Lecture 4: Sentence level processing

Content:

- Part of Speech (POS) tagging
- Named entity recognition (NER)
- Hidden Markov models (HMM), Viterbi algorithm
- Recurrent neural networks (RNN)

Presented by *Mikko Kurimo*

(Some adapted content from Oskar Kohonen and Teemu Ruokolainen - thanks!)

Why to study this?

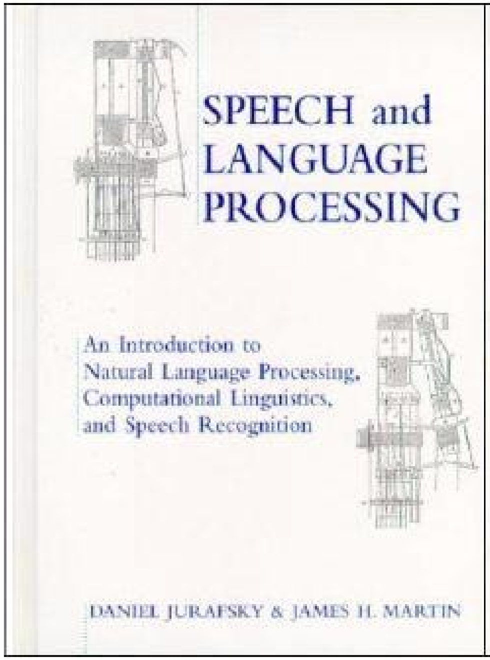
- Make a system that can answer questions!
- How much *understanding* is needed?
- Start by finding out **who did what to whom**
- “The classical NLP stuff”
- Sequence labeling

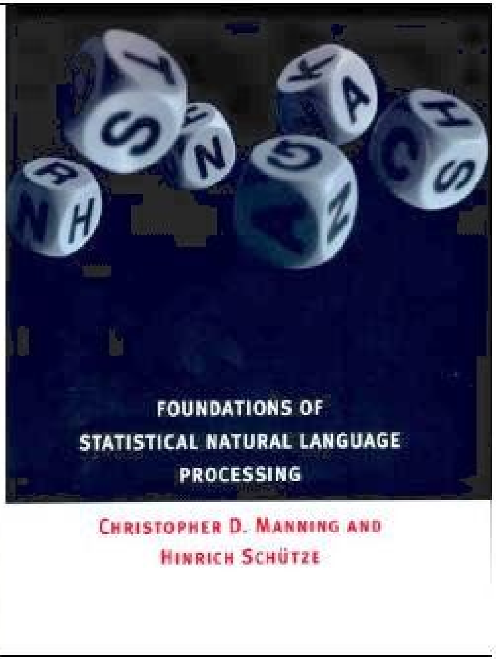
Goals of today

1. You can do sequence labeling by statistical methods
2. Apply hidden Markov models and Viterbi search to Part-of-Speech tagging
3. Learn the basic idea of tagging by neural networks

Reading material

- Manning, C. D. and Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press. (Ch 9-12)
- Jurafsky, D. and Martin, J. H. (2008). Speech and Language Processing. Prentice Hall. 2nd edition. (Chapter 4)
- Jurafsky, D. and Martin, J. H. (2020). Speech and Language Processing. 3nd edition. (Chapters 8, 9)





Lecture schedule 2022

1. 11 jan Introduction & Project groups / Mikko Kurimo
2. 18 jan Statistical language models / Mikko Kurimo
3. 25 jan Word2vec / Tiina Lindh-Knuutila
4. **01 feb Sentence level processing / Mikko Kurimo**
5. 08 feb Speech recognition / Janne Pylkkönen
6. 15 feb Morpheme-level processing / Mathias Creutz
7. 22 feb Exam week, no lecture
8. 01 mar Statistical machine translation / Jaakko Väyrynen
9. 08 mar Neural language modeling and BERT / Mittul Singh
10. 15 mar Neural machine translation / Stig-Arne Grönroos
11. 22 mar Chatbots and dialogue agents / Mikko Kurimo
12. 29 mar Societal impacts and course conclusion / Krista Lagus, Mikko

See Mycourses for updates

Part of Speech tagging

Task: Assign tags *y(t)* to each word *x(t)* in a sentence

Words: *x1 x2 x3 … xN*

=> Tags: *y1 y2 y3 … yN*

Words: The reaction in the newsroom was emotional.

=> Tags: *DT NN IN DT NN VB JJ*

DT = determiner

NN = noun

IN = preposition

VB = verb

JJ = adjective

Part of Speech (POS) tagging

Task: Assign tags for each word in a sentence

Applications: Tools for parsing the sentence

The reaction in the newsroom was emotional.

=> *DT NN IN DT NN VB JJ*

DT = determiner

NN = noun

IN = preposition

VB = verb

JJ = adjective

Named entity recognition (NER)

- Detect names of persons, organizations, locations
- Detect dates, addresses, phone numbers, etc
- Applications: Information retrieval, ontologies

UN official Ekeus heads for Baghdad.

=> ORG - PER - - LOC

(organization) (person) (location)

Discussion

- How would you start building a part-of-speech tagger?
- Or a named entity recognizer for news articles?

- Is it possible without any *understanding* by just counting statistics?
- If not, what is the problem?

A general approach

1. Generate tagging candidates
2. Score the candidates
3. Select the highest scoring ones

Example: count POS tags

Possible tags	Open	a	tuna	can	.
1.	VB	DT	NN	MD	
2.	JJ	NN		NN	
3.					
...					

Most words have several possible tags

DT = determiner

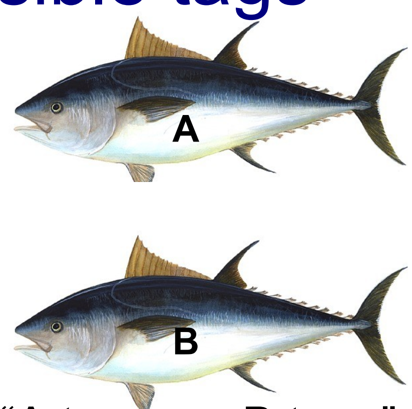
NN = noun

MD = modal verb

VB = verb

JJ = adjective

"open A"



"A tuna vs. B tuna"

"tuna can"

A simple scoring method

1. Find all appearances of the word in **an annotated corpus**
2. Count the frequency of each tag for that word
3. Select the most common tag for each word

Language resources POS

Annotated text corpora

- English: Penn Treebank (1993)
- Finnish: Turku Dependency Treebank (2014) http://bionlp.utu.fi/fintreebank.html

POS taggers

- English: Stanford POS tagger (around 2000) http://nlp.stanford.edu/software/tagger.shtml
- Finnish: FinnPos (2015)https://github.com/mpsilfve/FinnPos/
  - Helsinki + Aalto Univ. (Ruokolainen PhD, 2016)
  - CRF + Sub-label dependencies

Language resources NER

Corpora with named entity annotations

- English: MUC-6 (2003), CoNLL (2003)
- Finnish: FiNER (2018), TurkuNER (2020)

Named entity recognizers

- English: Stanford Named Entity Recognizer (2006)
  http://nlp.stanford.edu/software/CRF-NER.shtml
- Finnish: FiNER (U.Helsinki), TurkuNER (U.Turku)

  https://github.com/Traubert/FiNer-rules/blob/master/finer-readme.md

  https://github.com/TurkuNLP/turku-ner-corpus
- Spoken NER (Porjazovski MSc, Aalto 2020)
  https://memad.eu/2020/12/21/end-to-end_named_entity_recognition_spoken_finnish/

Example: Using Penn Treebank tag counts

Open	a	tuna	can
1. VB 46	DT 18446	NN 3	MD 893
2. JJ 85	NN 2		NN 3

DT = determiner

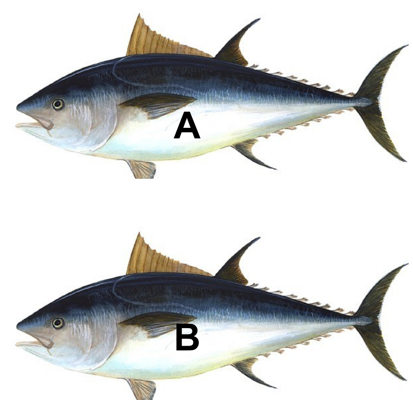
NN = noun

MD = modal verb

VB = verb

JJ = adjective

“open A”



“A tuna vs. B tuna”

“tuna can”

Using Penn Treebank tag counts

Open	a	tuna	can
1. **VB** 46	**DT** 18446	**NN** 3	MD 893
2. JJ 85	NN 2		**NN** 3

Proposed answer are the tags with highest counts

- vs. the correct answer bolded

This simple approach gives about 90% accuracy

Discussion: Not very good, how to do better? Any other information that could be used?

Using Penn Treebank tag counts

	Open	a	tuna	can
1.	VB 46	DT 18446	NN 3	MD 893
2.	JJ 85	NN 2		NN 3

Any other information that could be used?

Hint: Why did this example fail? JJ-DT pairs are rare, but JJ-NN and VB-DT are common

Count transitions

- Use the Penn Treebank corpus and count how often each **tag pair** appears
- Prepare a **tag transition matrix**
- Compute transition probabilities from the counts
  - Just like bigrams for words, but now for tags
  - P(y1), P(y2|y1), P(y3|y2), P(y4|y3)

Score the tags for the sentence

- Combine the transition probabilities:

  $P(y1)\ P(y2|y1)\ P(y3|y2)\ \ldots$

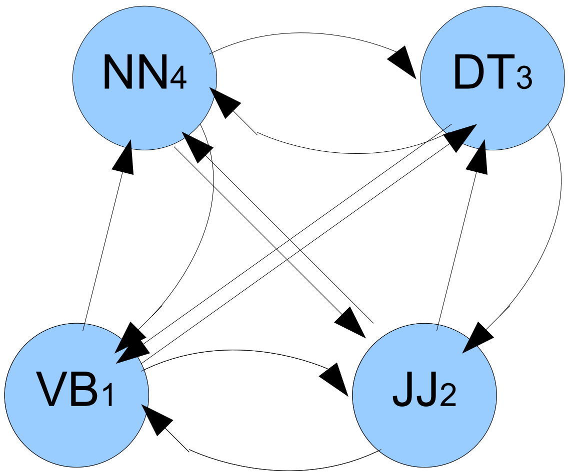
  with the tag-word pair observation probabilites:

  $P(x1|y1)\ P(x2|y2)\ P(x3|y3)$

  to get the total tagging score:

  $P(y1)P(x1|y1)\ P(y2|y1)P(x2|y2)\ P(y3|y2)P(x3|y3)$
- Known as Hidden Markov Model (HMM) tagger
- Achieves about 96% accuracy

Markov chains



A sequence of random variables called as "**states**"

The states can be words, phonemes, POS tags etc.

The transitions between states depend only on the current state

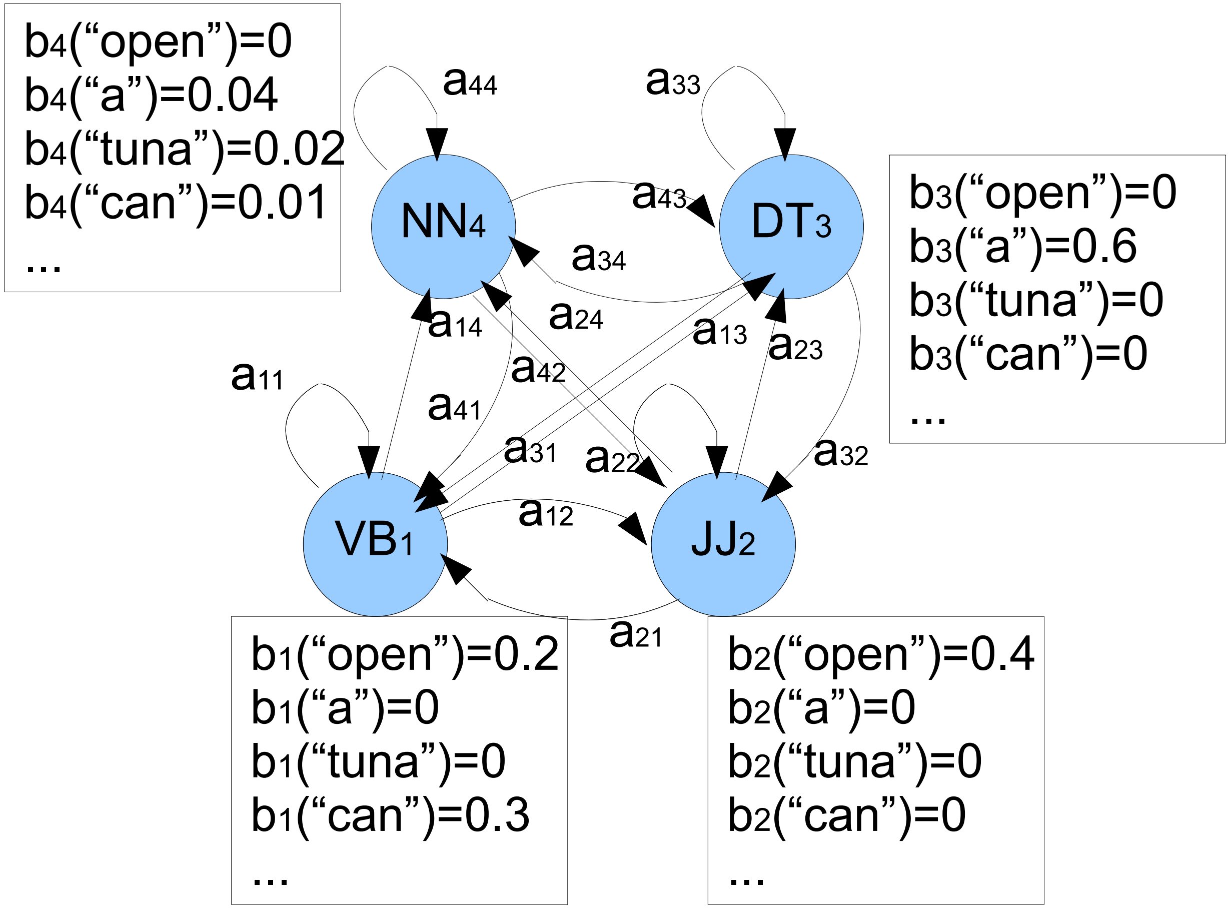
- No history, no time
- The probability of any sequence can be computed easily

Hidden Markov Model (HMM)

Markov chain where the states are hidden and only some features can be observed

Features can be words, speech sounds etc.

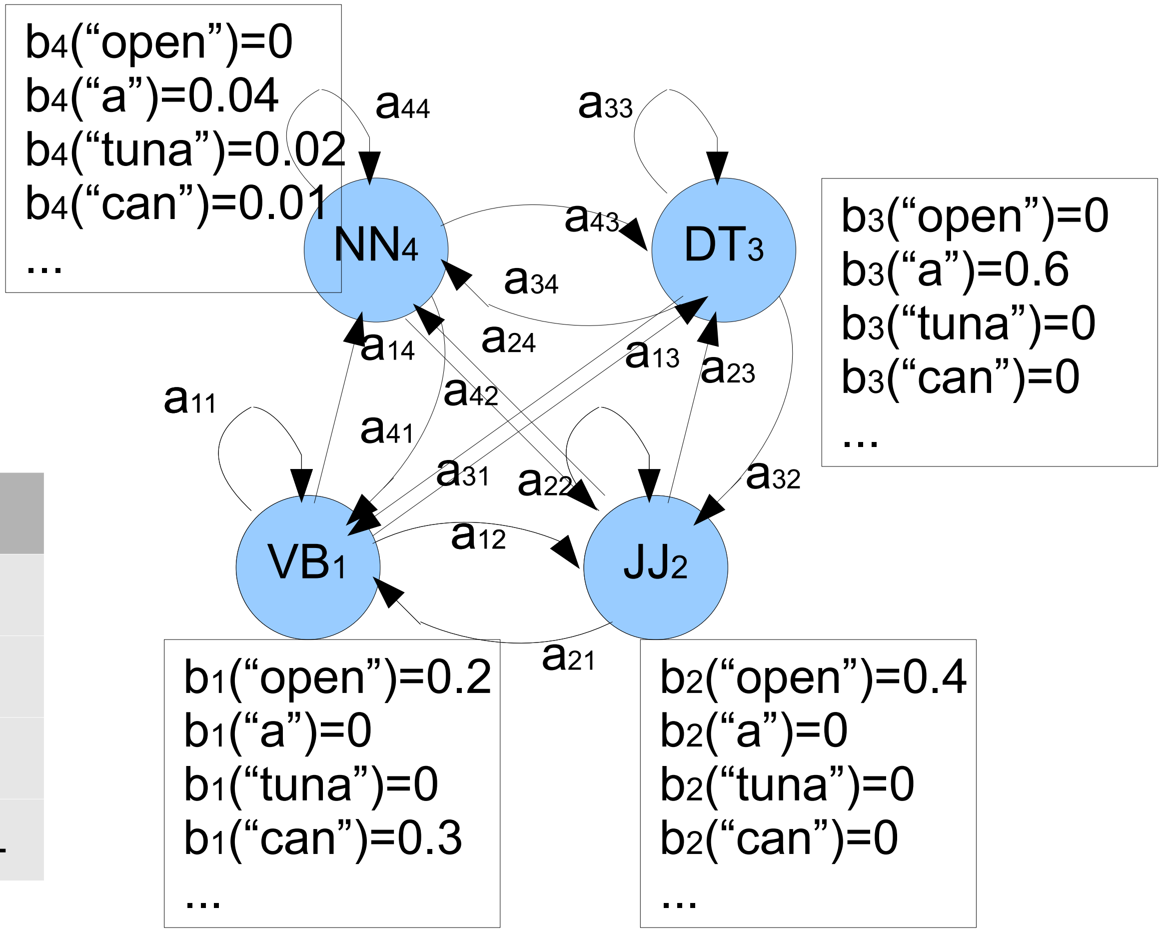
Defined by sets of **transition prob.** a_{ij} and **observation prob.** $b_i(feature)$ for each state i



HMM parameters

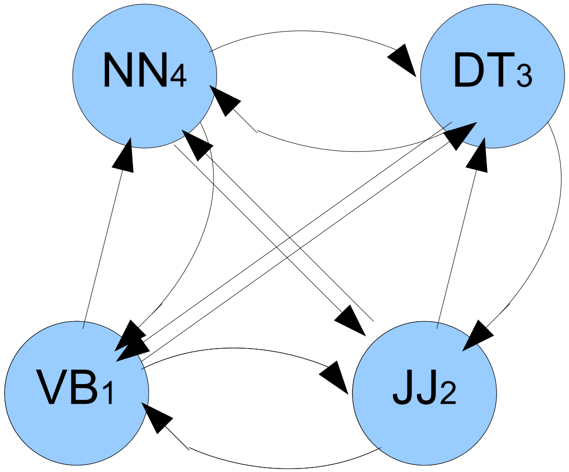
a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01



Note: In matrix *aij* rows sum to one, but in *bi* only four words are shown here.

HMM tagger

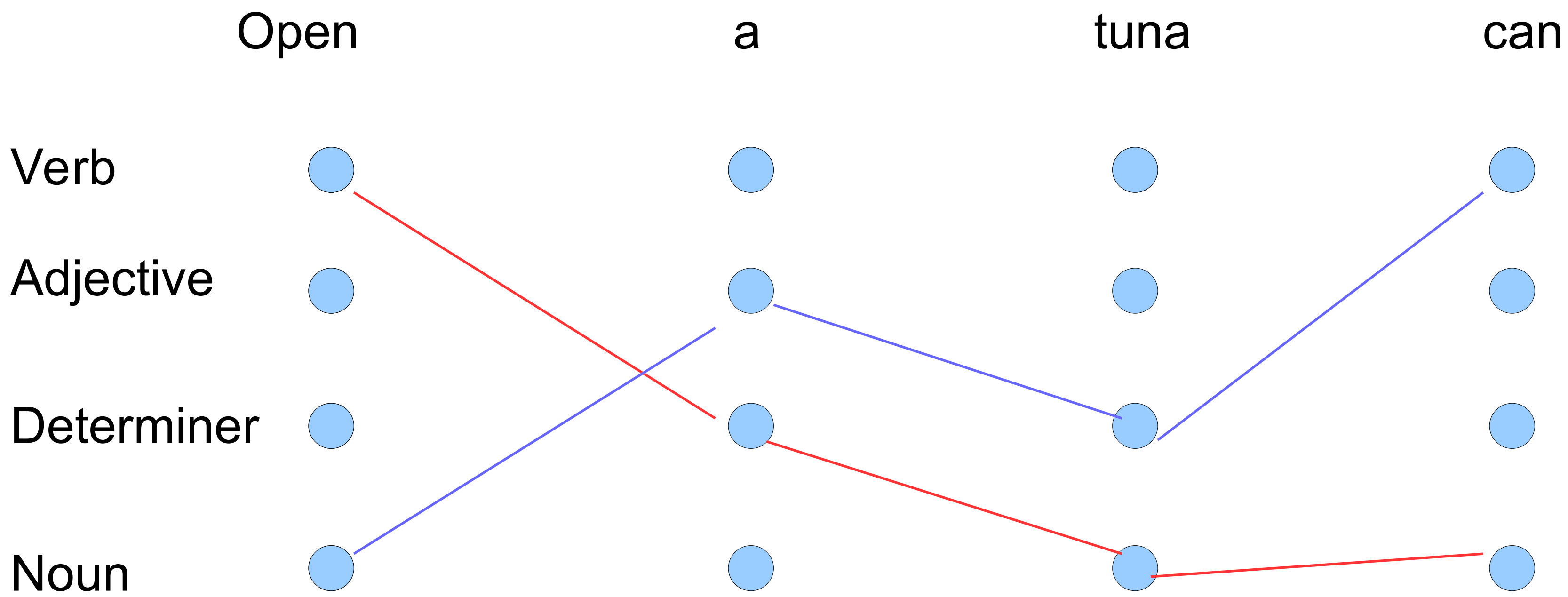


- Markov chain assumes that the next tag depends only on the previous tag
- In HMM the tags are hidden, we only see the words
- Viterbi search returns the most likely tag sequence for given word sequence

Solving the HMM

Must evaluate (*tag_num* ** *sequence_len*) candidate sequences

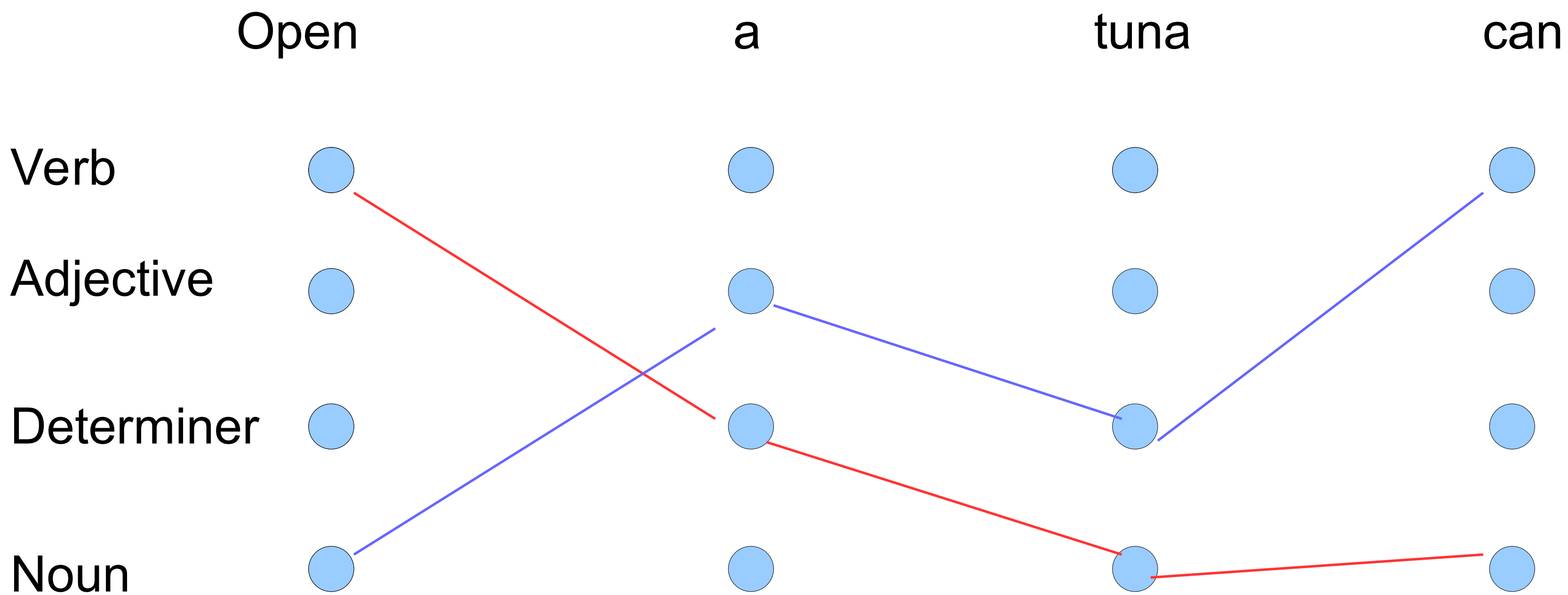
Can be slow. But there is a faster way...



Solving the HMM

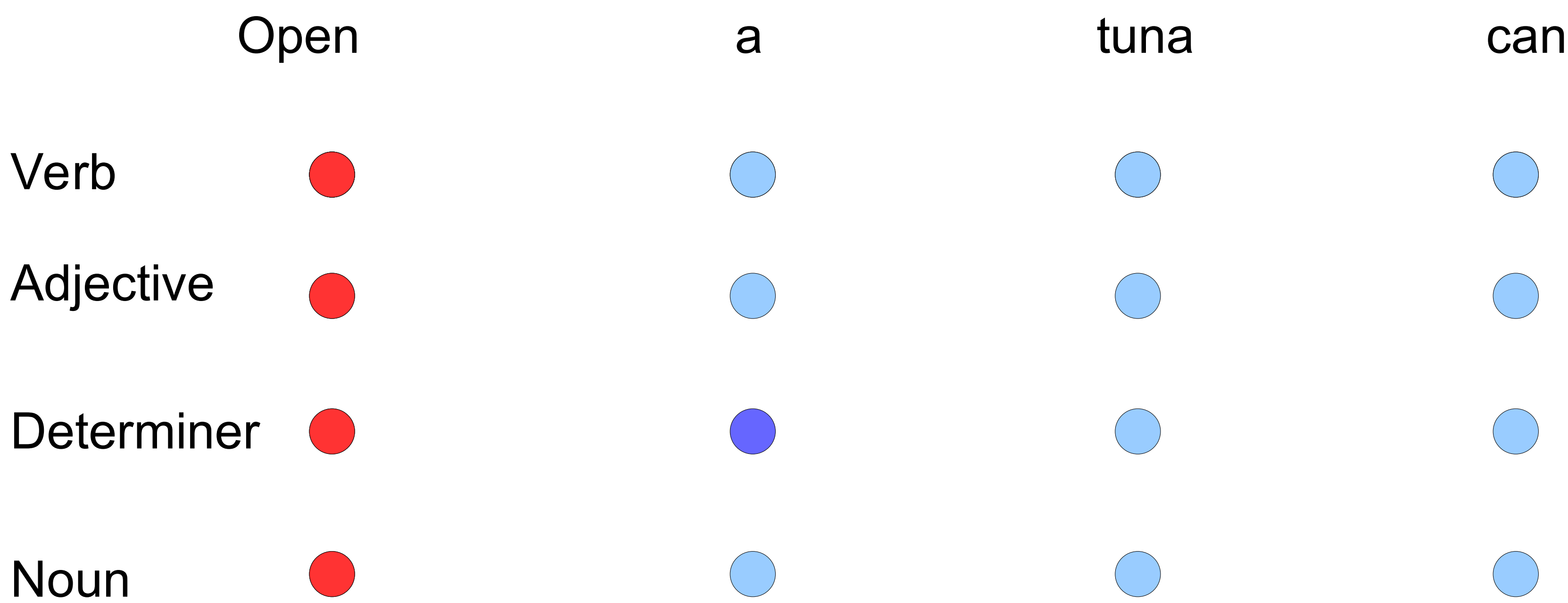
Must evaluate (*tag_num ** sequence_len*) candidate sequences

Can be slow. But there is a faster way...



Viterbi algorithm

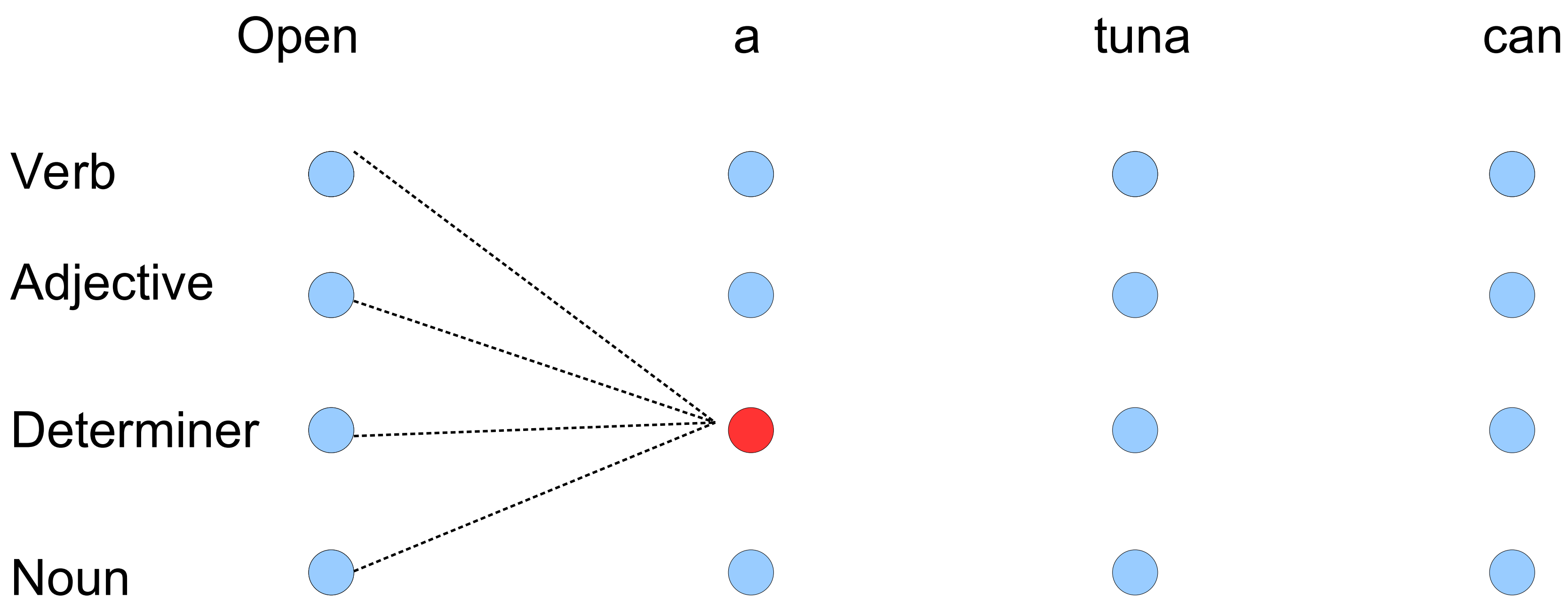
1. compute tag observation probabilities **P(“open”|y1)**



Viterbi algorithm

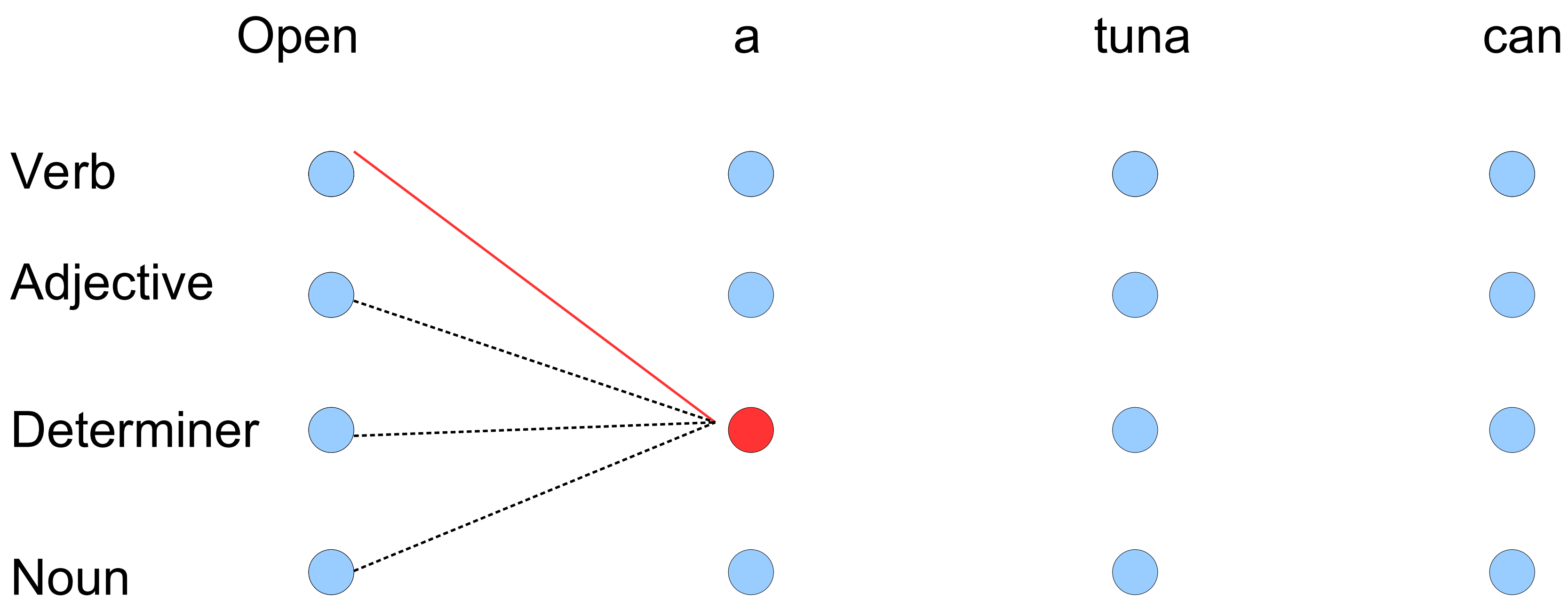
2. What is the best path to each tag at time step 2?

- multiply each path by tag observation **P("a"|y2)** and transition **P(y2|y1)**



Viterbi algorithm

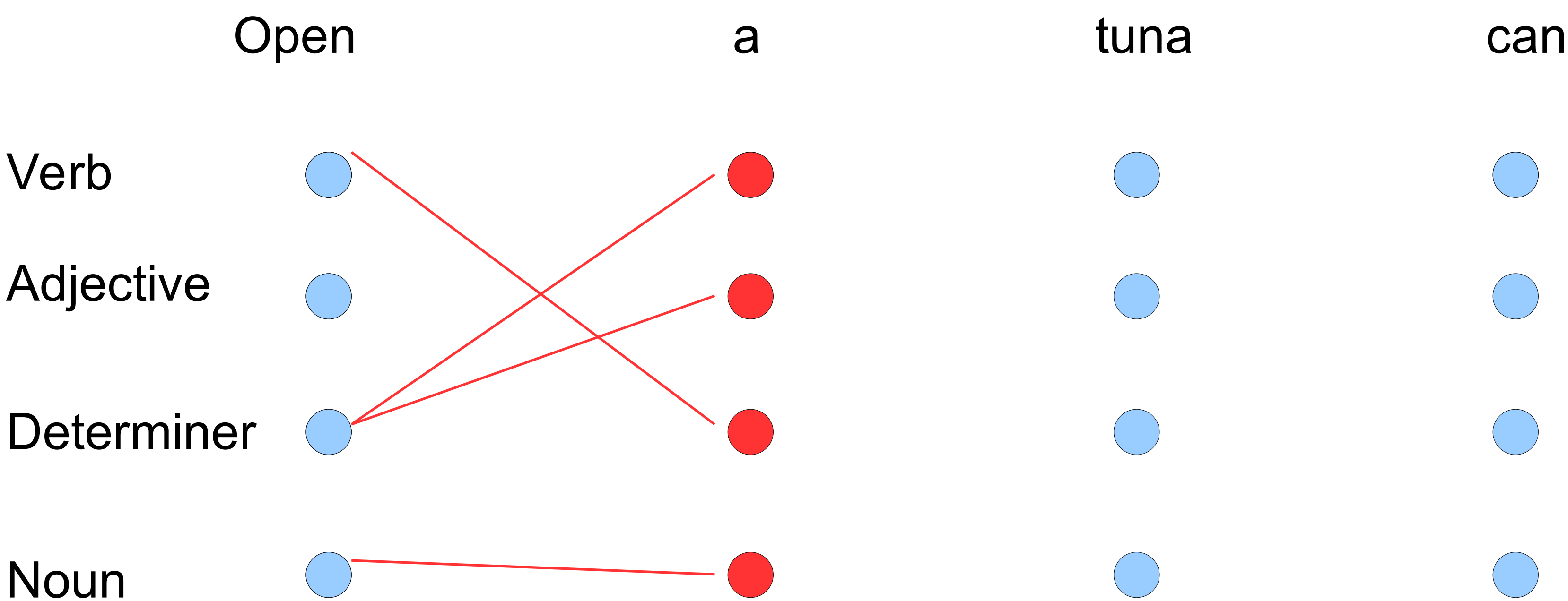
2. What is the best path to each tag at time step 2?
   - select the best path and its probability



Viterbi algorithm

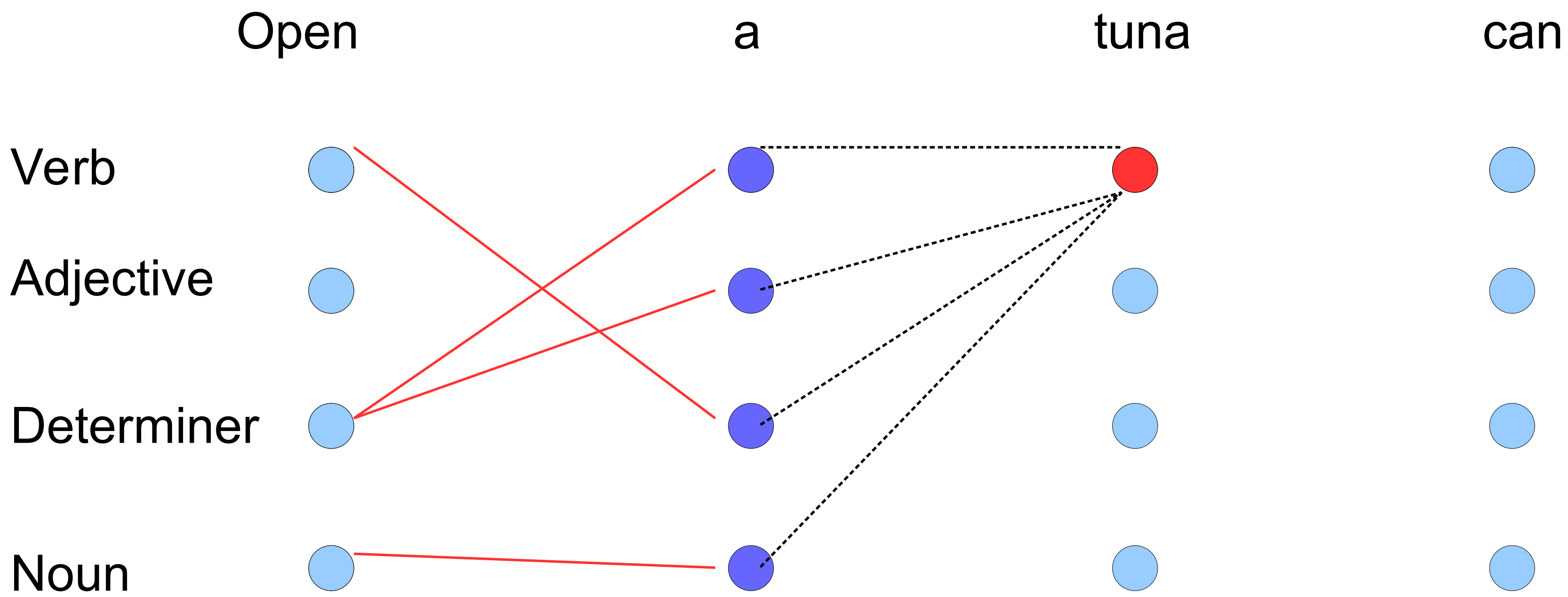
2. Store similarly the best path to each tag at time step 2

Note: The sketch below is not mathematically correct – it is just to explain the idea!



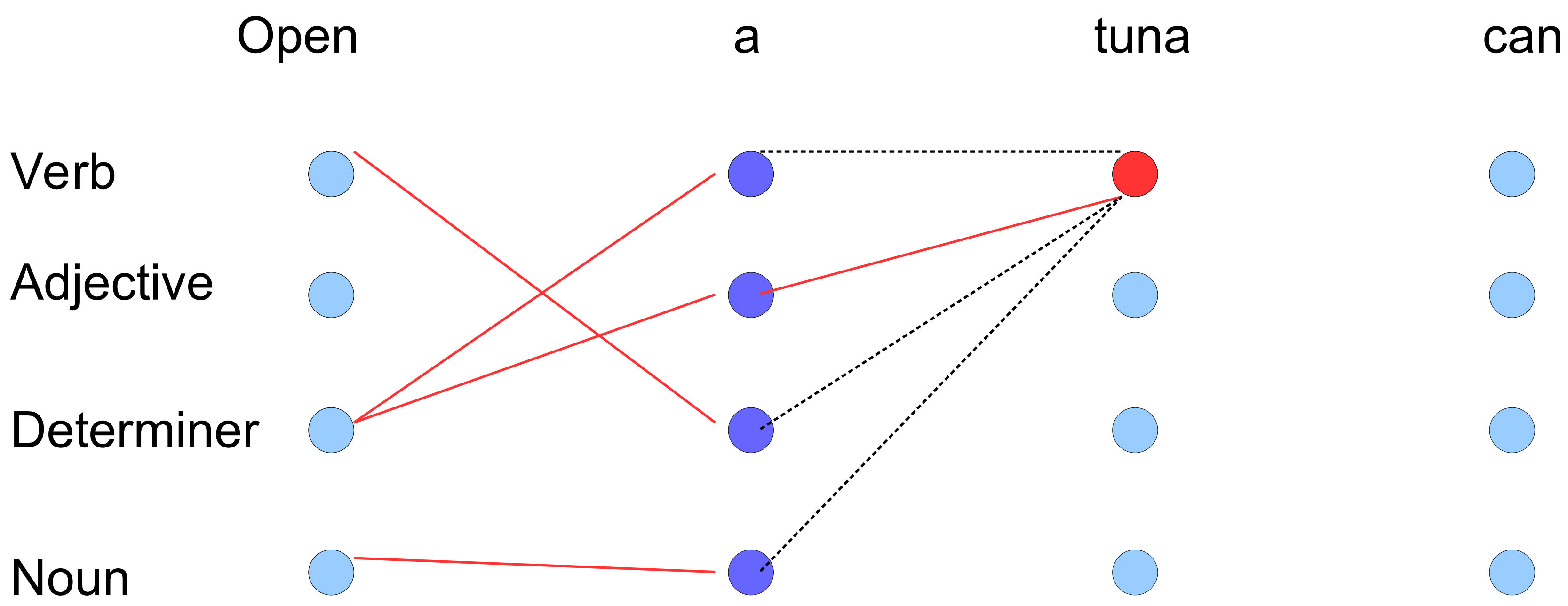
Viterbi algorithm

3. Find the best path to each tag at time step 3, continuing on the previous paths
 - multiply path by tag observation **P("tuna"|y3)** and transition **P(y3|y2)**



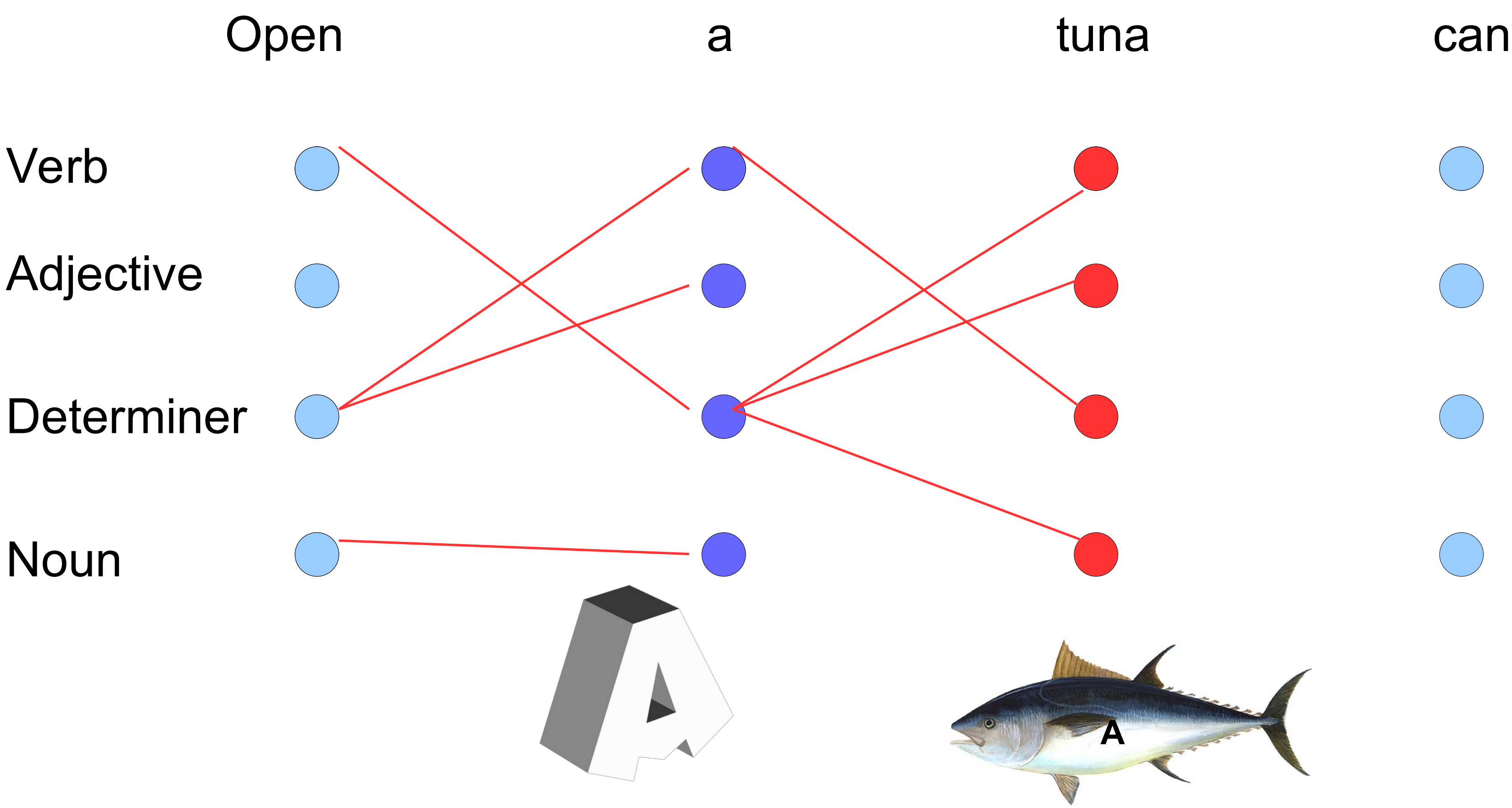
Viterbi algorithm

3. Find the best path to each tag at time step 3, continuing on the previous paths
 - select the best path



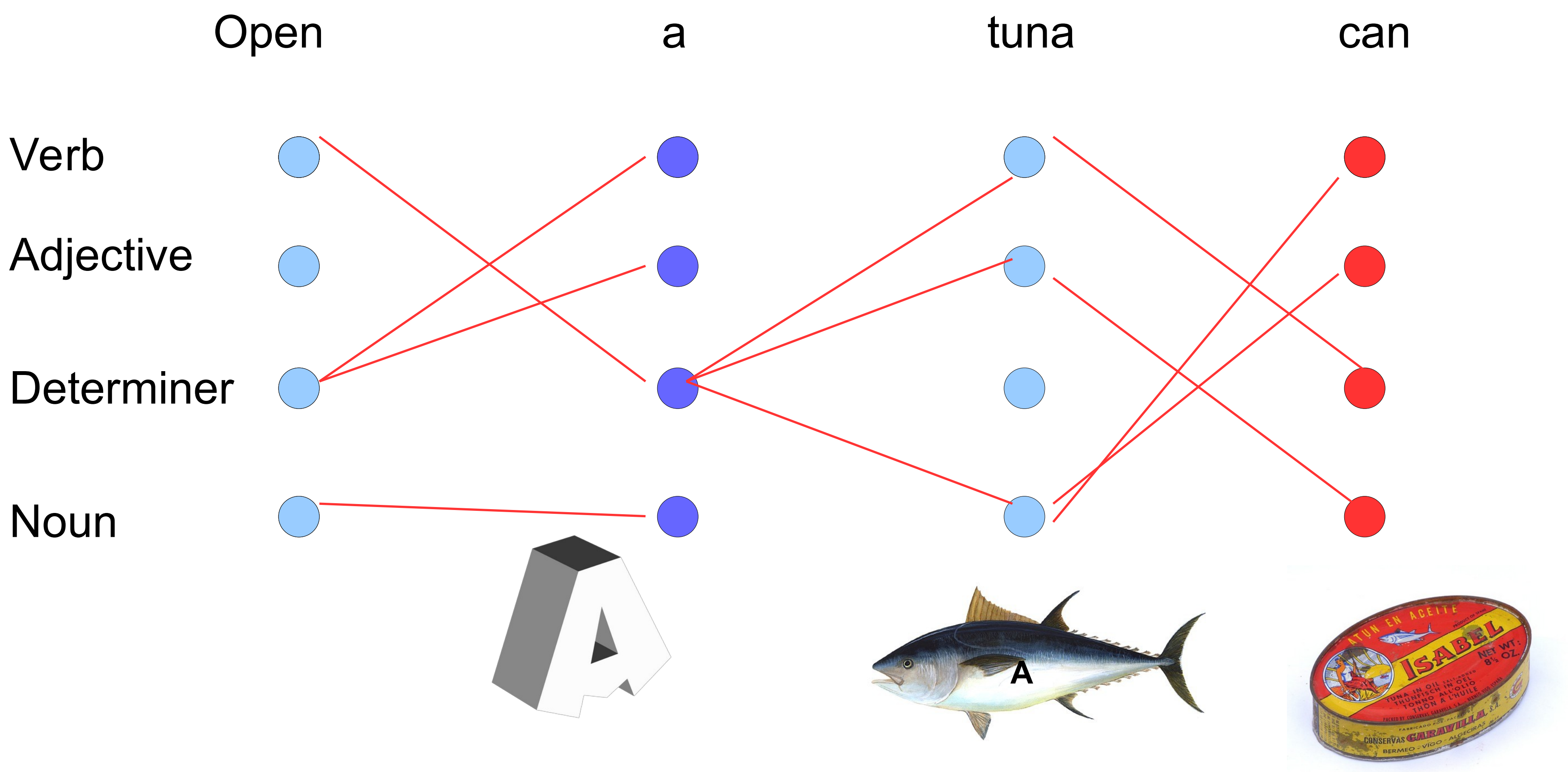
Viterbi algorithm

3. Find the best path to each tag at time step 3, continuing on the previous paths
 - select the best path **for each tag**



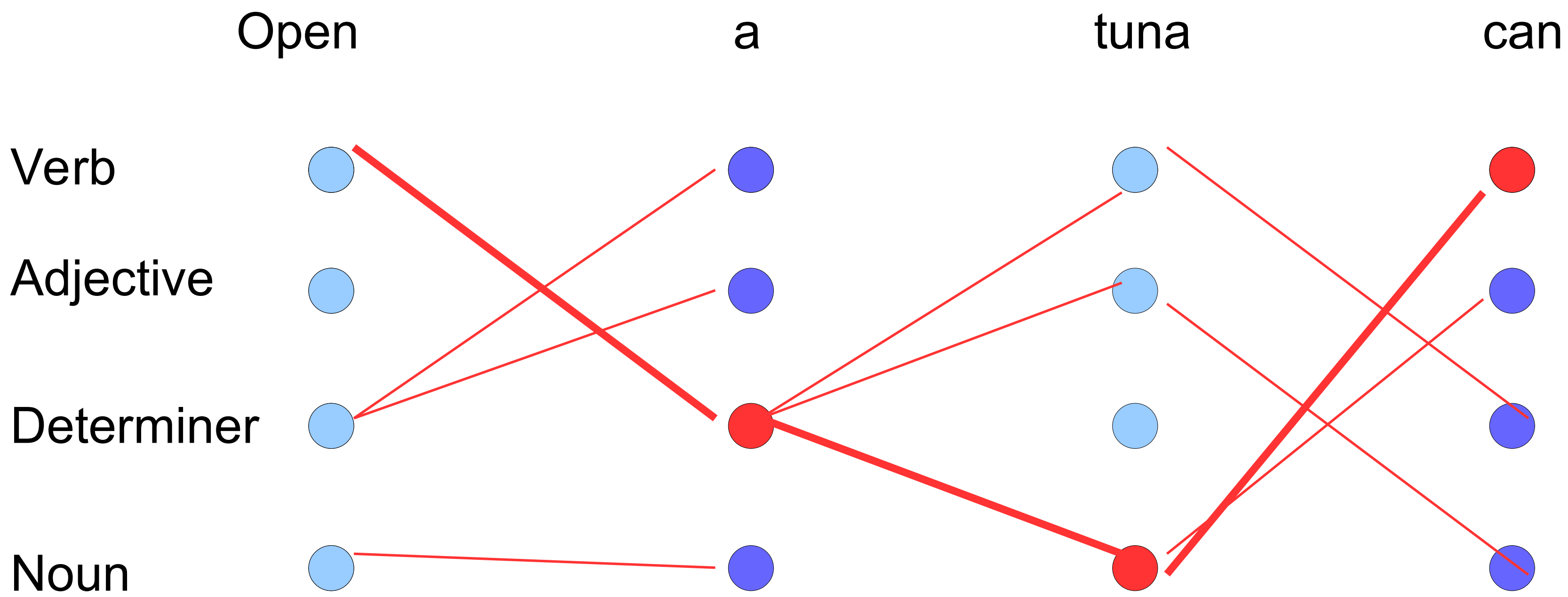
Viterbi algorithm

4. Find the best path to each tag at **time step 4**, continuing on the previous paths
 - select the best path for each tag



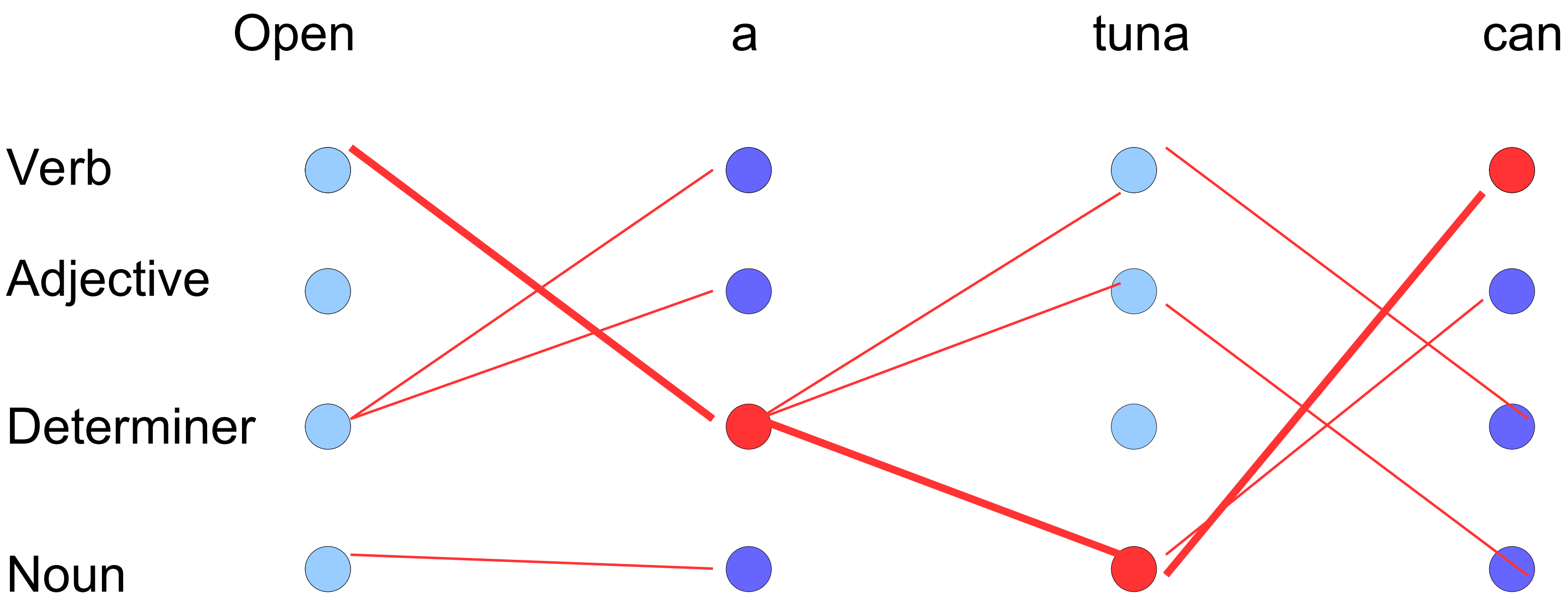
Viterbi algorithm

5. Select the best path overall
- this can still go wrong. **Why?**

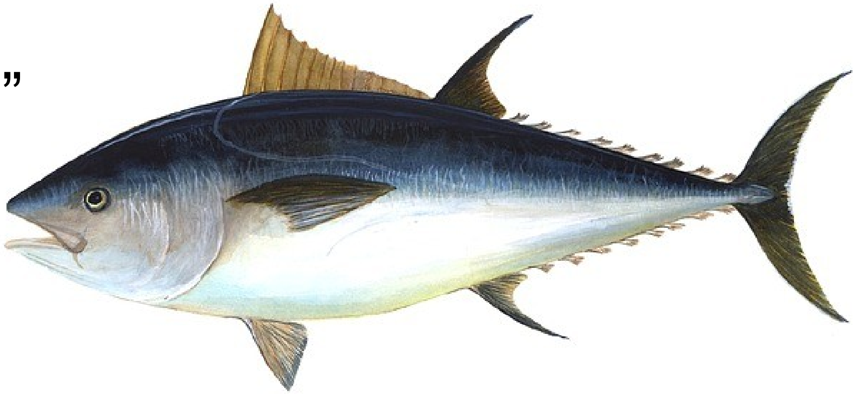


Viterbi algorithm

The local context is not enough!



“tuna **can** change your life”



Viterbi in Matlab HMM toolbox

```
t=1;
delta(:,t) = prior .* obslik(:,t);
psi(:,t) = 0; % arbitrary value, since there is no predecessor to t=1
for t=2:T
  for j=1:Q
    [delta(j,t), psi(j,t)] = max(delta(:,t-1) .* transmat(:,j));
    delta(j,t) = delta(j,t) * obslik(j,t);
  end
end
[p, path(T)] = max(delta(:,T));
for t=T-1:-1:1
  path(t) = psi(path(t+1),t+1);
end
```

Exercise 4: HMM and Viterbi

- Go in breakout rooms, discuss with each other and propose answers for these 3 questions in **MyCourses > Lectures > Lecture 4 exercise return box**:

1. Finish POS tagging by Viterbi search example by hand.
   - Return the values of the boxes and the final tag sequence. Either take a photo of your drawing, fill in the given ppt, or just type the values into the text box
2. Did everyone get the same tags? Is the result correct? Why / why not?
3. What are the pros and cons of HMM tagger?

All submissions, even incorrect or incomplete ones, will be awarded by one activity point.

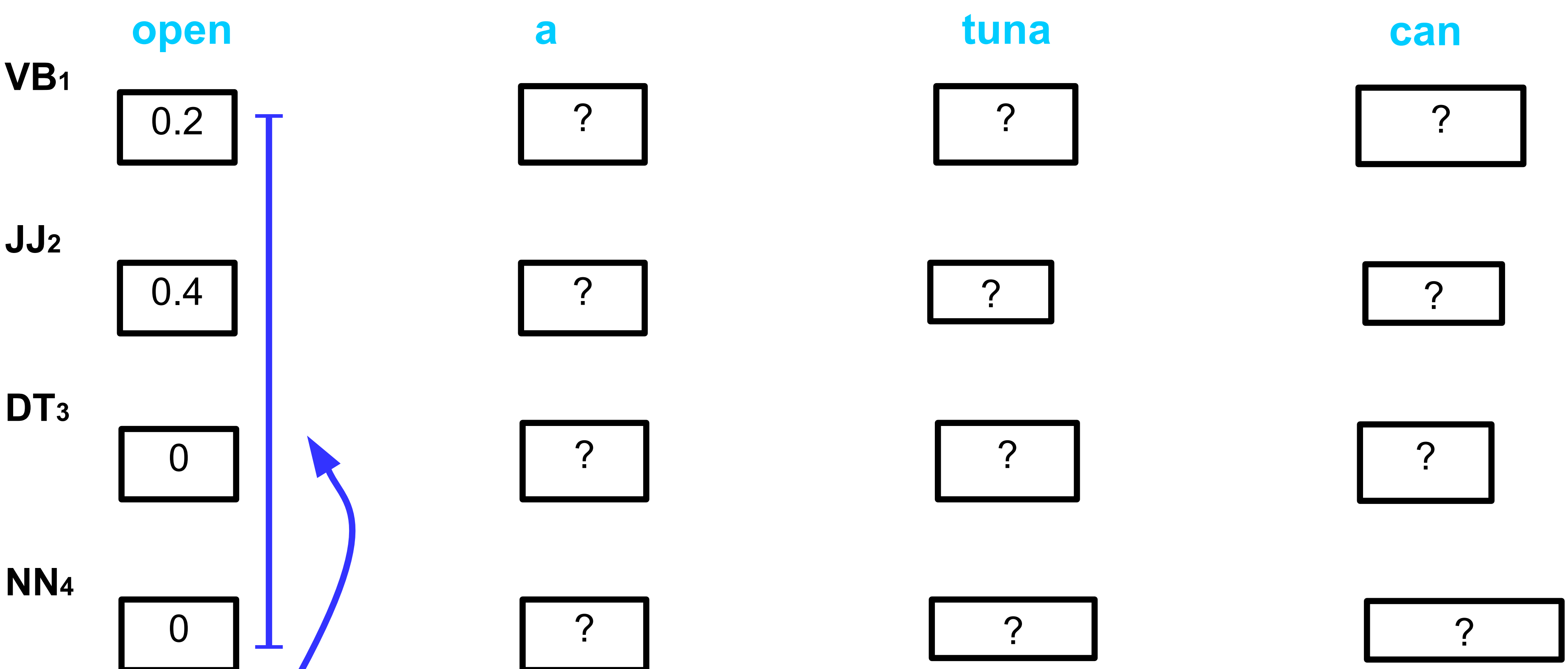
Exercise: Viterbi search

	open	a	tuna	can
VB_1	?	?	?	?
JJ_2	?	?	?	?
DT_3	?	?	?	?
NN_4	?	?	?	?

b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

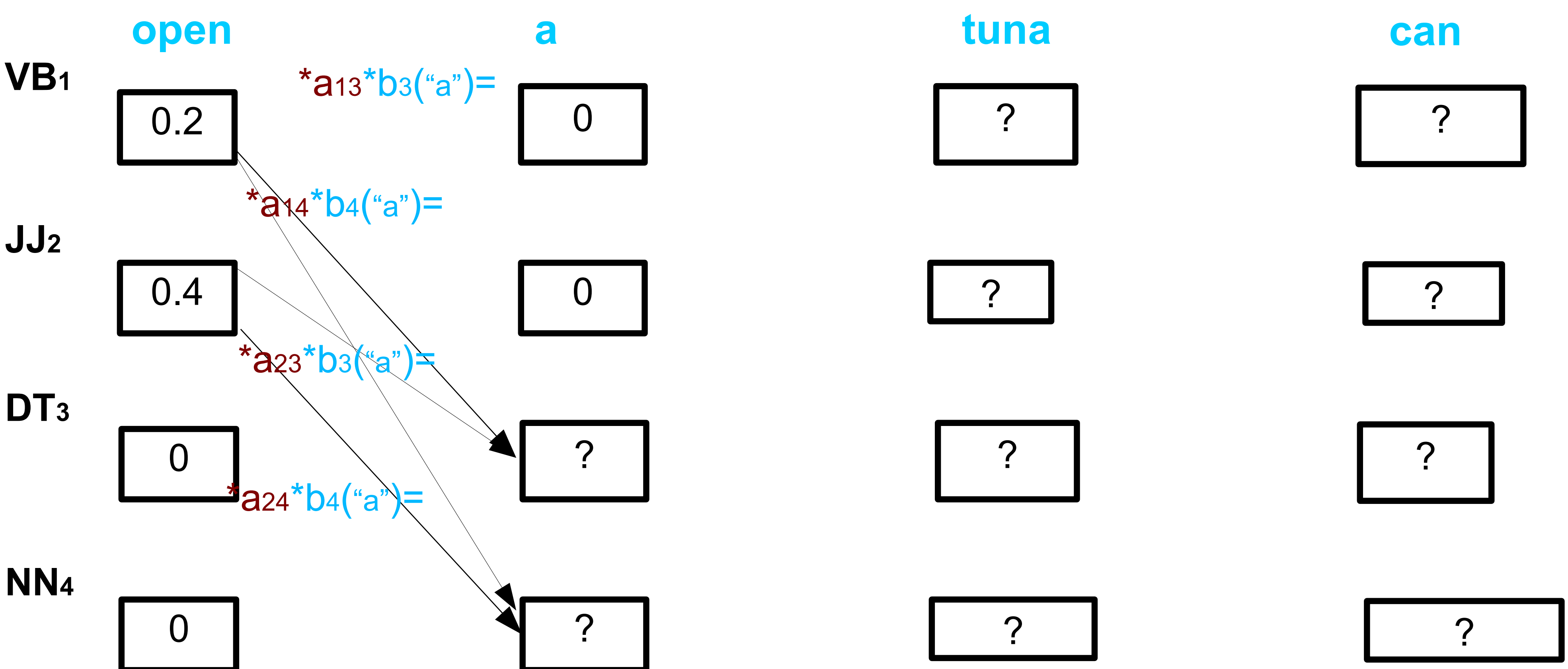
Exercise: Viterbi search



b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

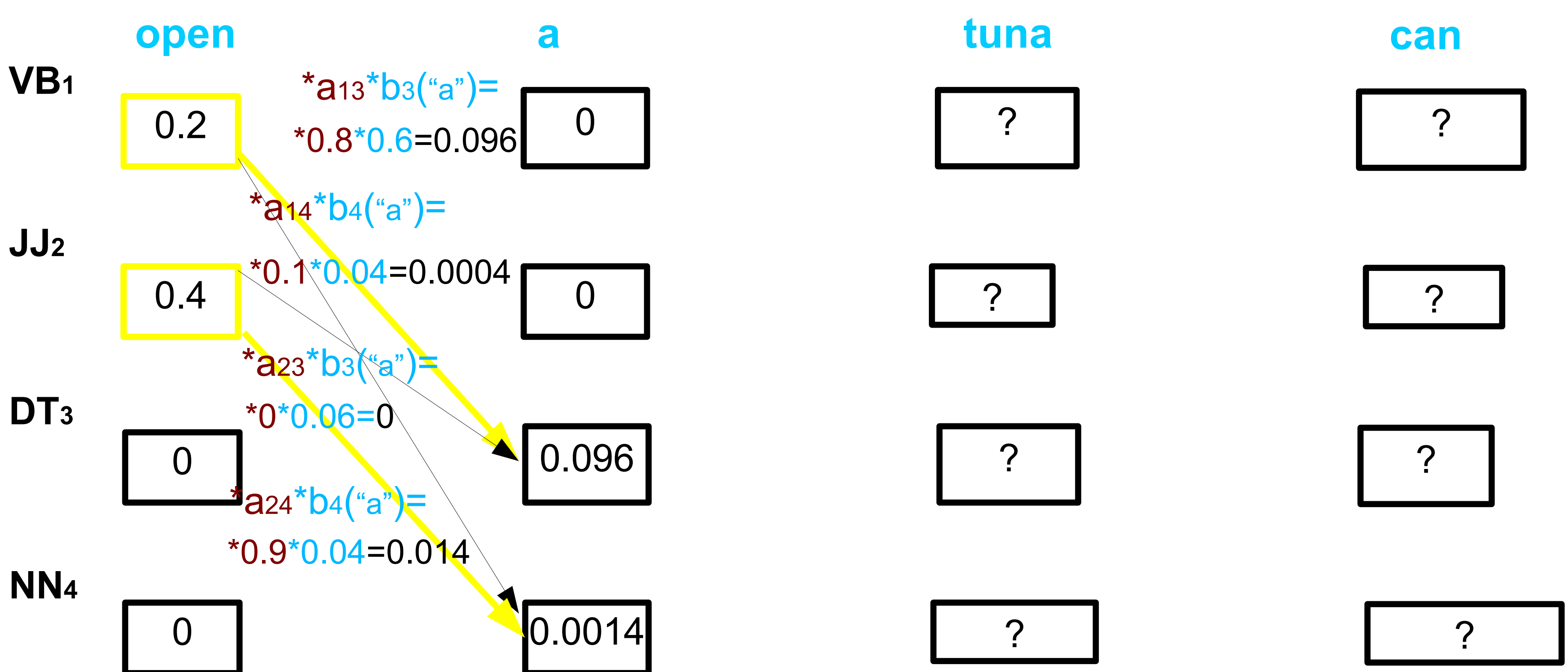
Exercise: Viterbi search



b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

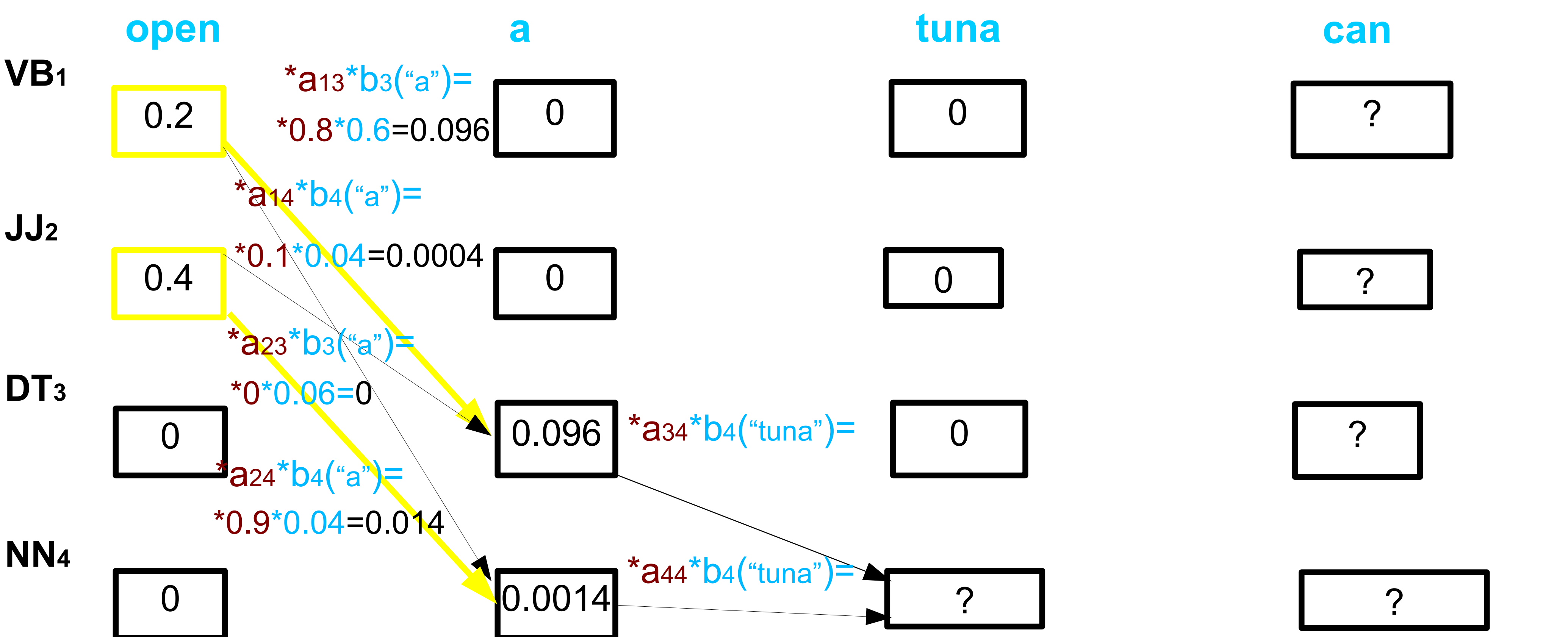
Exercise: Viterbi search



b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

Exercise: Viterbi search



bi	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

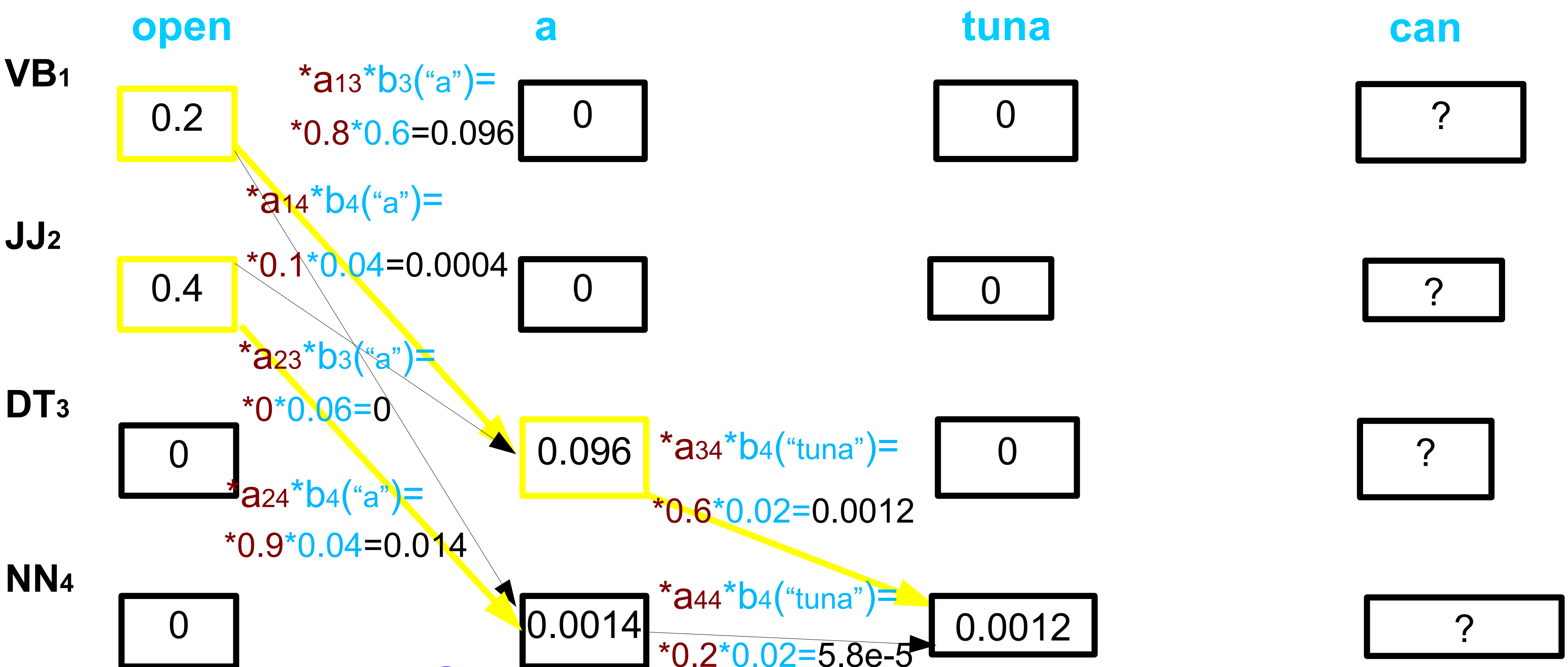
Exercise 4: HMM and Viterbi

- Go in breakout rooms, discuss with each other and propose answers for these 3 questions in **MyCourses > Lectures > Lecture 4 exercise return box**:

1. Finish POS tagging by Viterbi search example by hand.
    - Return the values of the boxes and the final tag sequence. Either take a photo of your drawing, fill in the given ppt, or just type the values into the text box
2. Did everyone get the same tags? Is the result correct? Why / why not?
3. What are the pros and cons of HMM tagger?

All submissions, even incorrect or incomplete ones, will be awarded by one activity point.

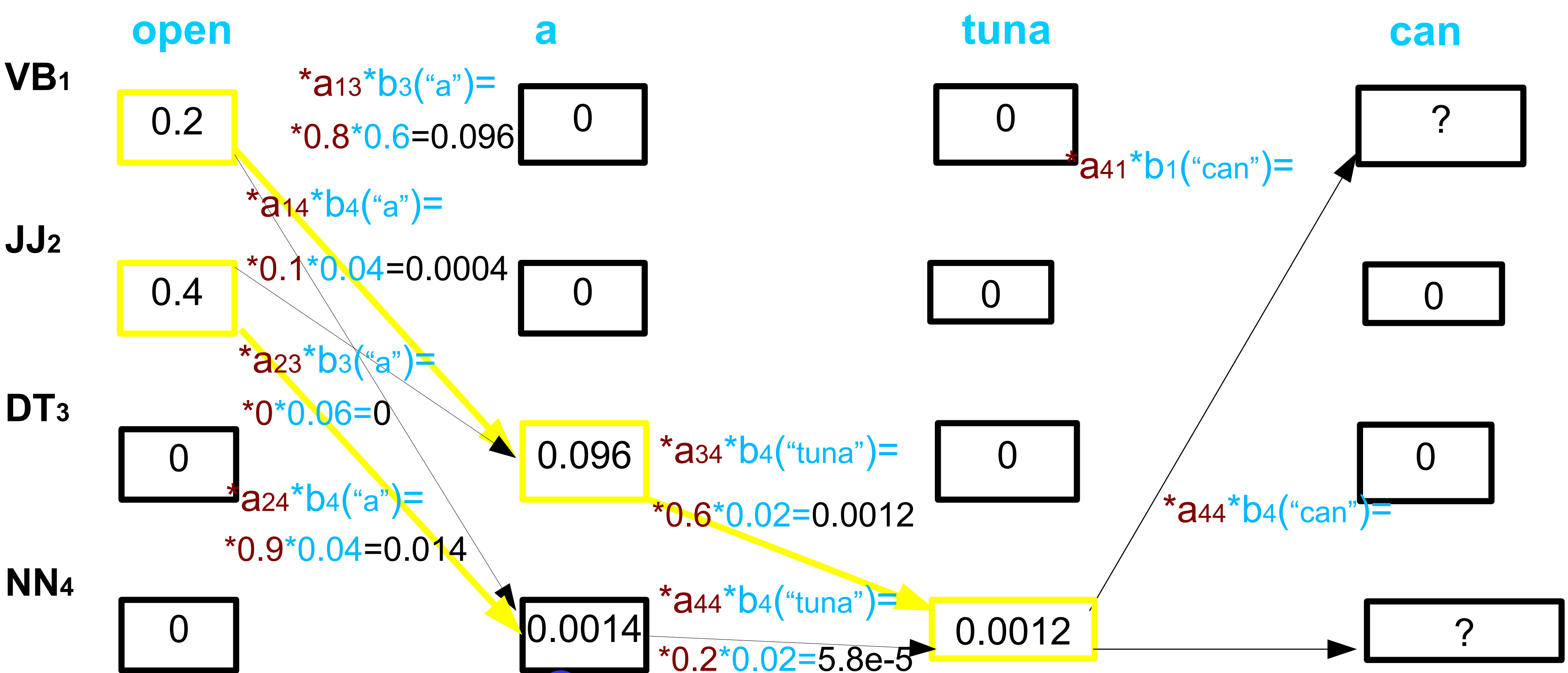
Exercise: Viterbi search



b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

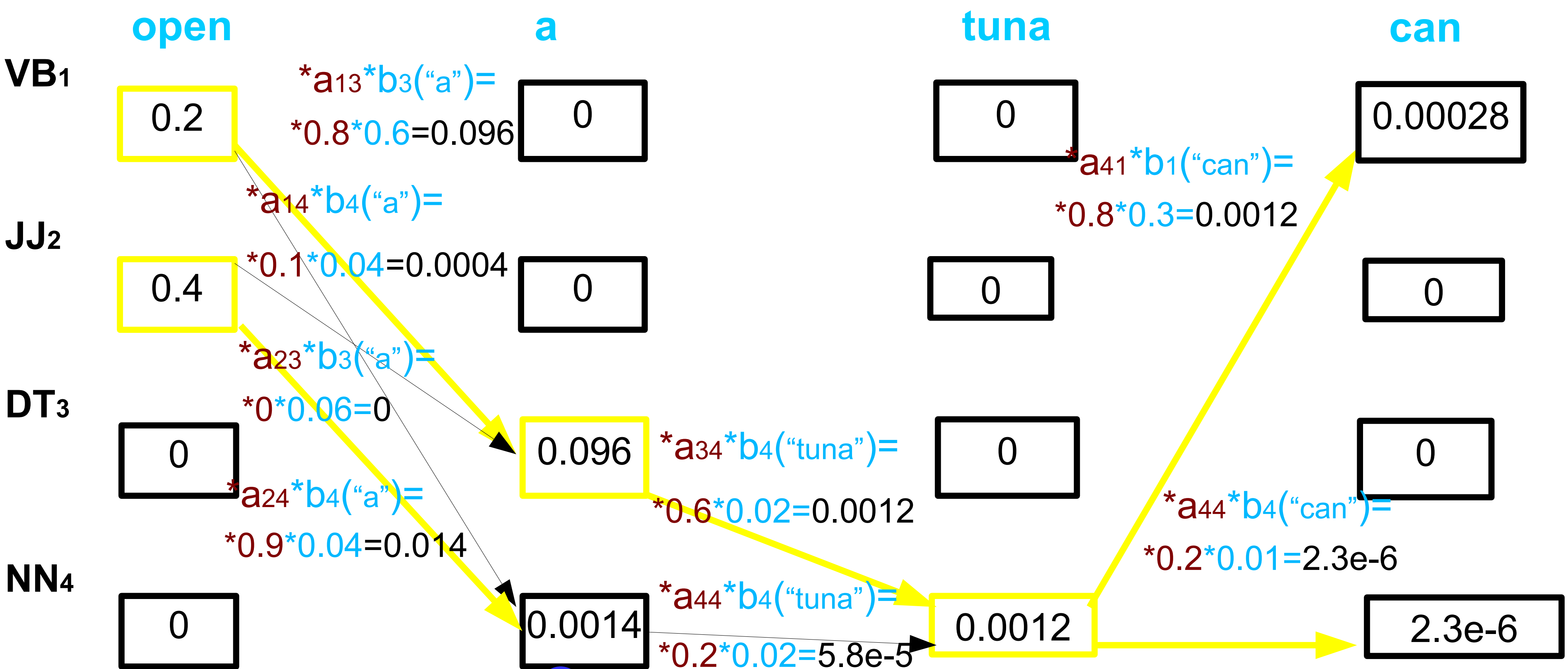
Exercise: Viterbi search



bi	open	a	tuna	can
VB1	0.2	0	0	0.3
JJ2	0.4	0	0	0
DT3	0	0.6	0	0
NN4	0	0.04	0.02	0.01

aij	VB1	JJ2	DT3	NN4
VB1	0	0.1	0.8	0.1
JJ2	0	0.1	0	0.9
DT3	0	0.4	0	0.6
NN4	0.8	0	0	0.2

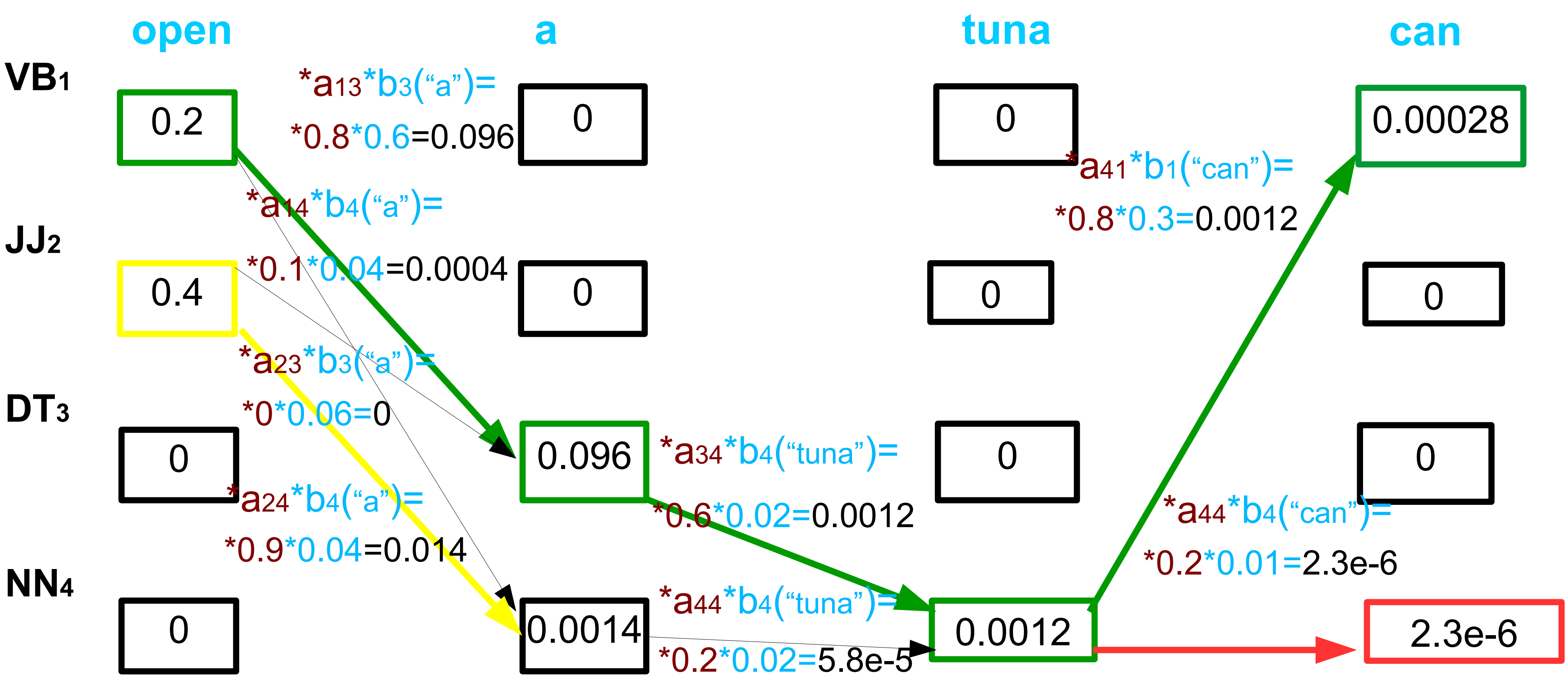
Exercise: Viterbi search



b_i	open	a	tuna	can
VB_1	0.2	0	0	0.3
JJ_2	0.4	0	0	0
DT_3	0	0.6	0	0
NN_4	0	0.04	0.02	0.01

a_{ij}	VB_1	JJ_2	DT_3	NN_4
VB_1	0	0.1	0.8	0.1
JJ_2	0	0.1	0	0.9
DT_3	0	0.4	0	0.6
NN_4	0.8	0	0	0.2

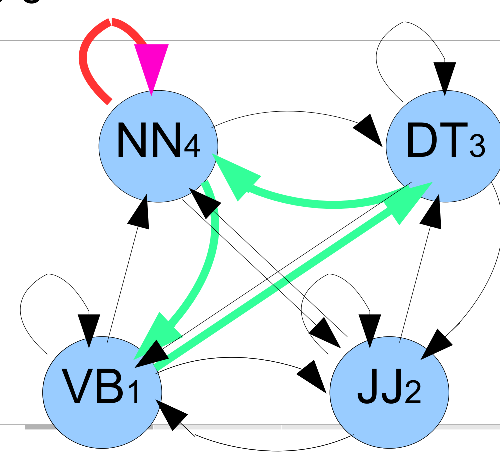
Solution: Viterbi search



Answer: “Open-a-tuna-can”

=> VB-DT-NN-VB

(why not **VB-DT-NN-NN**?)



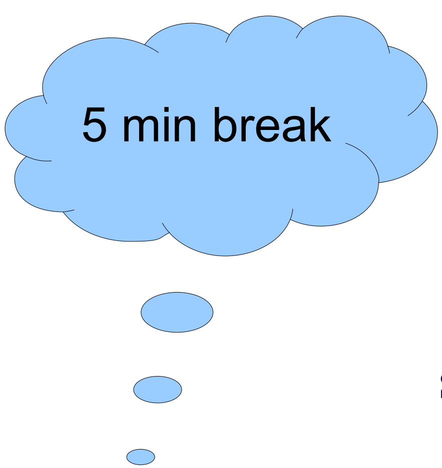
Discussion

- How would you start building a part-of-speech tagger?
- Or a named entity recognizer for news articles?

- Is it possible without any *understanding* by just counting statistics?
- If not, what is the problem?

Hints for solving the Viterbi exercise

- Some tags have zero probability, e.g. “tuna” can only be a noun, never verb, adjective, determiner
  - No need to compute paths which will be zero, anyway
- Some transitions have zero probability, e.g. verb-verb or noun-determiner
  - No need to compute those paths, either
- Once you have done the computations, back-track the path to read the overall best sequence



Decoding the HMM

- **Given an observation sequence,**

$$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_T\}$$

Here: Words X={x_1,…,x_T}

- **Find the single best sequence of states,**

$$q = \{q_1, q_2, \cdots, q_T\}$$

Here: Tags Y={y_1,…,y_T}

- **Which maximizes,**

$$P(\mathbf{O}, q \mid \lambda)$$

Here: P(X,Y | A,B)

Viterbi algorithm

1. Initialization $\delta_1(i) = \pi_i b_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t)$$

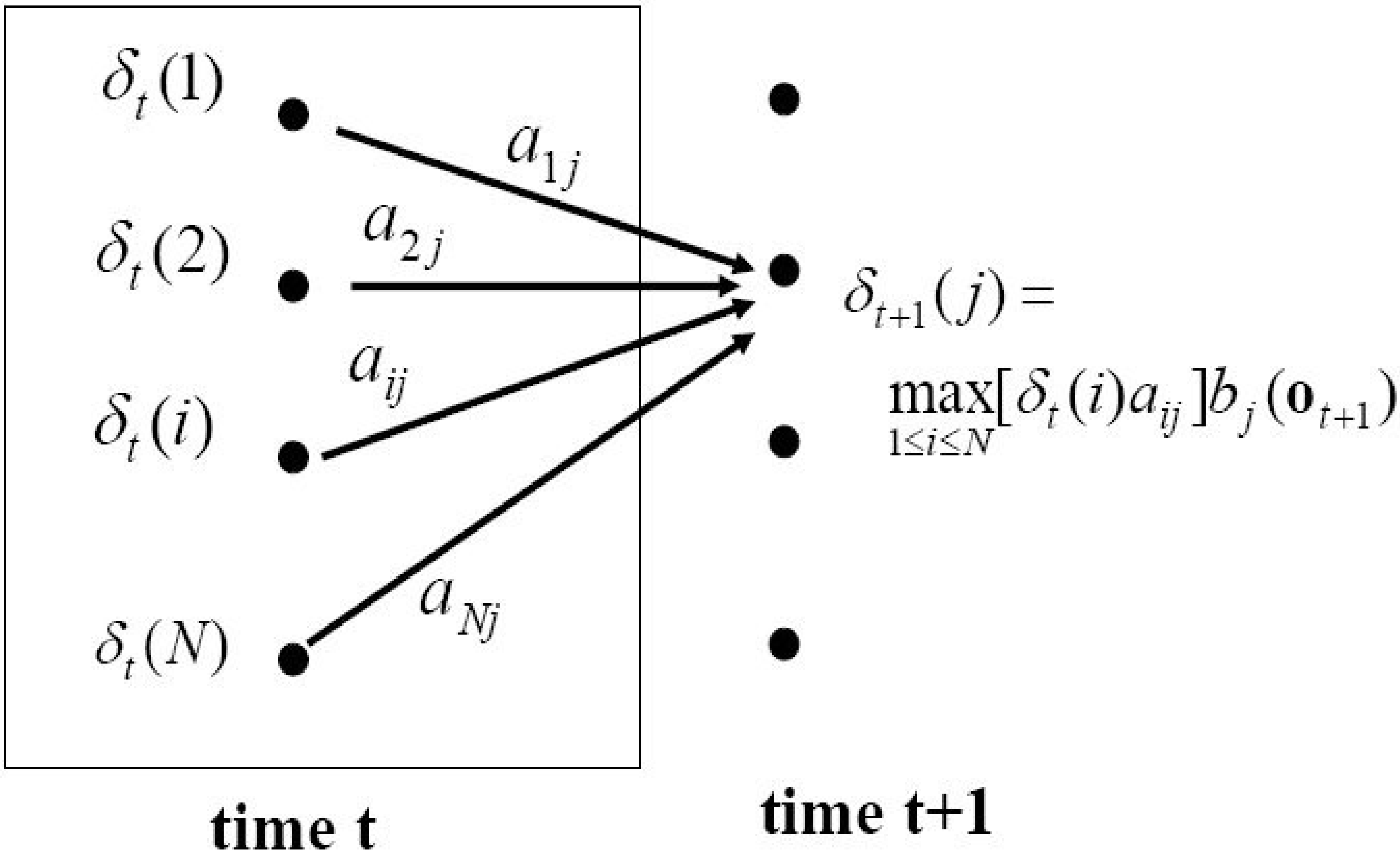
$$\psi_t(j) = \arg\max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}]$$

3. Termination $P^* = \max_{1 \le i \le N}[\delta_T(i)] \quad q_T^* = \arg\max_{1 \le i \le N}[\delta_T(i)]$

4. Path Back trace $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Picture by B.Pellom

Viterbi step 2: Recursion



Estimation of HMM parameters

- For corpora annotated with POS tags
  - Just count each tag observations $P(x(t)|y(t))$
  - And tag transitions $P(y(t)|y(t-1))$
- For unknown data use e.g. Viterbi to first estimate labels and then re-estimate parameters and iterate

Parsing

- Who did what to whom?
- Language dependent rules
  - Context-Free Grammar (CFG)
  - English: Pekka bought a car.
    - “The first noun is the subject”
    - “The noun after the verb is the object”
  - Finnish: Pekka osti auto**n**. / Auto**n** osti Pekka.
    - “The case of the noun marks the semantic role”

Probabilistic context free grammars

- Each production rule will have a probability
- Probabilities estimated from a large annotated corpus

Even better POS tags? Discriminative models

- Use previous words and tags as features
- The context is computed from a sliding window
- Train a classifier to predict the next tag
  - Jurafsky: Maximum entropy Markov model (MEMM)
  - Support vector machine (SVM)
  - Deep (feed-forward) neural network (DNN)
  - Conditional random field (CRF) is a bidirectional extension of MEMM that uses also tags on right
  - Combining bidirectional recursive DNN and CRF[1]

[1] D.Porjazovski, J.Leinonen, M.Kurimo. Named Entity Recognition for Spoken Finnish. In AI4TV 2020, ACM.

Recurrent neural network tagger

- No fixed-length context window
- Loop in the hidden layer adds an infinite memory
- Can provide word-level tags:
  - POS or NER
- Or sentence-level tags:
  - Sentiment analysis
  - Topic or spam detection

Maximum entropy models

- Represents dependency information by a weighted sum of features f(x,h)

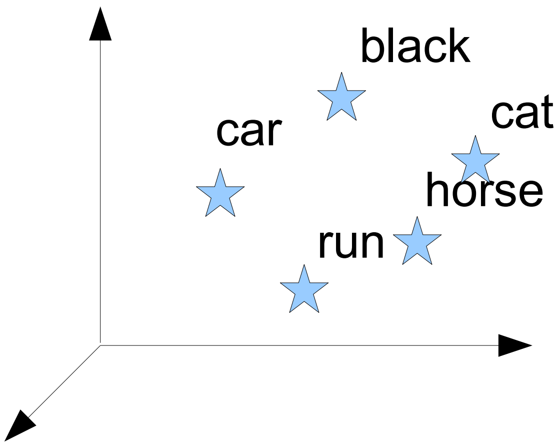
$$P(x|h) = \frac{e^{\sum_i \lambda_i f_i(x,h)}}{\sum_{x'} e^{\sum_j \lambda_j f_j(x',h)}}$$

- Features X can be e.g. **tags and words**
  - Previous tags y(t-1), y(t-2)
  - Word x(t) and previous words x(t-1), x(t-2)
- Alleviates the data sparsity problem by smoothing the feature weights (lambda) towards zero
- Resemble to MaxEnt language models [2]
- Called Maximum Entropy Markov Models (MEMM) in Jurafsky's text book

---

[2] T.Alumäe, M.Kurimo. Domain adaptation of maximum entropy language models. Proc. ACL 2010.

Mapping words into continuous space

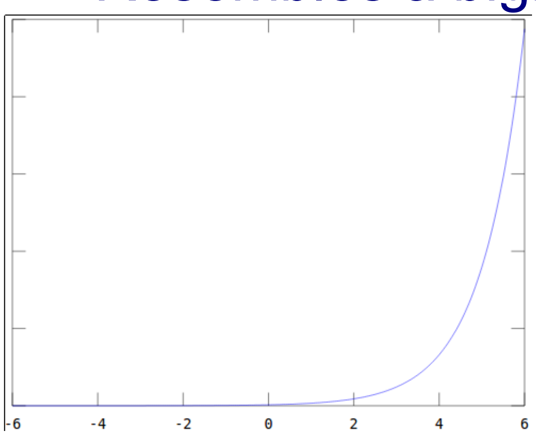


- Map words into a continuous vector space to learn a distributed representation known as *word embedding*
- The goal is to use a vector space that keeps similarly behaving words near each other
- Words can be clustered by context, e.g. n-gram probabilities
  - *word2vec* [3] is one widely used option
  - Other embeddings to reflect various contextual properties

[3] T.Mikolov et al. Efficient Estimation of Word Representations in Vector Space. 2013. ArXiv:1301.3781.

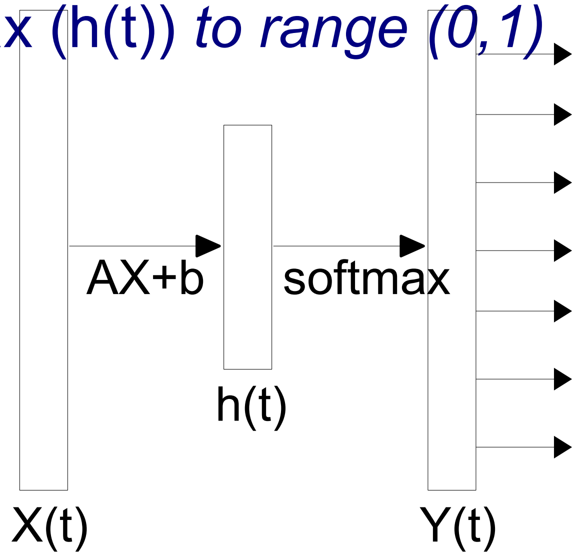
A simple bigram NN tagger

- Outputs the *probability of tags* Y(t) given the word x(t) and tag y(t-1)
- **Input layer** maps the word x(t) and previous tag y(t-1) as an input vector X(t)
- **Hidden layer** has a linear transform h(t) = AX(t) + b to compute a representation of *linear distributional features*
- **Output layer** maps the values by Y(t) = softmax (h(t)) *to range (0,1) that add up to 1*
- Resembles a bigram maximum entropy model



Softmax:

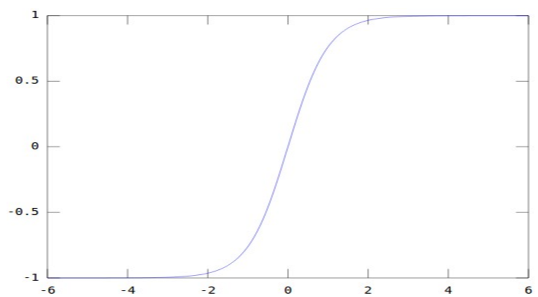
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



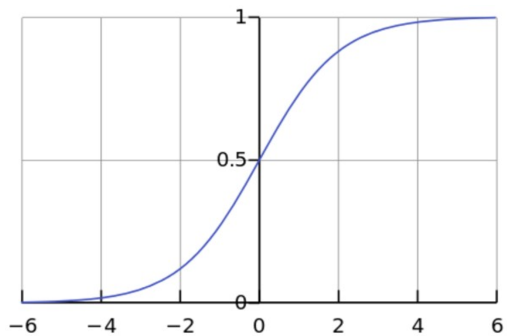
Note: Here X(t) contains both word x(t) and tag y(t-1)

A non-linear bigram NN tagger

- The only difference to the simple NN tagger is that the hidden layer h(t) now includes a non-linear function $h(t) = U(AX(t) + b)$
- Can learn more complex feature representations
- Common examples of non-linear functions U:

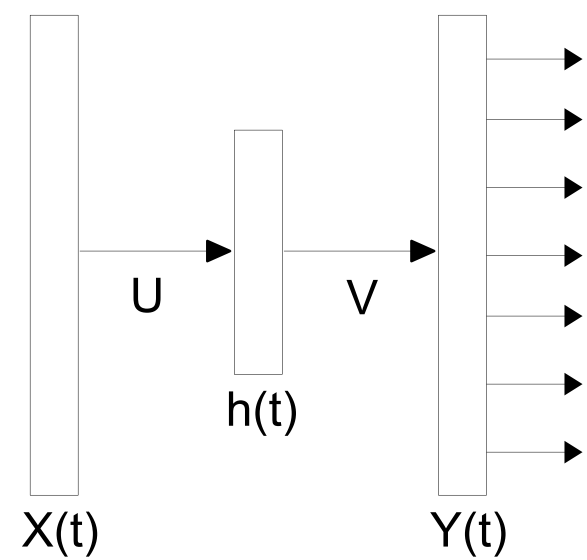


$$U(t) = \tanh(t)$$



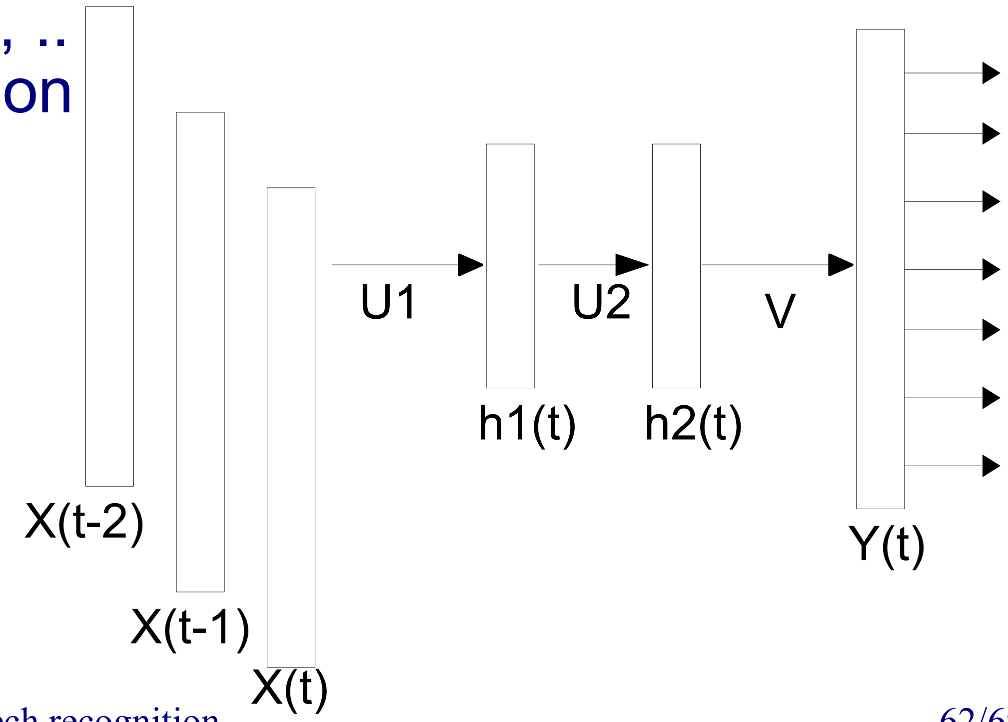
Sigmoid

$$U(t) = \frac{1}{1 + e^{-t}}$$



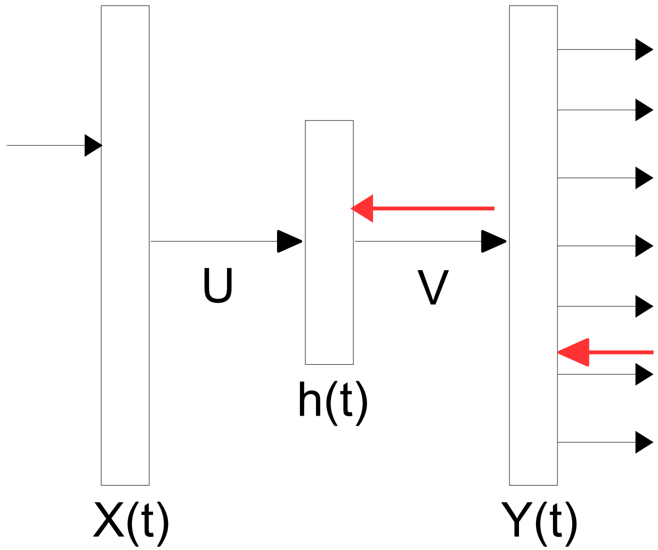
Common NN extensions

- ***Input layer*** is expanded *over several previous words* x(t-1), x(t-2), .. and tags y(t-1), y(t-2), .. to learn richer representations
- *Deep neural networks* have *several* ***hidden layers*** h1, h2, .. to learn to represent information at several hierarchical levels



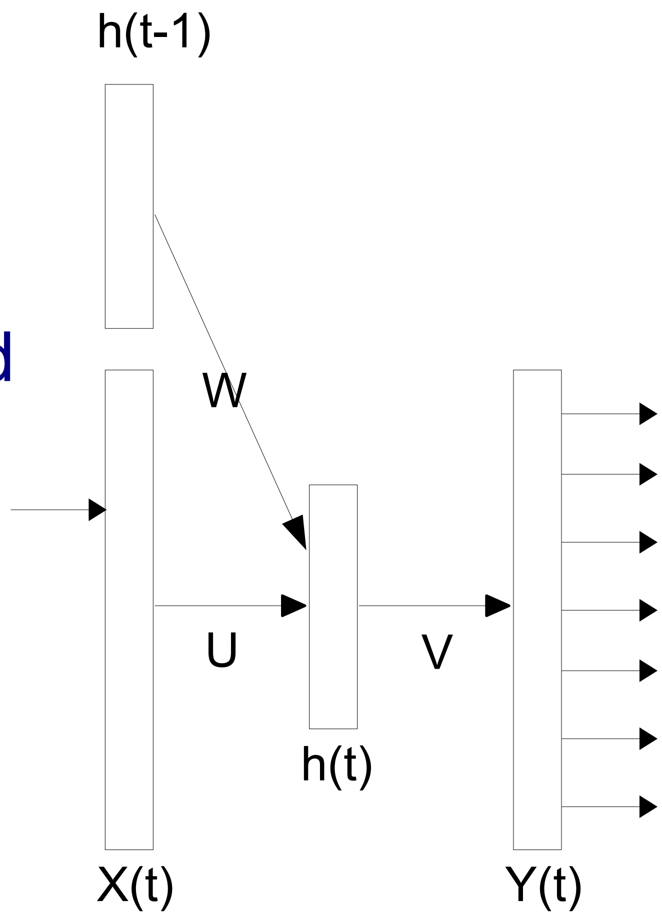
NN tagger training

- Supervised training minimizes the **output errors** by training the weights for V by *stochastic gradient descend*
- Propagate the output **error to hidden layer** to train the weights for U
- In practice, a deep NN will require more complex training procedures, since the gradients *vanish* quickly



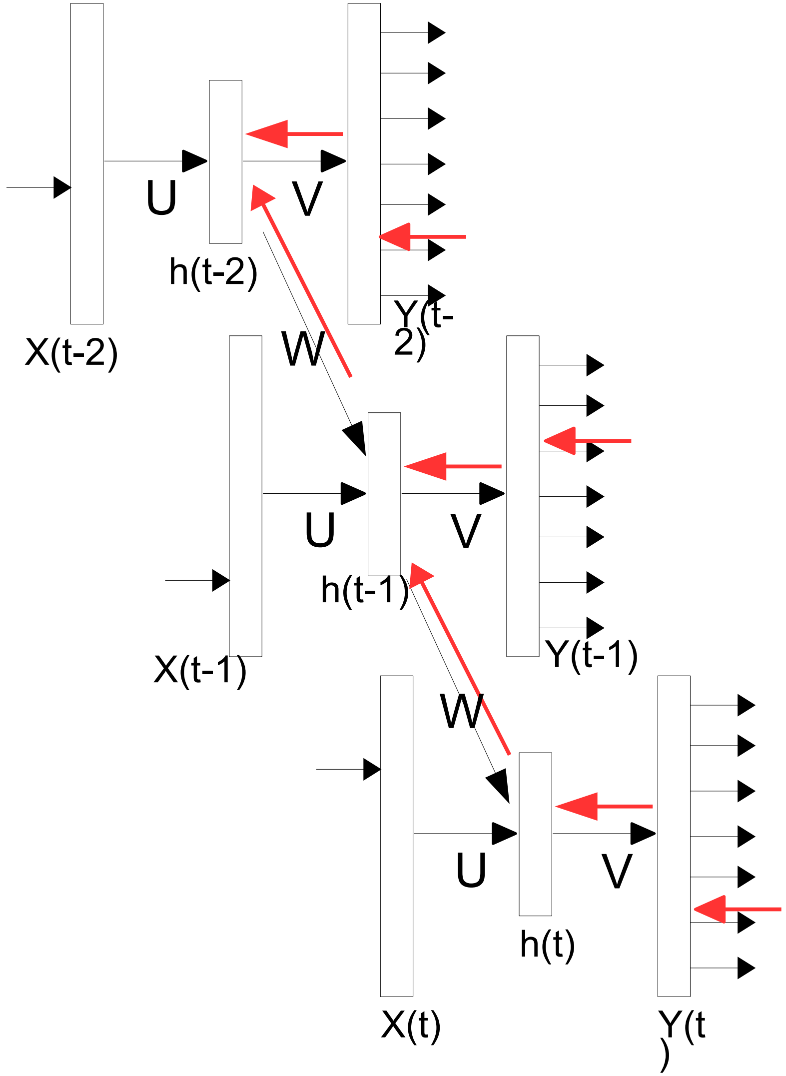
Recurrent Neural Network (RNN) tagger

- **Looks like** a bigram NN tagger
- **But**, takes an additional input from the hidden layer of the *previous time* step
- Hidden layer becomes a compressed representation of the word history
- Can learn to represent unlimited memory, in theory



RNN tagger training

- Minimizes the output error by training the weights by *stochastic gradient descend*
- Propagates the output error to all *layers and time* steps (called *backpropagation through time)* to train the hidden layer
- **Looks now like** a very deep neural network with shared weights U and W



References

- Manning, C. D. and Schütze, H. (1999). Foundations of Statistical Natural Language Processing. The MIT Press. (Chapters 9-12)
- Jurafsky, D. and Martin, J. H. (2008). Speech and Language Processing. Prentice Hall. 2nd edition. (Chapter 4)
- Jurafsky, D. and Martin, J. H. (2018). Speech and Language Processing. 3nd edition. (Chapters 8, 9)
- Silfverberg M., Ruokolainen T., Linden K, and Kurimo M. (2006). FinnPos: An Open-Source Morphological Tagging and Lemmatization Toolkit for Finnish. Journal of Language Resources and Evaluation. DOI 10.1007/s10579-015-9326-3
- Ruokolainen, Teemu (2016). Contributions to Morphology Learning using Conditional Random Fields. Aalto University, Doctoral dissertation, 2016. https://aaltodoc.aalto.fi/handle/123456789/20370
- Porjazovski, Dejan (2020). End-to-end named entity recognition for spoken Finnish. Aalto University, MSc thesis, 2020. https://aaltodoc.aalto.fi/handle/123456789/47383
- Markov, A. A. (1913). An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. (In Russian.) Imperial Academy of Sciences of St. Petersburg 7(3):153–162.

Feedback

Go to **MyCourses > Lectures > Feedback for Lecture 4** and fill in the form.

Some of the feedback from the previous week:

+ A lot of important information and concepts. Practical examples are great.

+ I really liked the break-out room discussions and exercise. The exercise really activated me during the lecture

+ Video from Stanford for Good Turing

+ The guest lecturer from a company was a good addition!

- A short break (max 5 mins) would have been nice

- Maybe the lecture material should be shorter since we failed to finish it all

- I don't think I understood the NN section. Hopefully it will be covered in more detail in another lecture

- I’m missing kahoot quiz, it is better than breakout rooms

- I think the group of discussion would be better if there are 4-5 participants

Thanks for all the valuable feedback!

Reminder: Project DLs

- Topic selection: submit a team abstract (one-paragraph description of the intended topic). Deadline 3 February
- Project plan and Literature survey: Deadline 10 March
- Peer grading for the Project plan and the Literature survey: Deadline 17 March
- Reaction to peer grading: 24 March
- Full project report: submission of the final report. See the details below. Deadline 28 April
- Project Presentation video (5 min): Deadline 5 May
- Vote for the best Project Presentation video: Deadline 19 May

Follow MyCourses for updates!

First home assignment DLs

Assignment	Released	Returned
00-intro	13 Jan	23 Jan
01-text	18 Jan	31 Jan
02-ngrams	25 Jan	7 Feb
03-word2vec	1 Feb	14 Feb
04-POS	8 Feb	28 Feb
Continues in March		