Natural Language Processing

Lecture 5: Sequence Labeling with Hidden Markov Models. Part-of-Speech Tagging.

9/20/2024

COMS W4705
Daniel Bauer

Garden-Path Sentences

- *The horse raced past the barn.*
- *The horse raced past the barn fell.*
- *The old dog the footsteps of the young.*
- *The cotton clothing is made of grows in Egypt.*

Garden-Path Sentences

- Why does this happen?

past tense verb

VBD **???**

The horse ***raced*** *past the barn fell*

- **raced** can be a past tense verb or a a past participle (indicating passive voice).

- The verb interpretation is more likely before *fell* is read.

Garden-Path Sentences

- Why does this happen?

past participle

VBN **VBD**

*[The horse **raced** past the barn] fell*

NP

- **raced** can be a past tense verb or a a past participle (indicating passive voice).

- Once *fell* is read, the verb interpretation is impossible.

Garden-Path Sentences

- Why does this happen?

adjective

JJ **NN**

[The old dog] [the footsteps of the young]

NP **NP**

- **dog** can be a noun or a verb (plural, present tense)

Garden-Path Sentences

- Why does this happen?

NNS VB

[The old] dog [the footsteps of the young]

NP NP

- **dog** can be a noun or a verb (plural, present tense)

Parts-of-Speech

- Classes of words that behave alike:
  - Appear in similar contexts.
  - Perform a similar grammatical function in the sentence.
  - Undergo similar morphological transformations.
  - Have similar meaning.

- ~9 traditional parts-of-speech:
  - noun, pronoun, determiner, adjective, verb, adverb, preposition, conjunction, interjection

Syntactic Ambiguities and Parts-of-Speech

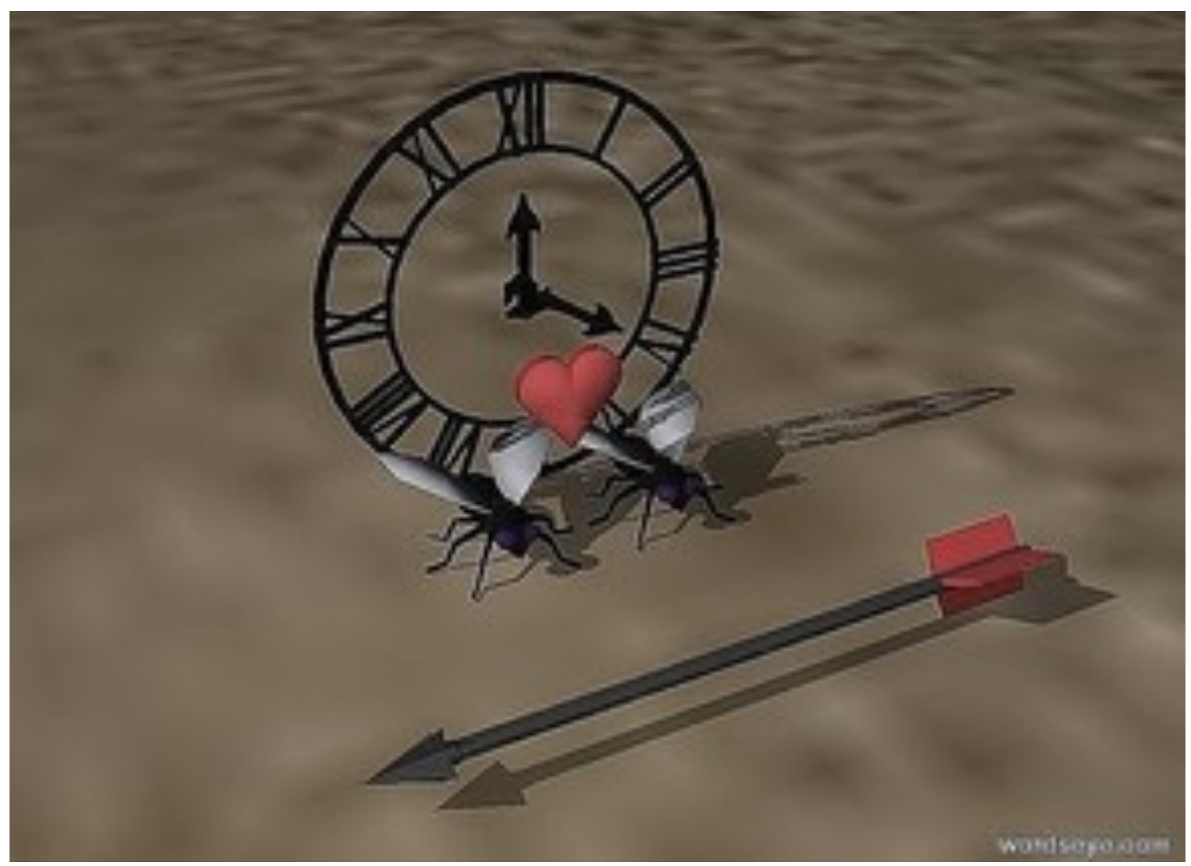
N / V?	**N / V?**	**V / Preposition?**	
Time	*flies*	*like*	*an arrow.*

Syntactic Ambiguities and Parts-of-Speech

N N V

- [*Time* *flies]* *like* *an arrow.*

NP



Why do we need P.O.S.?

- Interacts with most levels of linguistic representation.
- Speech processing:
  - *lead (V) vs. lead (N).*
  - ***in**sult, in**sult***
  - ***ob**ject, ob**ject***
  - *con**tent**, **con**tent*
- Syntactic parsing
- …
- P.O.S. tag-set should contain morphological and maybe syntactic information.

Penn Treebank Tagset

CC	Coordinating conjunction	PRP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential *there*	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WP	Wh-pronoun
PDT	Predeterminer	WP$	Possessive wh-pronoun
POS	Possessive ending	WRB	Wh-adverb
PRP	Personal pronoun		

plus punctuation symbols

P.O.S. Tagsets

- Tagset is language specific.
- Some language capture more morphological information which should be reflected in the tag set.
- “Universal Part Of Speech Tags?”
  - Petrov et al. 2011: Mapping of 25 language specific tag-sets to a common set of 12 universal tags
  - "Universal Dependencies" framework uses 17 tags. https://universaldependencies.org/u/pos/

Part-of-Speech Tagging

- Goal: Assign a part-of-speech label to each word in a sentence.

DT	***NN***	***VBD***	***DT***	***NNS***	***IN***	***DT***	***NN***	***.***
the	*koala*	*put*	*the*	*keys*	*on*	*the*	*table*	*.*

- This is an example of a **sequence labeling** task.

- Think of this as a translation task from a sequence of words $(w_1, w_2, \ldots, w_n) \in V^*$, to a sequence of tags $(t_1, t_2, \ldots, t_n) \in T^*$.

Part-of-Speech Tagging

- Goal: Translate from a sequence of words $(w_1, w_2, \ldots, w_n) \in V^*$, to a sequence of tags $(t_1, t_2, \ldots, t_n) \in T^*$.

- NLP is full of translation problems from one structure to another. Basic solution:

  - For each translation step:

    1. Construct search space of possible translations.

    2. Find best paths through this space (decoding) according to some performance measure.

Bayesian Inference for Sequence Labeling

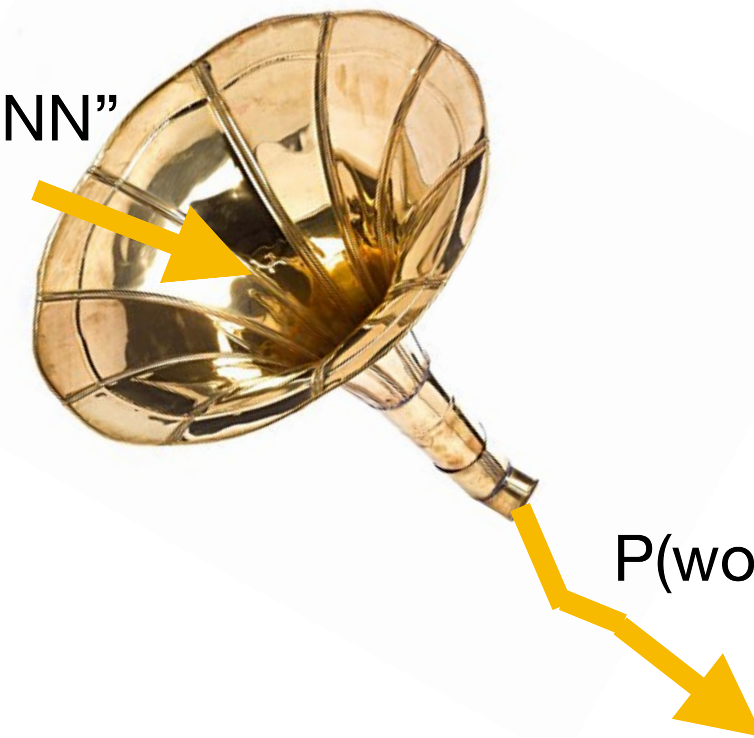
- Recall Bayesian Inference (Generative Models): Given some observation, infer the value of some *hidden variable.* (see Naive Bayes')

- We can apply this approach to sequence labeling:

  - Assume each word w_i in the observed sequence $(w_1, w_2, \ldots, w_n) \in V^*$ was generated by some hidden variable t_i.

  - Infer the most likely sequence of hidden variables given the sequence of observed words.

Noisy Channel Model



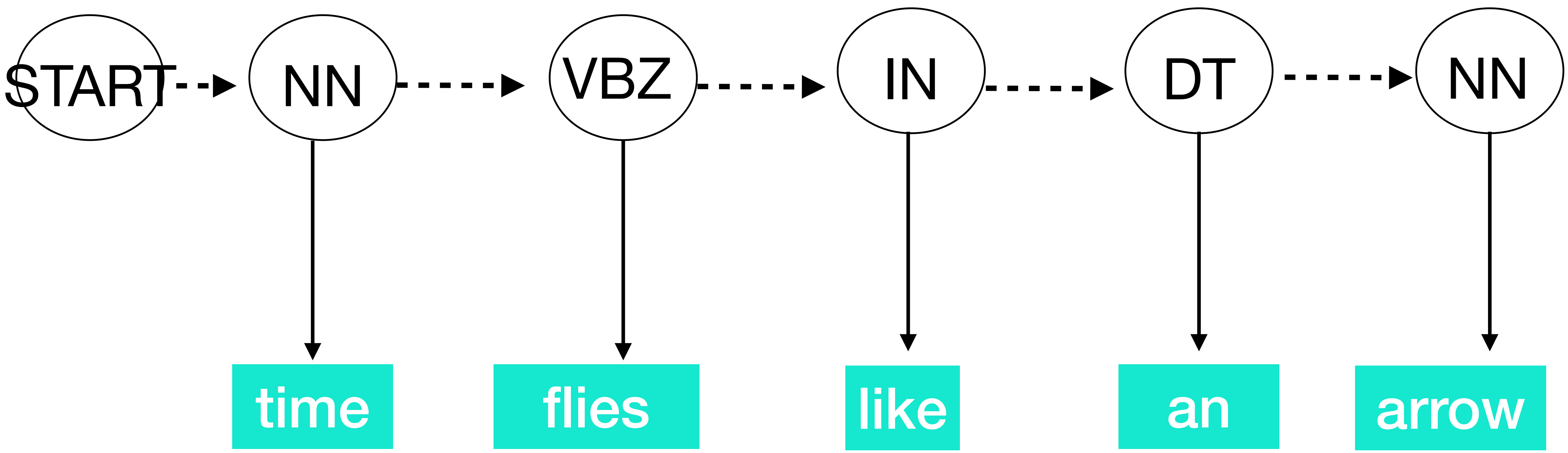
- Goal: figure out what the original input to the the channel was. Use Bayes’ rule:

$$\arg\max_{tags} P(tags|words) = \arg\max_{tags} \frac{P(tags) \cdot P(word|tags)}{P(words)}$$

- This model is used widely (speech recognition, MT)

Hidden Markov Models (HMMs)

- Generative (Bayesian) probability model.
  Observations: sequences of words.
  Hidden states: sequence of part-of-speech labels.

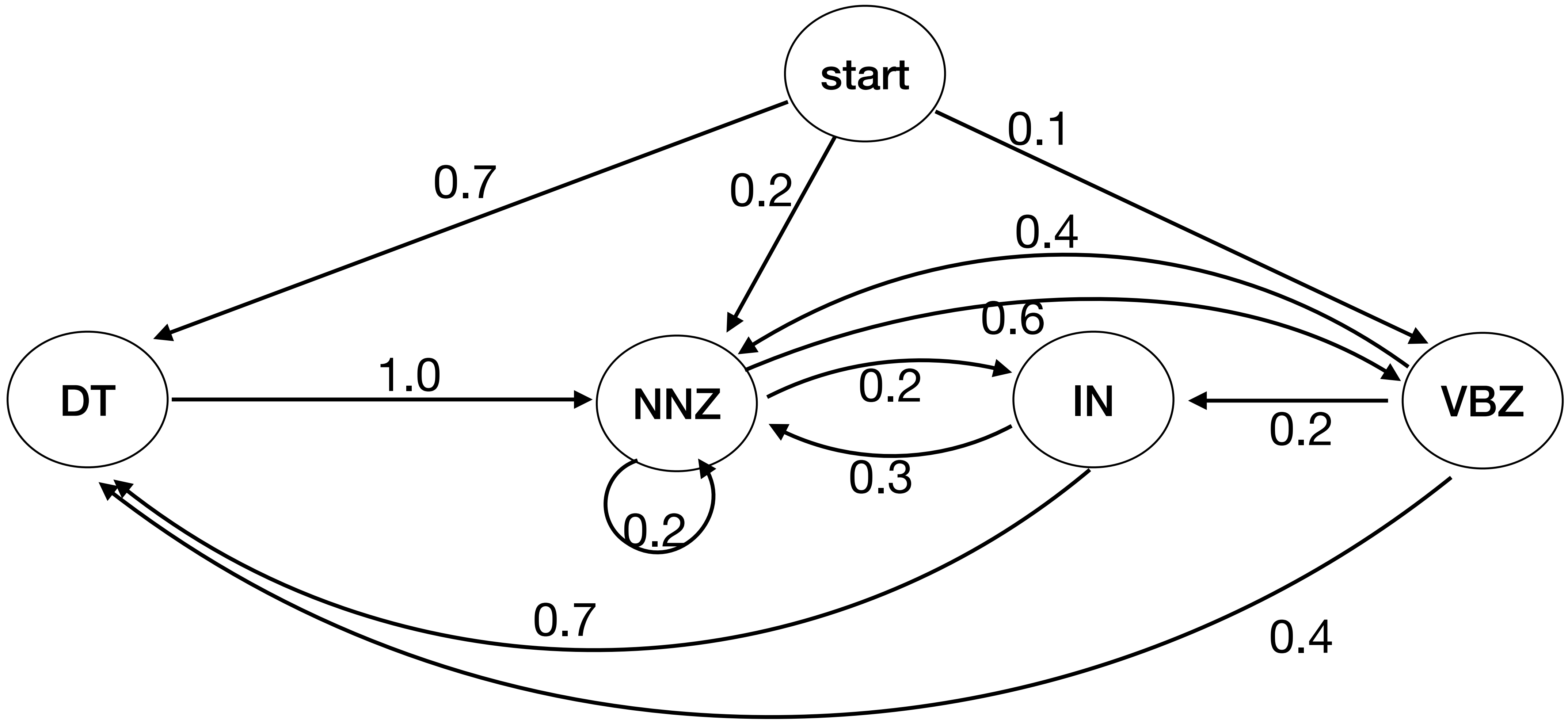


- Hidden sequence is generated by an n-gram language model (typically a bi-gram model)

$t_0 = START$

$$P(t_1, t_2, \ldots, t_n) = \prod_{i=1}^{n} P(t_i | t_{i-1})$$

Markov Chains



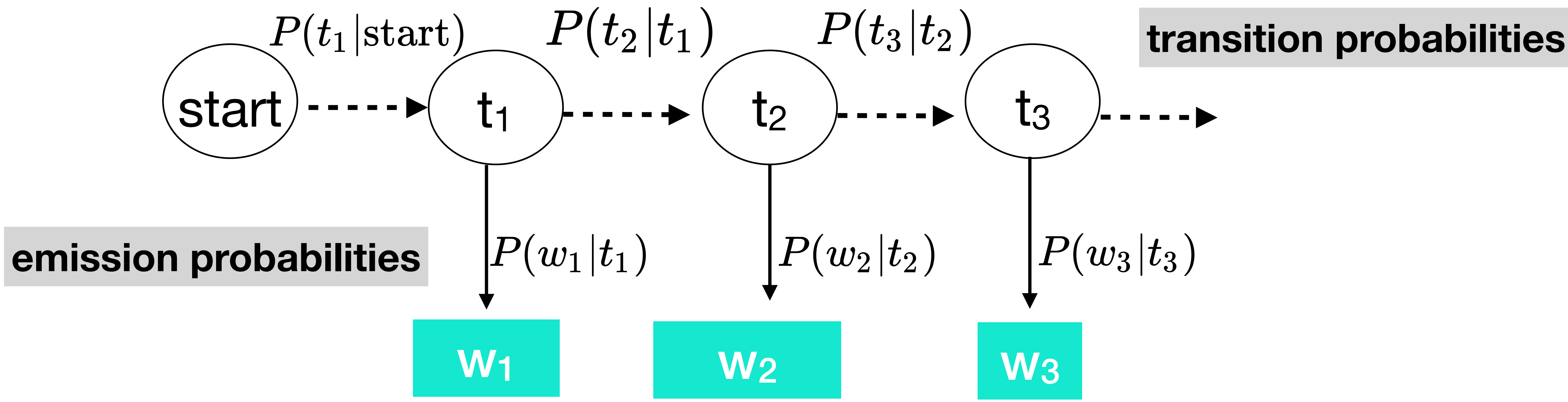
- A **Markov chain** is a sequence of random variables $X_1, X_2, \ldots$
- The domain of these variables is a set of states.
- **Markov assumption:** Next state depends only on current state.

$$P(X_{n+1}|X_1, X_2, \ldots, X_n) = P(X_{n+1}|X_n)$$

- This is a special case of a weighted finite state automaton (WFSA).

Hidden Markov Models (HMMs)

- There are two types of probabilities:
  Transition probabilities and Emission Probabilities.



$$P(t_1, t_2, \ldots, t_n, w_1, w_2, \ldots, w_n) =$$

$$P(t_1|start)P(w_1|t_1)P(t_2|t_1)P(w_2|t_2)\cdots P(t_n|t_{n-1})P(w_n|t_n)$$

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

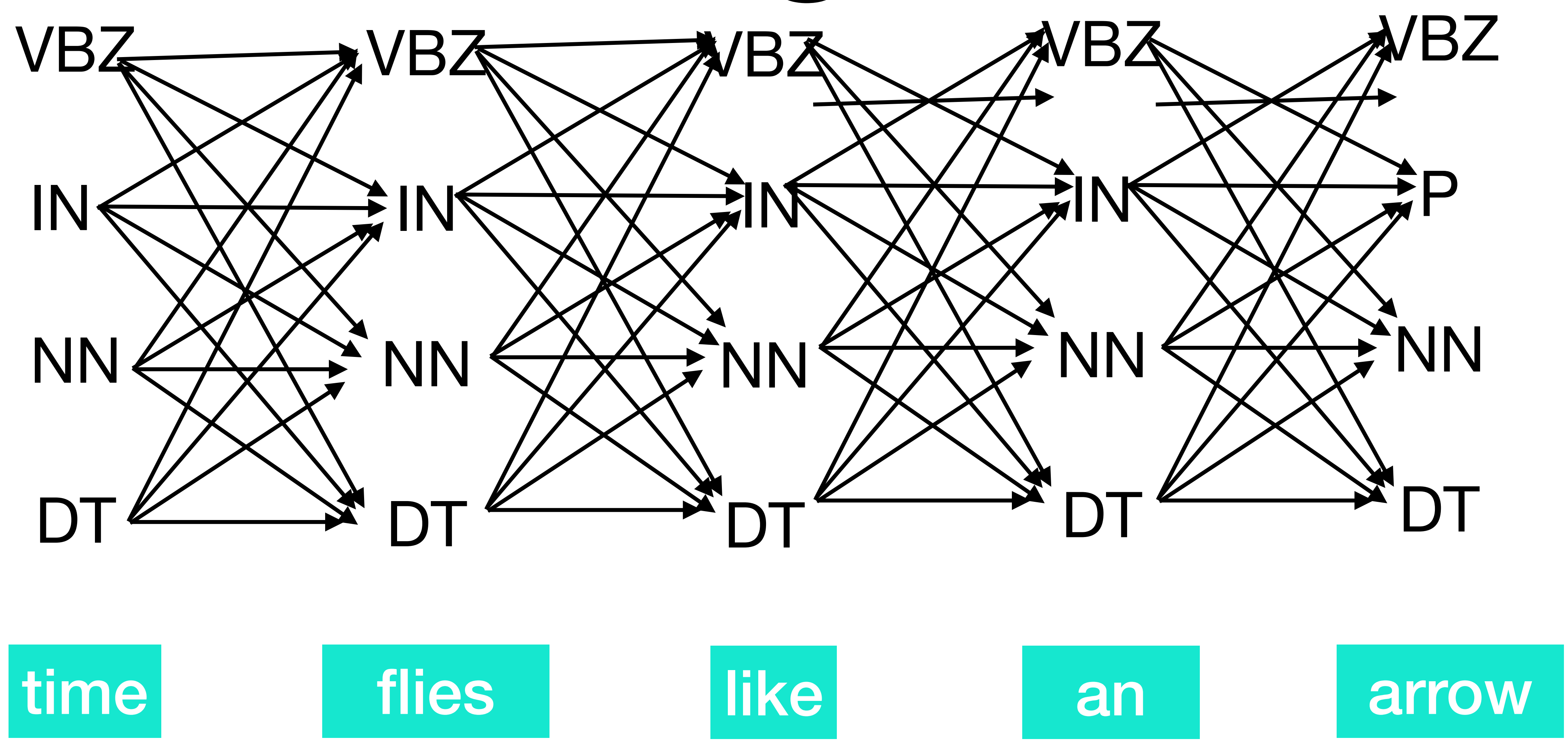
Important Tasks on HMMs

- **Decoding:** Given a sequence of words, find the *most likely* tag sequence.
  (Bayesian inference using *Viterbi algorithm*).

- **Evaluation:** Given a sequence of words, find the *total probability for this word sequence* given an HMM.
  Note that we can view the HMM as another type of language model. (Forward algorithm)

- **Training:** Estimate emission and transition probabilities from training data. (MLE, Forward-Backward a.k.a Baum-Welch algorithm)

Decoding HMMs



Goal: Find the path with the highest total probability (given the words)

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

There are d^n paths for n words and d tags.

Emission Probabilities

- P(time | VB) = 0.2
  P(flies | VB) = 0.3
  P(like | VB) = 0.5

- P(time | NN) = 0.3
  P(flies | NN) = 0.2
  P(arrow | NN) = 0.5

- P(like | IN) = 1.0

- P(an | DT) = 1.0

(these are used in the example on the next slide)

Viterbi Algorithm

.1 x .2 = .02	VBZ	VBZ	VBZ	VBZ	VBZ
0	IN	IN	IN	IN	IN
.2 x .3 = .06	NN	NN	NN	NN	NN
0	DT	DT	DT	DT	DT
	time	flies	like	an	arrow

- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm

.02 VBZ VBZ VBZ

IN

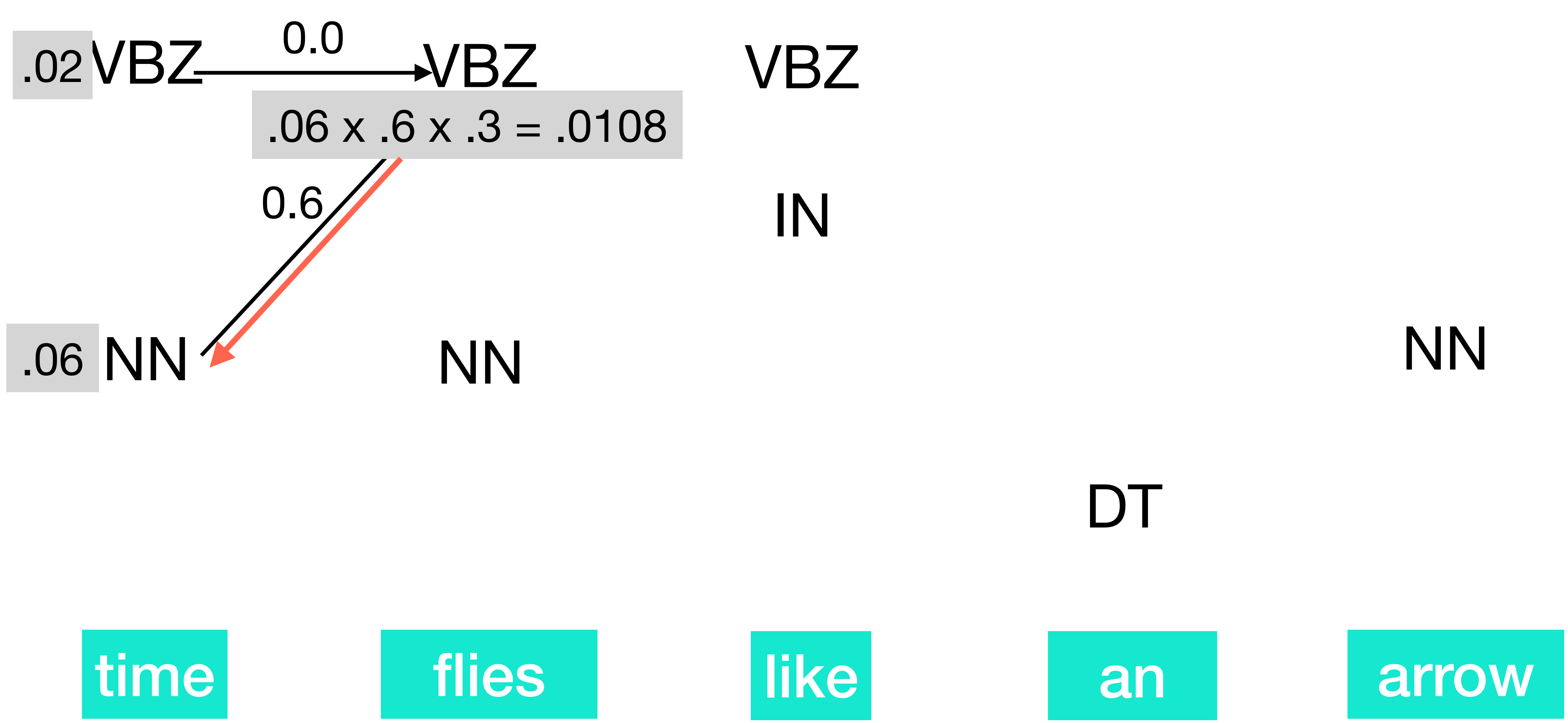
.06 NN NN NN

DT

time flies like an arrow

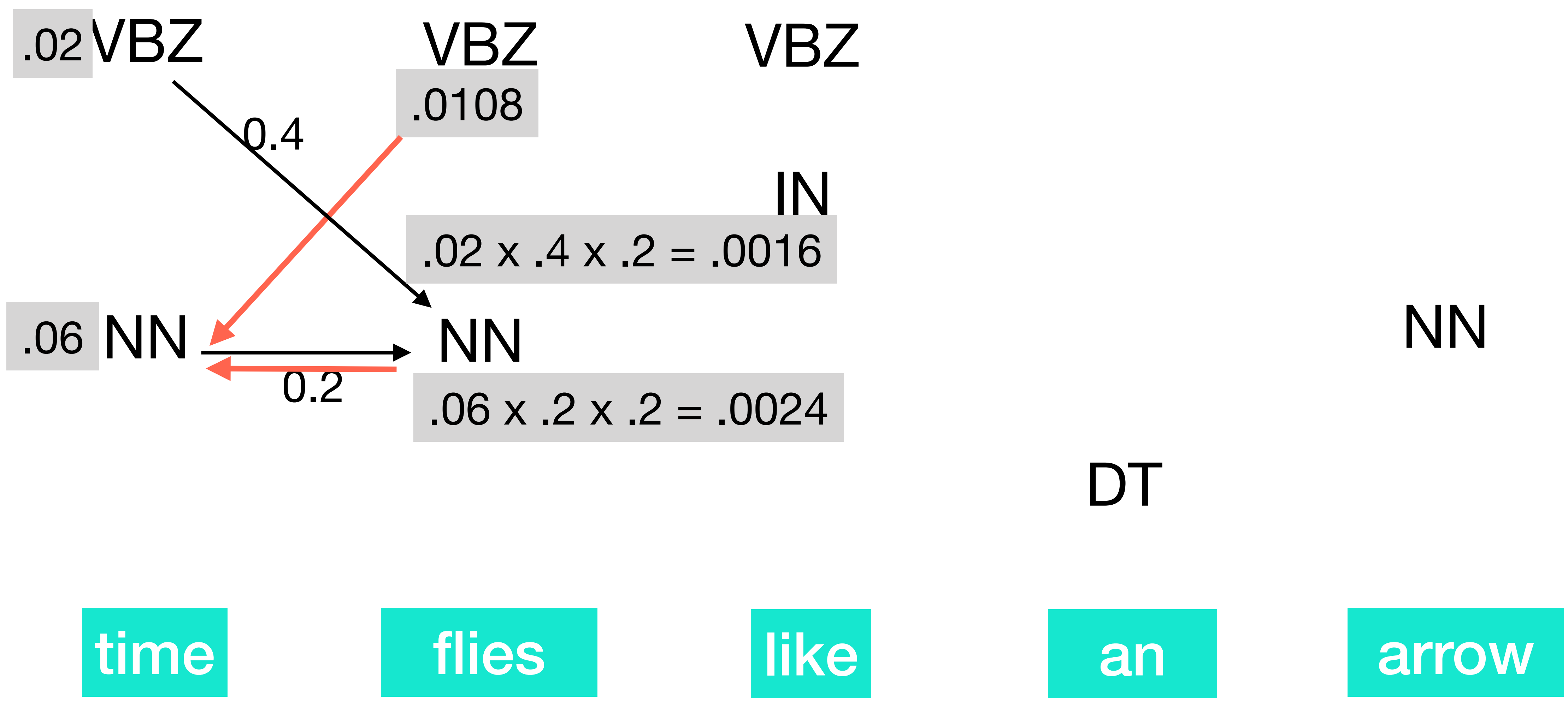
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



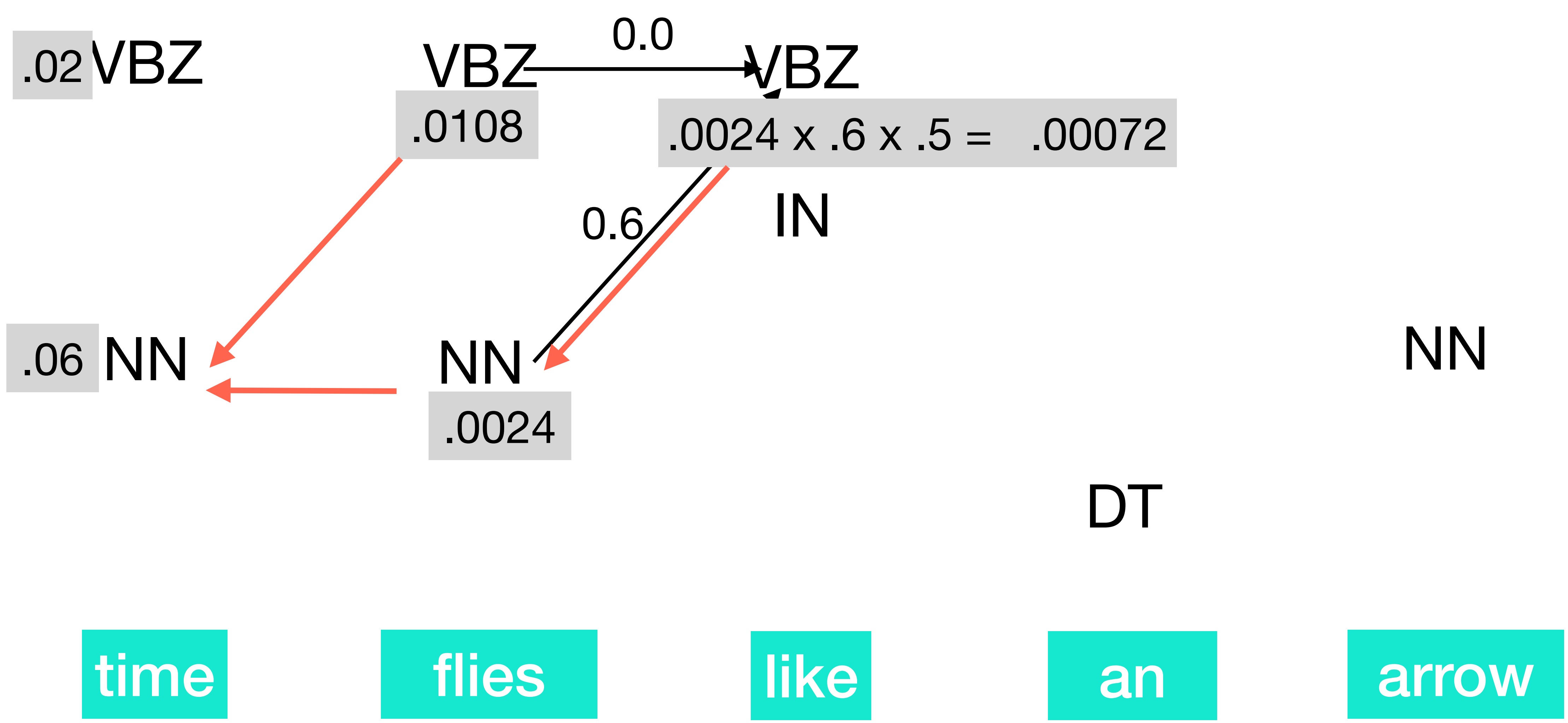
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



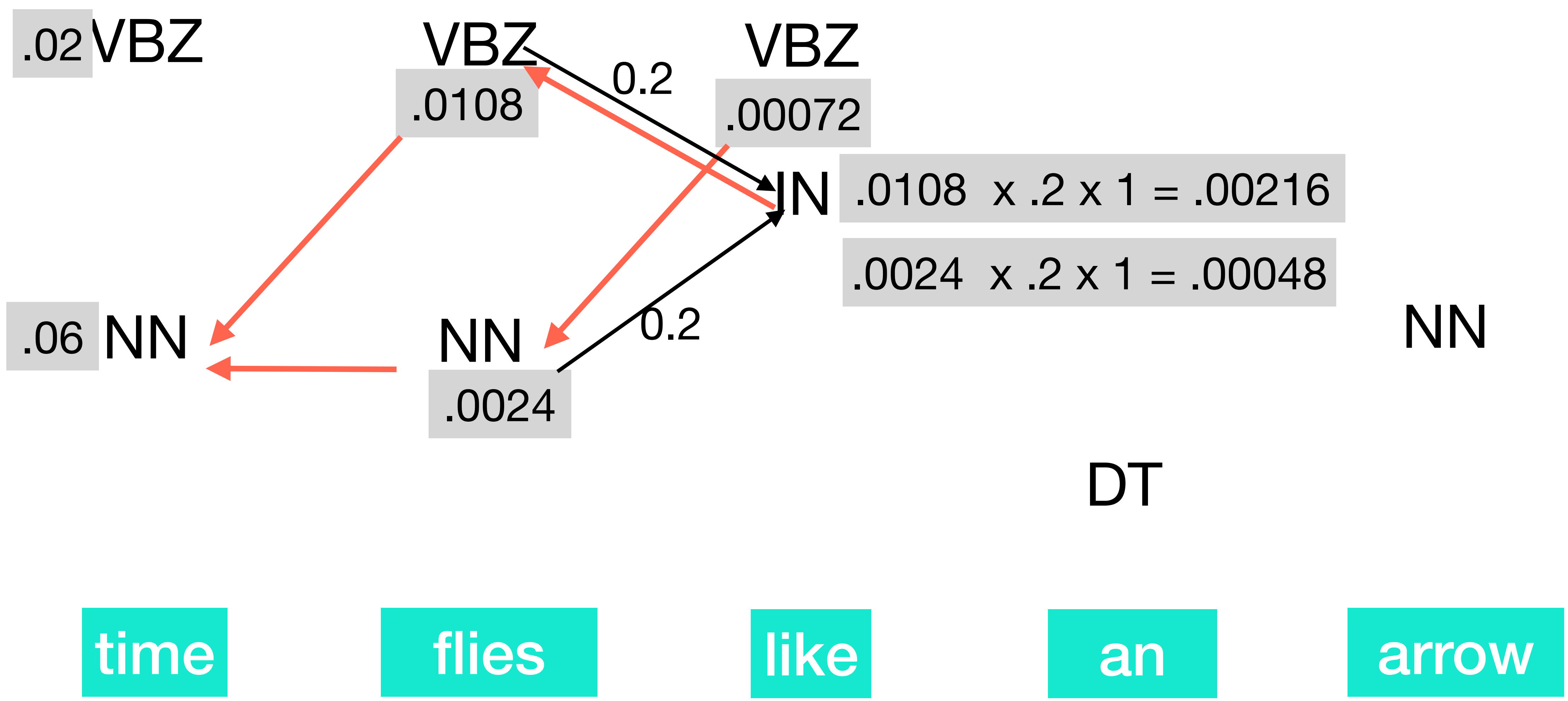
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



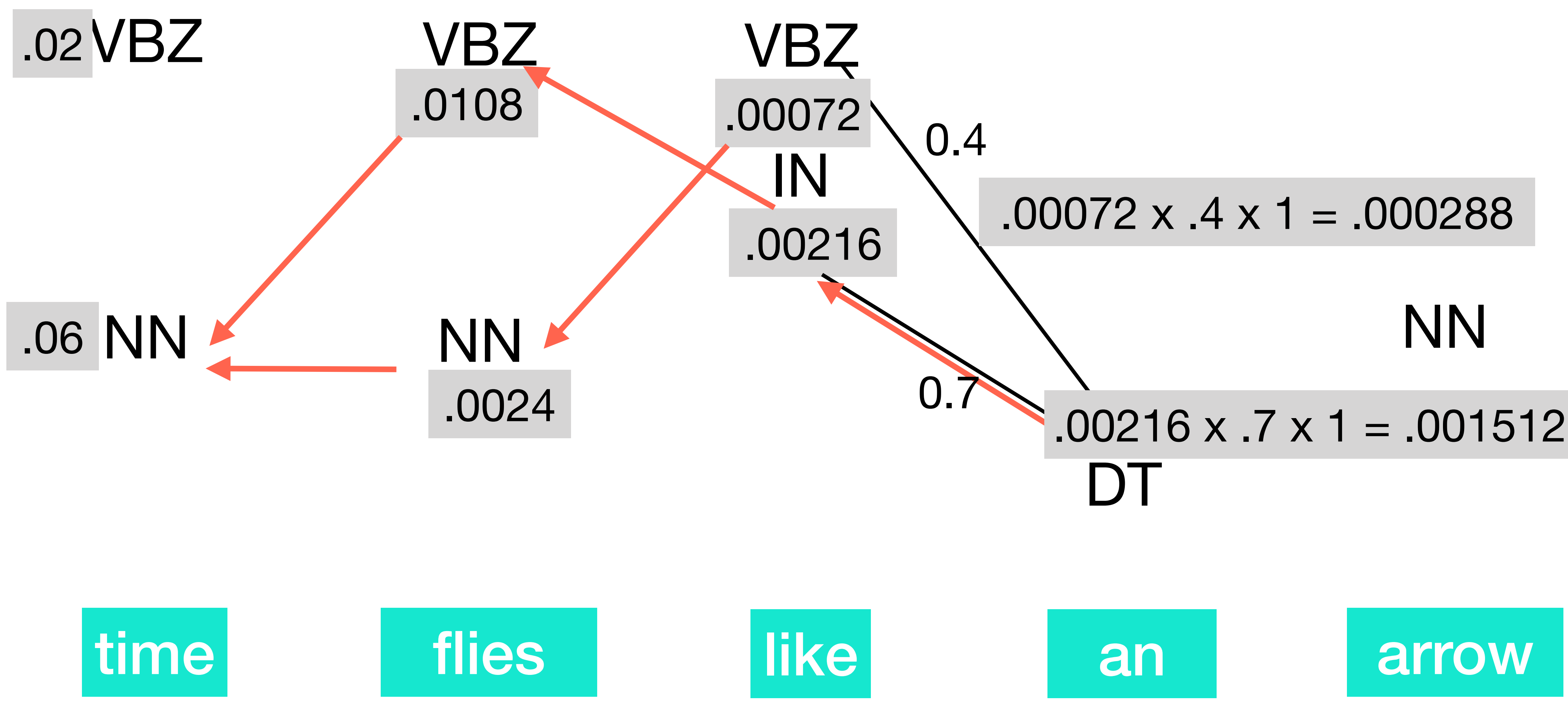
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



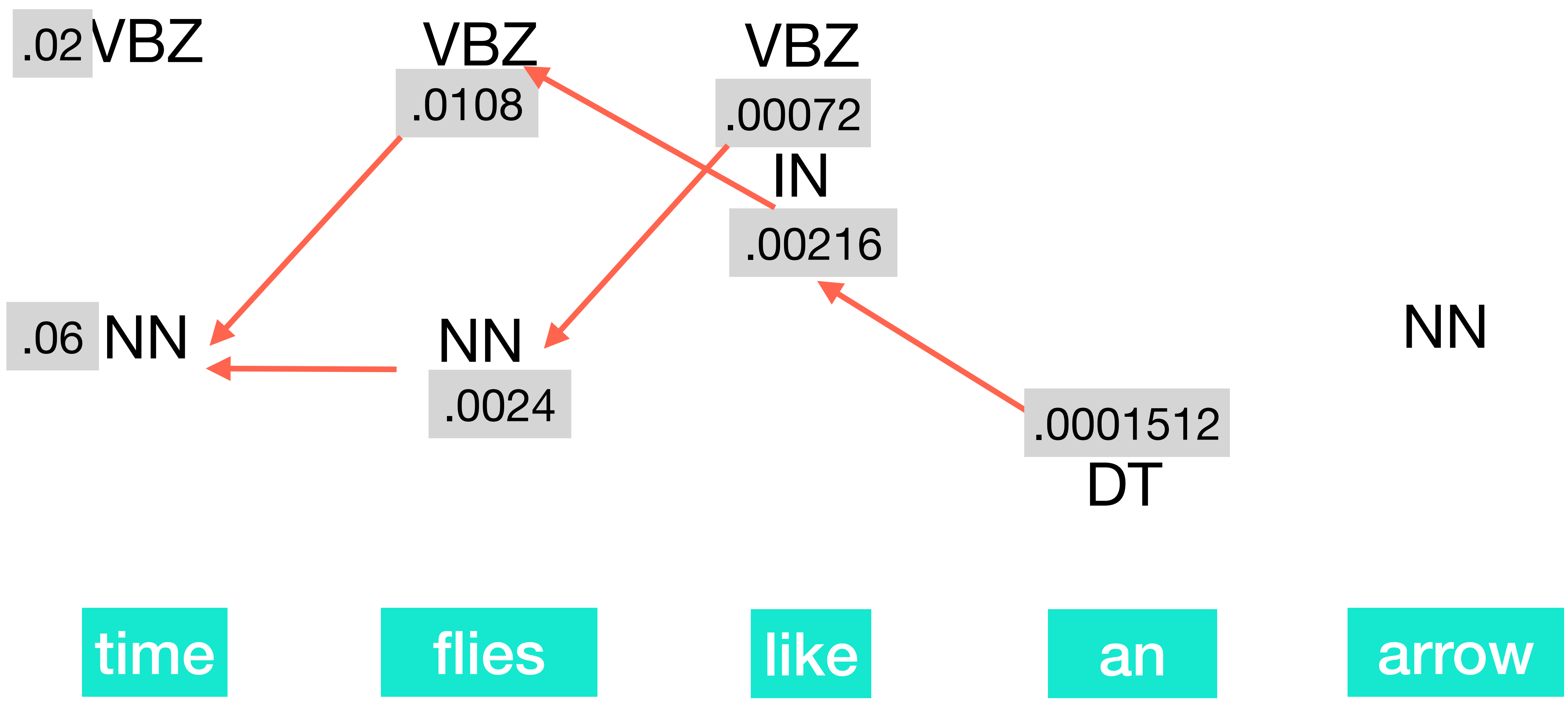
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



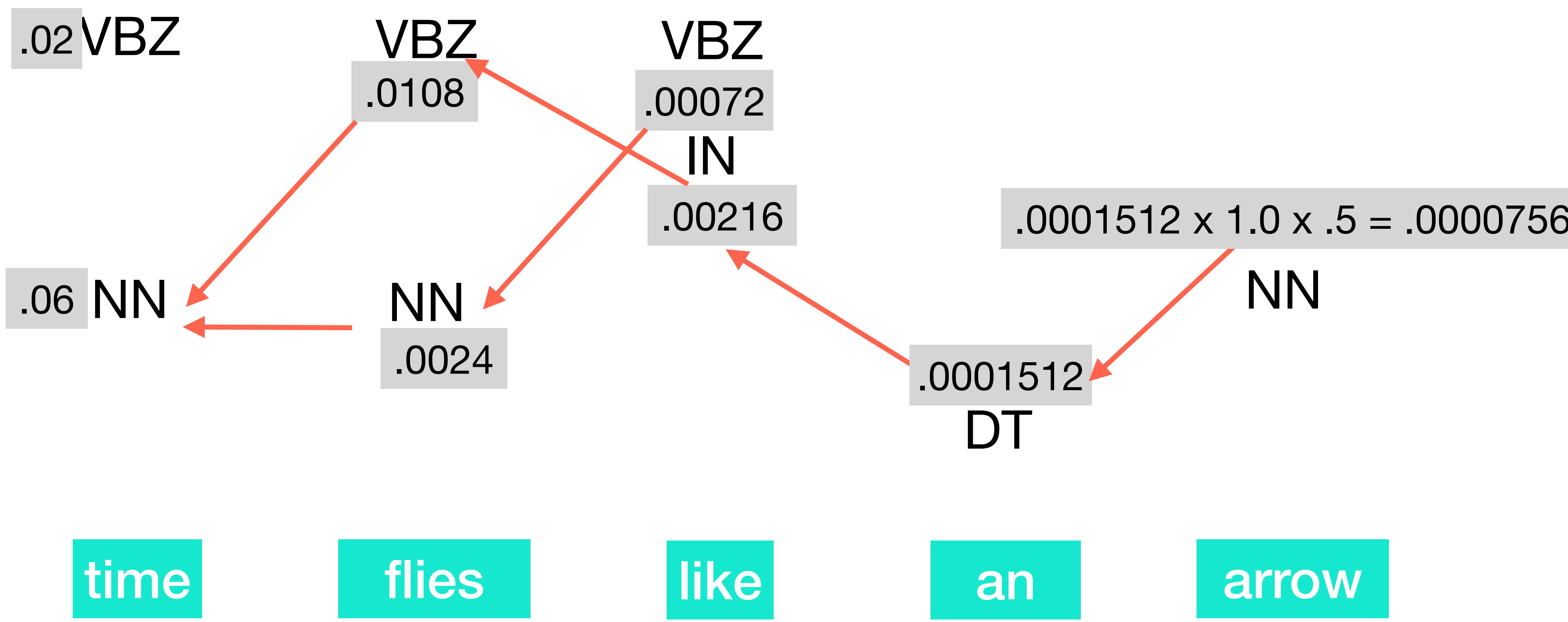
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



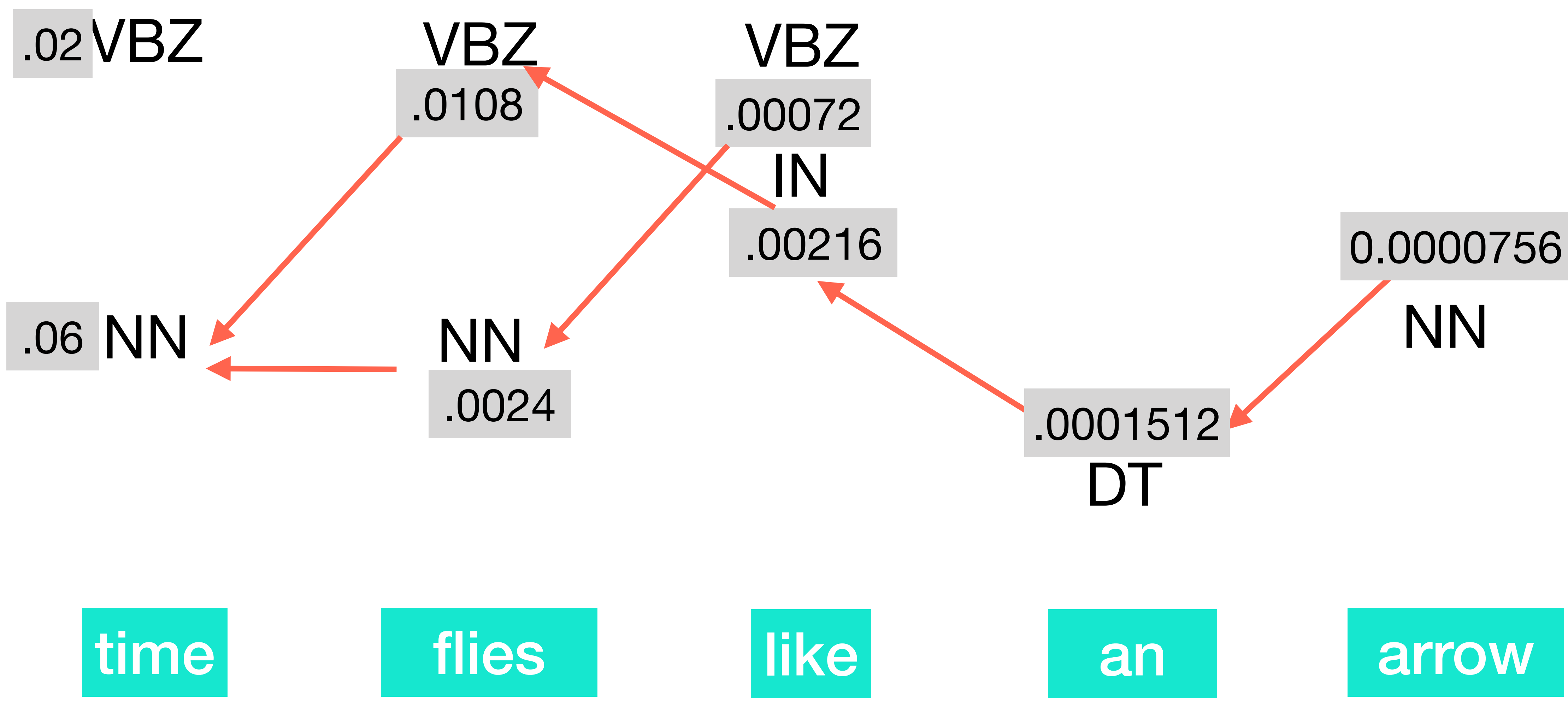
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



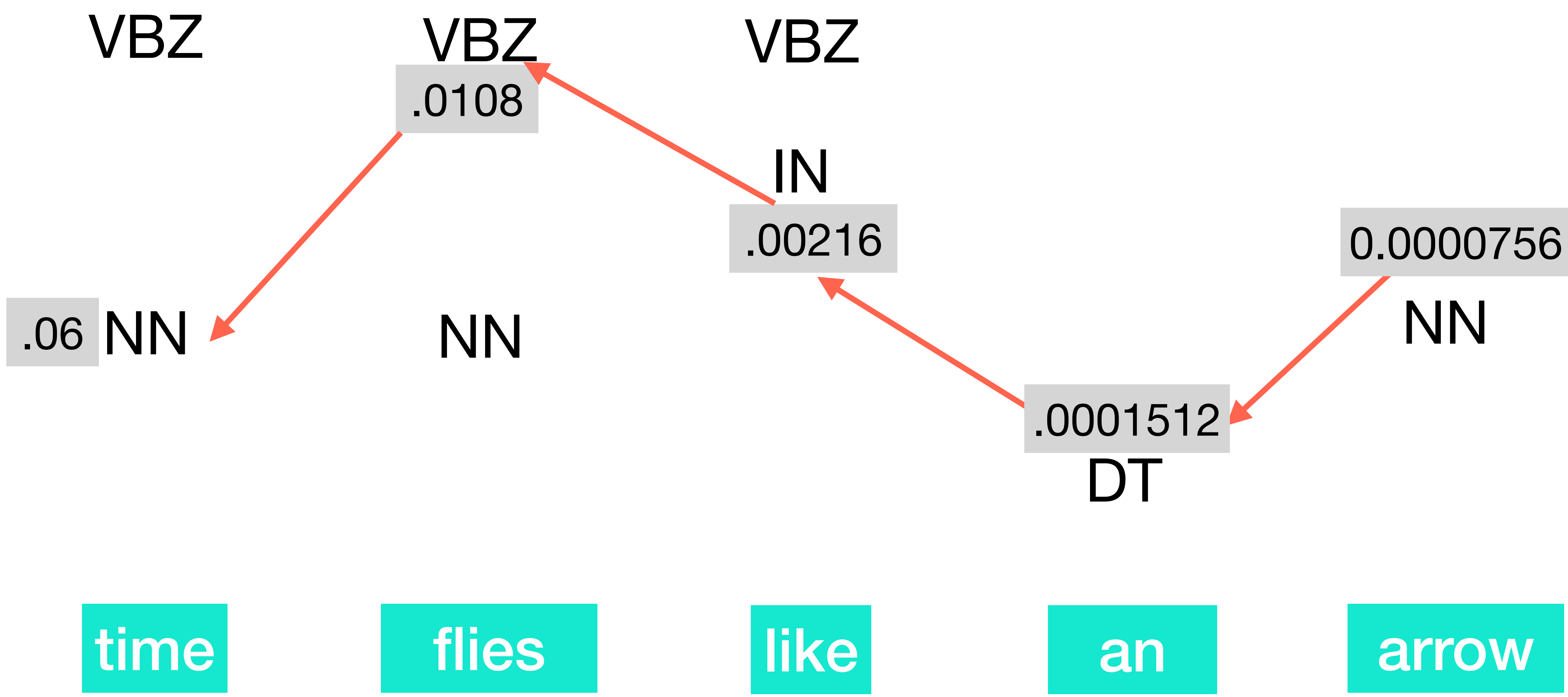
- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm



- Idea: Because of the Markov assumption, we only need the probabilities for X_n to compute the probabilities for X_{n+1}.
  This suggests a **dynamic programming** algorithm.

Viterbi Algorithm

- **Input**: Sequence of observed words $w_1, \ldots, w_n$
- Create a table π, such that each entry $\pi[k,t]$ contains the score of the highest-probability sequence ending in tag t at time k.
- initialize $\pi[0,\text{start}]=1.0$ *and* $\pi[0,t]=0.0$ *for all tags* $t \in T$.
- *for* $k=1$ *to* n*:*
  - *for* $t \in T$*:*
    - $$\pi[k,t] \leftarrow \max_s \pi[k-1,s] \cdot \underbrace{P(t|s)}_{\text{transition probability}} \cdot \overbrace{P(w_k|t)}^{\text{emission probability}}$$
- ***return*** $\max_s \pi[n,s]$

Trigram Language Model

- Instead of using a unigram context $P(t_i|t_{i-1})$, use a bigram context $P(t_i|t_{i-2}t_{i-1})$.
    - Think of this as having states that represent *pairs of tags.*
- So the HMM probability for a given tag and word sequence is:

$$\prod_{i=1}^{n} P(t_i|t_{i-2}t_{i-1})P(w_i|t_i)$$

- Need to handle data sparseness when estimating transition probabilities (for example using backoff or linear interpolation)

HMMs as Language Models

- We can also use an HMM as language models (language generation, MT, …), i.e. **evaluate** $P(w_1, \ldots, w_n)$ for a given sentence.
  What is the advantage over a plain word n-gram model?

- Problem: There are many tag-sequences that could have generated $w_1, \ldots w_n$.

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

- This is an example of **spurious ambiguity.**

- Need to compute: $$P(w_1, \ldots w_n) = \sum_{t_1, \ldots t_n} P(w_1, \ldots, w_n, t_1, \ldots, t_n)$$
$$= \sum_{t_1, \ldots t_n} \left[\prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i) \right]$$

Forward Algorithm

- **Input**: Sequence of observed words $w_1, \ldots, w_n$
- Create a table π, such that each entry $\pi[k,t]$ contains the sum of the probabilities of all tag/word sequences ending in tag t at time k.
- initialize $\pi[0,\text{start}]=1.0$ *and* $\pi[0,t]=0.0$ *for all tags* $t \in T$.
- *for* $k=1$ *to* n*:*
  - *for* $t \in T$*:*
    - $\pi[k,t] \leftarrow \sum_s \pi[k-1,s] \cdot P(t|s) \cdot P(w_k|t)$
- ***return*** $\sum_s \pi[n,s]$

Named Entity Recognition as Sequence Labeling

- Use 3 tags:
    - O - outside of named entity
    - I - inside named entity
    - B - first word (beginning) of named entity

	O	O	B	I	O	
…	identification	of	tetronic	acid	in	…

- Other encodings are possible (for example, NE-type specific)
- This can also be used for other tasks such as phrase chunking and semantic role labeling.