding, data structures

Data structures in the bigram case:

- `a[i][k]` = probability of going from q_i to q_k
- `b[i][k]` = probability of state q_k emitting o_i
- `v[t][k]` = probability of being in state q_k after having seen the first t observations, and passing through the most probable state sequence $q_{i_1} \dots q_{i_{t-1}}$.

DD1418:
5d: More on Viterbi decoding

Johan Boye, KTH

Viterbi algorithm

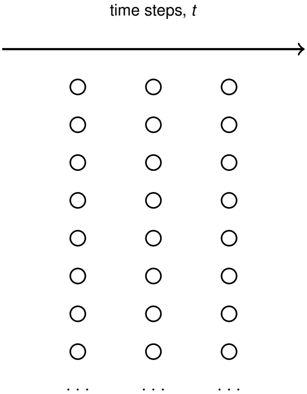
Trigram case:

$$\mathtt{v}(0, \mathtt{START}, \mathtt{k}) = \mathtt{p}(\mathtt{q}_\mathtt{k} | \mathtt{START}\ \mathtt{START}) \mathtt{p}(\mathtt{o}_0 | \mathtt{q}_\mathtt{k})$$

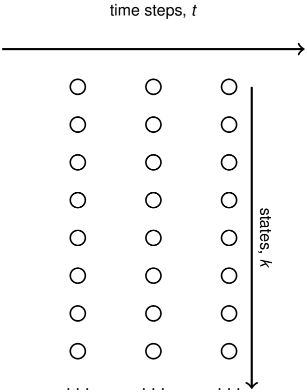
$$\mathtt{v}(\mathtt{t}, \mathtt{j}, \mathtt{k}) = \max_{\mathtt{i}} [\ \mathtt{v}(\mathtt{t} - 1, \mathtt{i}, \mathtt{j})\ \mathtt{p}(\mathtt{q}_\mathtt{k} | \mathtt{q}_\mathtt{i}\, \mathtt{q}_\mathtt{j})\ \mathtt{p}(\mathtt{o}_\mathtt{t} | \mathtt{q}_\mathtt{k})\ ]$$

$$\mathtt{backptr}(\mathtt{t}, \mathtt{j}, \mathtt{k}) = \arg\max_{\mathtt{i}} [\ \mathtt{v}(\mathtt{t} - 1, \mathtt{i}, \mathtt{j})\ \mathtt{p}(\mathtt{q}_\mathtt{k} | \mathtt{q}_\mathtt{i}\, \mathtt{q}_\mathtt{j})\ \mathtt{p}(\mathtt{o}_\mathtt{t} | \mathtt{q}_\mathtt{k})\ ]$$

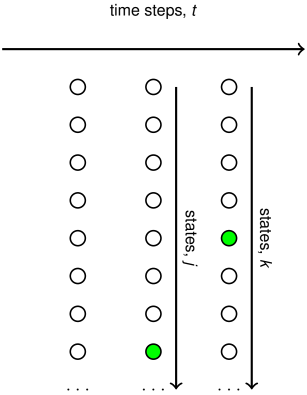
The Viterbi loop, trigram case



The Viterbi loop, trigram case

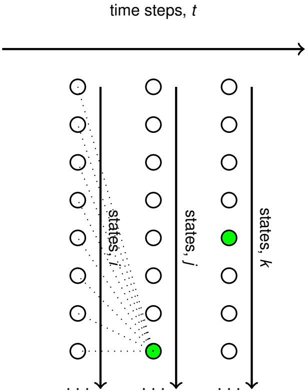


The Viterbi loop, trigram case



$v[t][j][k] =$

The Viterbi loop, trigram case



$\mathtt{v[t][j][k]} =$

Viterbi decoding, data structures

Data structures in the trigram case:

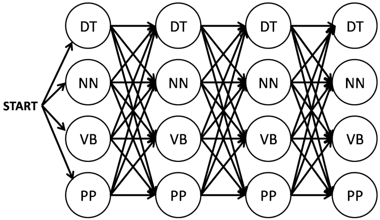
- `a[i][j][k]` = probability of going to q_k given that the preceding state was q_j, and the state before that was q_i
- `b[i][k]` = probability of state q_k emitting o_i
- `v[t][j][k]` = probability of being in state q_k after having seen the first t observations, and passing through the most probable state sequence $q_{i_1} \ldots q_{i_{t-2}} q_j$.

Smoothing

Normally the context model is **smoothed**, just as in the case of language models.

That is: Even if we have never seen the combination DET DET in our data, we will still give $P(\texttt{DET}|\texttt{DET})$ a non-zero probability.

This means that we have to consider all possible state transitions in every step of the Viterbi algorithm.



Beam search

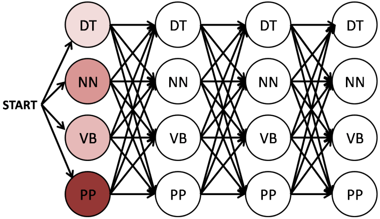
If we consider all possible state transitions, the Viterbi algorithm can still be quite time-consuming.

Common solution: **Beam search**

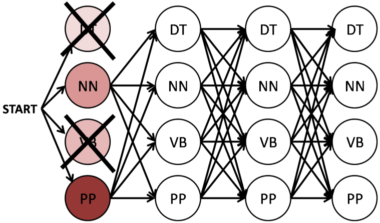
In every time step t, only consider the β states with the highest v-value. Paths ending at other states at time step t are terminated.

The parameter β is called the **beam width.**

Beam search (example)

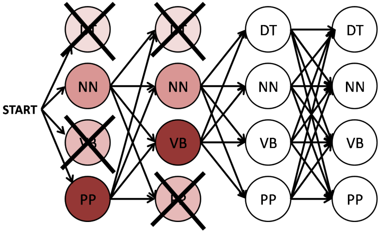


1a. In time step 1, the state `PP` is best, `NN` is second best.



1b. Remove the other states in step 1, and their outbound transitions.

Beam search (example)



2. In time step 2, `VB` and `NN` are best. Remove `DT` and `PP`.

etc.