- Jurafsky, D. and Martin, J. H. (2009): Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Second Edition. Pearson: New Jersey: Chapter 5
- Lafferty, J., McCallum, A., Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. Proc. 18th International Conf. on Machine Learning. Morgan Kaufmann. pp. 282–289. http://www.cis.upenn.edu/~pereira/papers/crf.pdf.
- Lipton, Z.C., Berkowitz, J., Elkan C. (2015): A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 https://arxiv.org/pdf/1506.00019.pdf

HMM, MEMM, CRF, LSTM

SEQUENCE TAGGING

SEQUENCE TAGGING

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- We want to know properties of words for further processing, e.g. word classes, names, etc.
- It is possible to learn a method that assigns these properties from **labeled training text**.
- In Machine Learning, this is a classification task. If the sequence of events is taken into account, this is called **sequence tagging**

Examples for tagged text:

- Part-of-Speech:
  `I/PRO saw/V the/DET man/N with/P the/DET saw/N ./P`
- Name tagging:
  `Valerie/B-PERS and/O Rose/B-PERS travel/O to/O New/B-LOC York/I-LOC ./O`

NO INDEPENDENCE ASSUMPTION ON SAMPLES

- Standard ML setups: Assumption on the independence of training resp. test examples
  - Can shuffle and sample training examples
  - Can classify test examples in parallel
- Sequence Learning
  - Previous train/test examples are an informative context
  - Previous classifications/outputs are an informative context
- Examples for sequential data:
  - Frames from video
  - Snippets from audio
  - Text: streams of words or characters
  - DNA

PARTS-OF-SPEECH

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- About 8 coarse classes: Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
- Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
- Lots of debate in linguistics about the number, nature, and universality of these. We ignore this debate.

Why distinctions can be tricky:

- Singular and plural different POS?
- Common nouns vs. proper nouns?
- tense and number of verbs
- domain-specific POS like in

```
@joe: check out http://bit.ly/dflwkln #funny
```

SOME WORDS ON POS TAG SETS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- POS tags are linguistic categories, defined by a linguistic theory
- Different linguistic theories define different POS tags
- Number of tags differ within different tag sets for the same language
- Number of tags differ considerably for different languages:
  - German: STTS 54 tags
  - Prague Treebank (Czech): 1300+ tags
- Some tag sets are hierarchical: e.g. NOUN-Plural vs. NOUN-Singular etc.

- POS tagging is not an end application, but a preprocessing step
- POS tags are commonly used as features for higher level tasks, e.g. name tagging or chunking

THE POS TAGGING PROBLEM: AMBIGUITY, AS USUAL

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Words often have more than one POS: back

- The *back* door = JJ
- On my *back* = NN
- Win the voters *back* = RB
- Promised to *back* the bill = VB

The POS tagging problem is to determine the POS tag label sequence L for a particular sequence of words W:

$$L_{\max} = (l^1_{\max}, l^2_{\max}, \ldots l^T_{\max}) = \underset{L}{\operatorname{argmax}} \; P(L \mid W)$$

HOW HARD IS POS TAGGING?

MEASURING AMBIGUITY IN THE BROWN CORPUS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

		Original 87-tag corpus	Treebank 45-tag corpus
Unambiguous (1 tag)		**44,019**	**38,857**
Ambiguous (2–7 tags)		**5,490**	**8844**
Details:	2 tags	4,967	6,731
	3 tags	411	1621
	4 tags	91	357
	5 tags	17	90
	6 tags	2 (*well, beat*)	32
	7 tags	2 (*still, down*)	6 (*well, set, round, open, fit, down*)
	8 tags		4 (*'s, half, back, a*)
	9 tags		3 (*that, more, in*)

Brown corpus: 1 Million tokens of English, manually tagged and annotated with grammar structure

TAGS CAN BE VIEWED AS HIDDEN STATES!

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

I can see the can

TAGS: STATES OF THE HMM

Hidden States: $l_i \in$ Tagset

I can see the can

- Transition between POS-states: local dependencies of word classes

WORDS: EMISSION FROM STATES

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Hidden States: $l_i \in$ Tagset

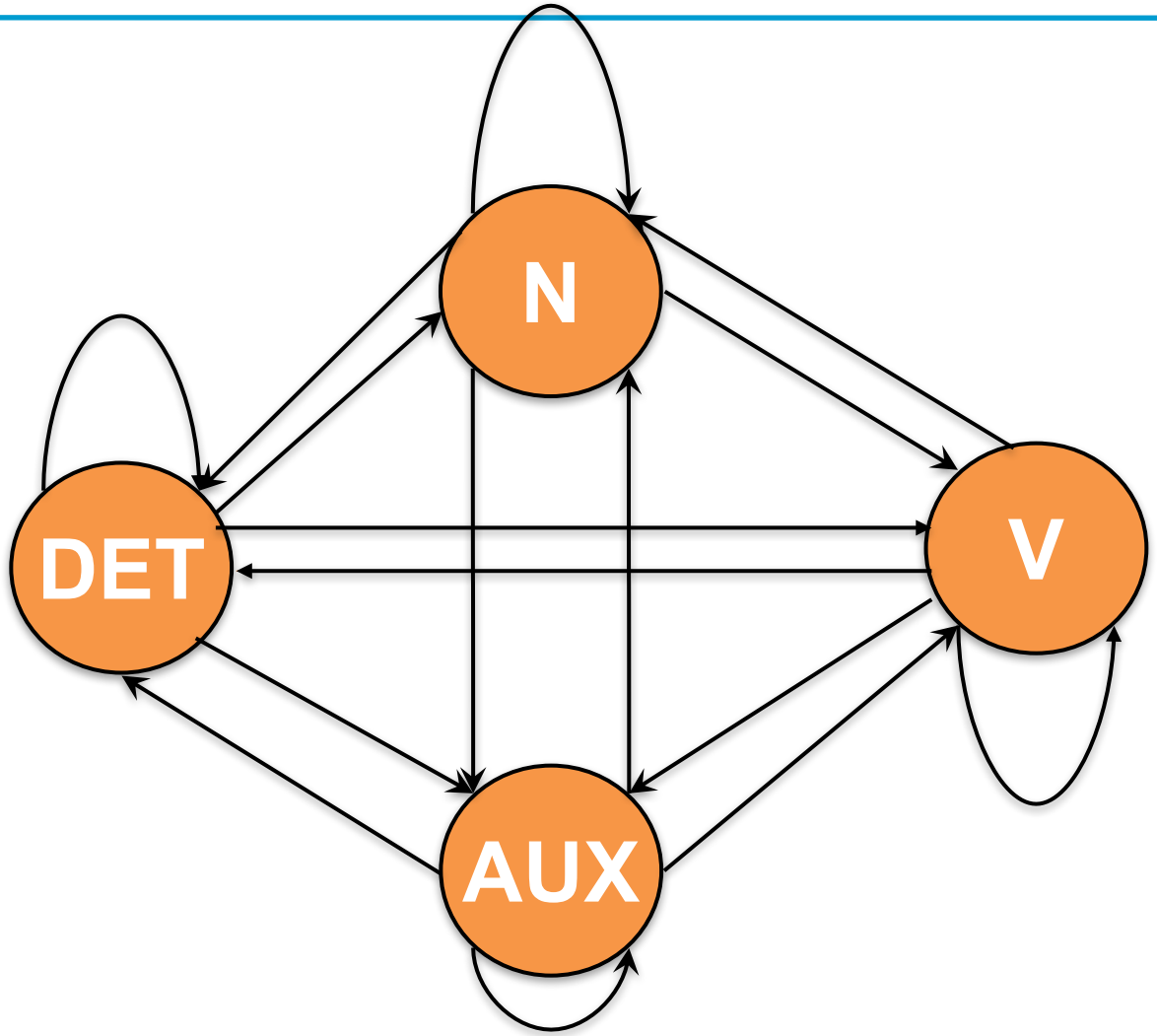
<s> PRON AUX V DET N </s>

I can see the can

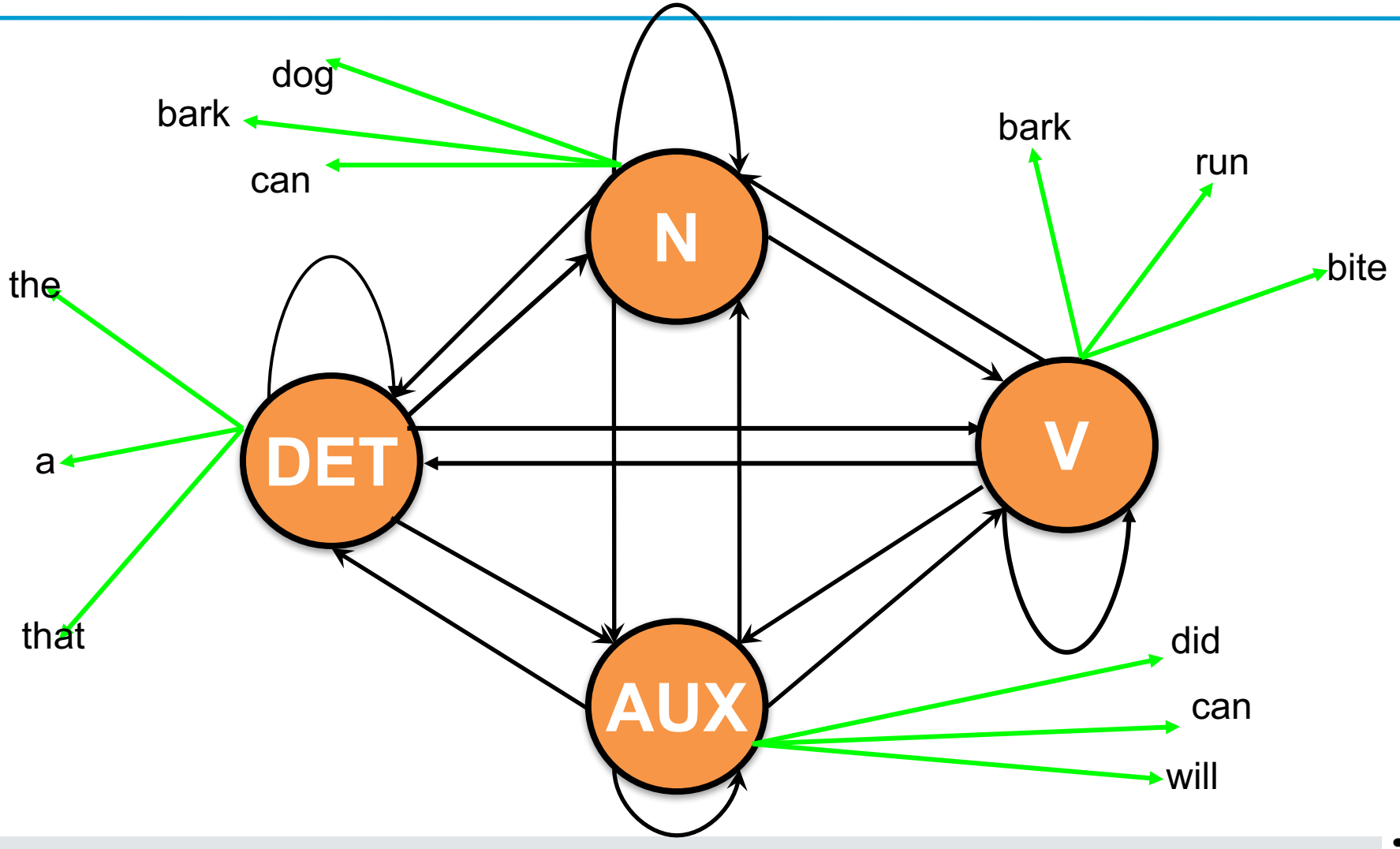
Observations: $w_i \in$ Vocabulary

- States have possible words with probabilities

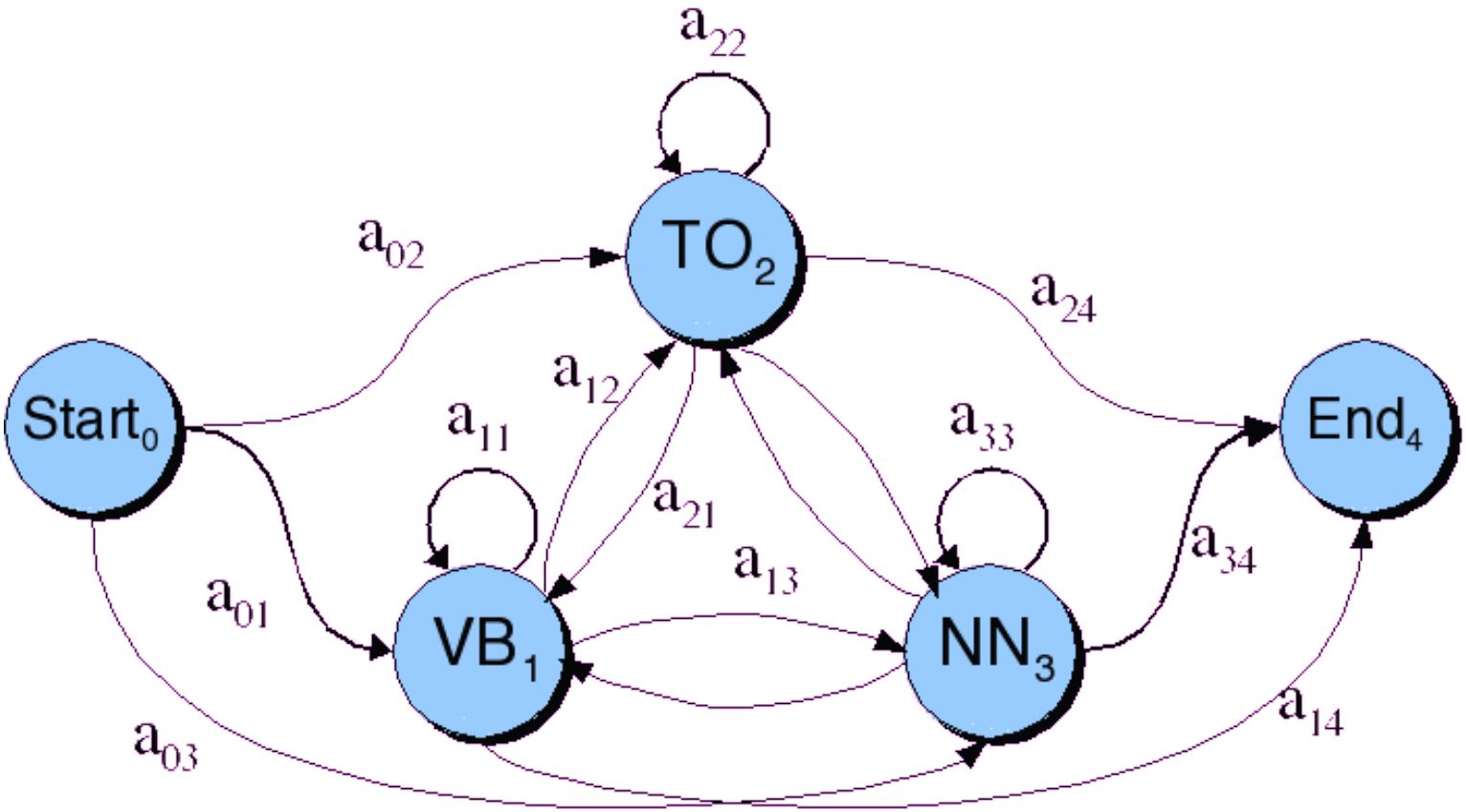
STATE TRANSITIONS



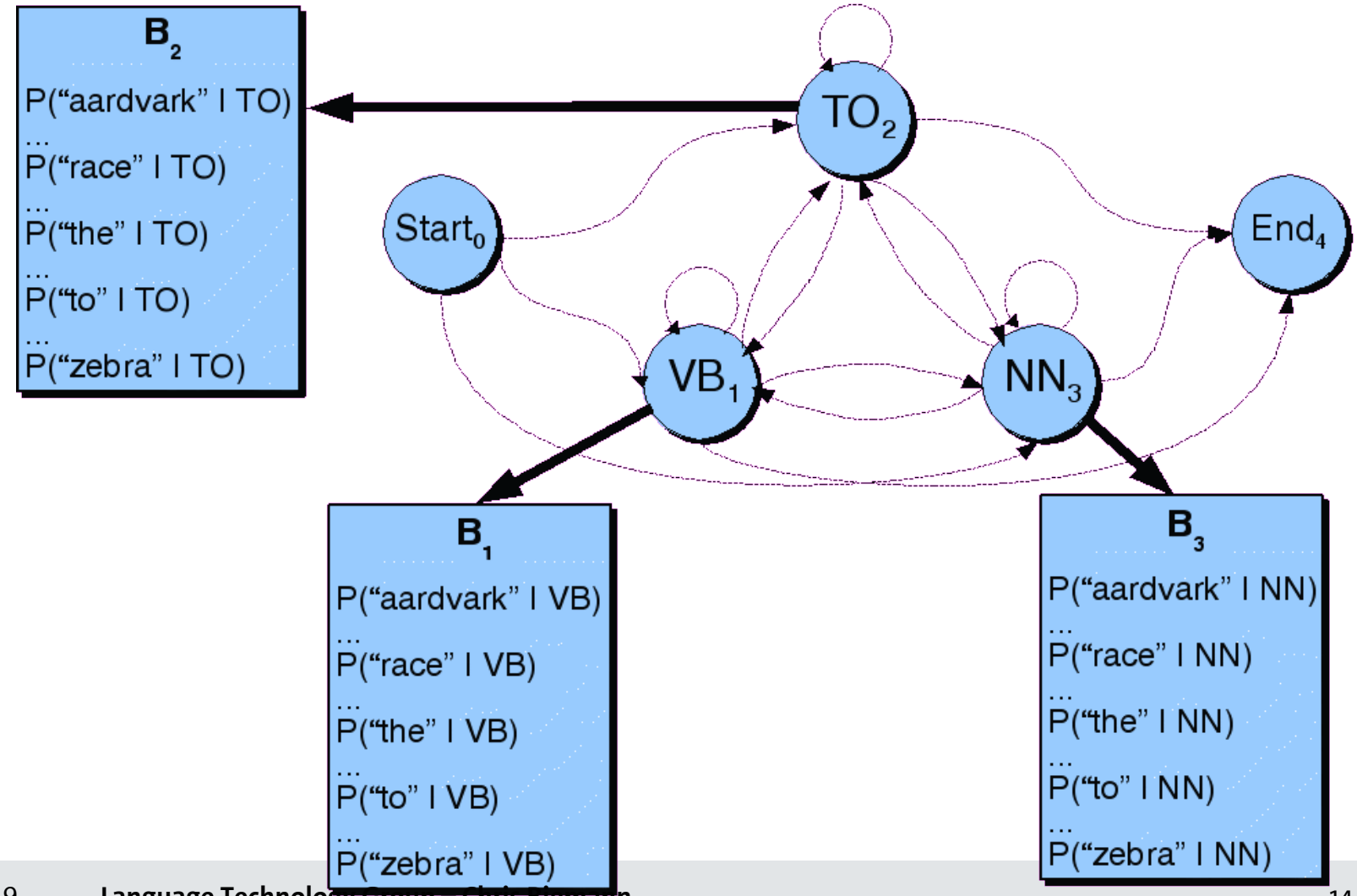
STATE TRANSITIONS AND OBSERVATIONS



TRANSITIONS BETWEEN STATES, WITH PROBABILITIES



OBSERVATION LIKELIHOODS: OTHER PROBABILITIES



STATE TRANSITION MATRIX FOR THE POS HMM

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

Tag transition probabilities $p(l_i \mid l_j)$ computed from the 87-tag Brown corpus, without smoothing.
Rows: Conditioning event, thus P(PPSS|VB) is 0.0070.

This matrix is not sparse! Is smoothing necessary?

TAG-TO-WORD PROBABILITIES FOR THE POS HMM

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Observation likelihoods computed from the 87-tag Brown corpus without smoothing.

This matrix is sparse. What do zeros mean?

THE POS HMM

We are looking for the most probable tag label sequence L_{max}, given observable words $w^1,\ldots,w^T$ and a POS HMM of order n:

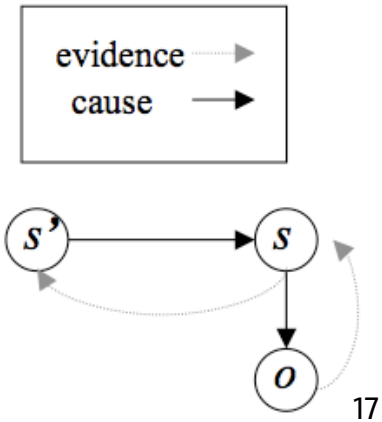
$$L_{\max}^{HMM} = (l_{\max}^1, l_{\max}^2, \ldots l_{\max}^T) \approx \underset{l^1,\ldots l^T}{\operatorname{argmax}} \prod_{t=1}^{T} P(w^t \mid l^t) P(l^t \mid l^{t-n+1} \ldots l^{t-1})$$

This a **generative model**: We were looking for labels, given words, and compute these labels using probabilities for words, given labels and the probability for the label sequence.

$P(w^t \mid l^t)$ is called the **likelihood,**

$P(l^t \mid l^{t-n+1} \ldots l^{t-1})$ is called the **prior.**

Dependency graph



Generative: we are reasoning 'backwards' from models that could have produced such an output.

POS HMM TRAINING AND APPLICATION

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Training:

- The POS HMM is trained from POS-annotated training text
- Thus, the transitions between 'hidden' states are observable during training

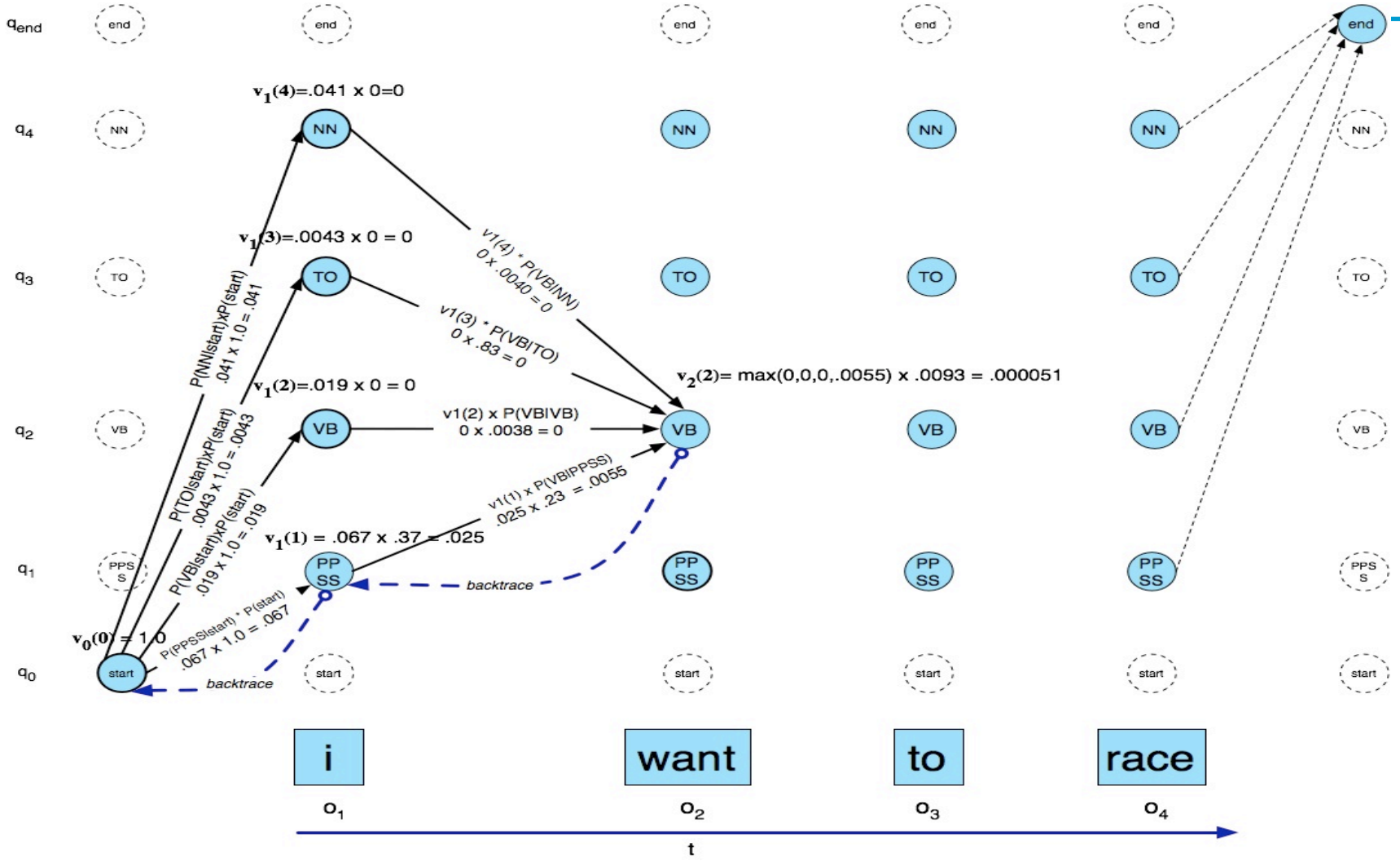
➔ no need for EM training: we can simply count and smooth MLE values using any of the smoothing methods if we have fully labeled training text

➔ we can use EM training if we have unlabeled text and a tag distribution per word

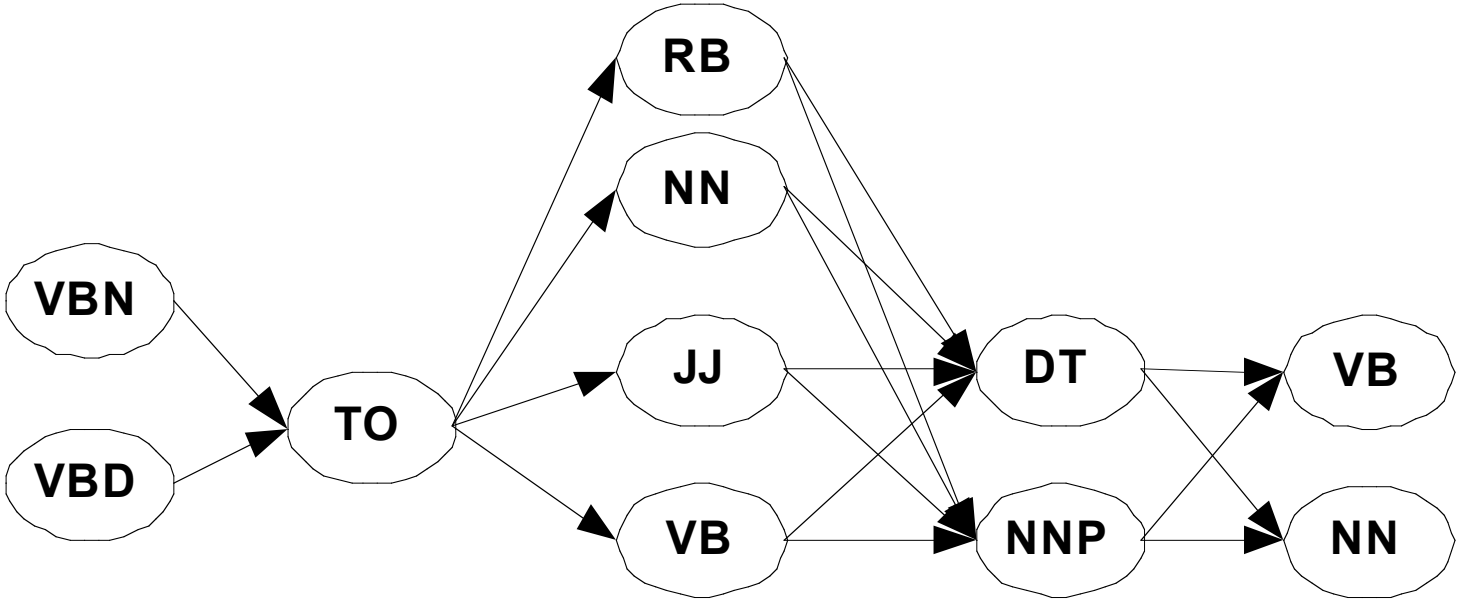
Application:

- for unlabeled text, label state transitions are hidden
- we compute the most likely state sequence using a modification of the Viterbi algorithm that accounts for likelihoods

VITERBI EXAMPLE WITH LIKELIHOODS AND PRIORS



DISAMBIGUATION: FIND THE MOST PROBABLE TAG SEQUENCE



promised to back the bill

Structural zeros on p(w|l) give rise to optimization techniques

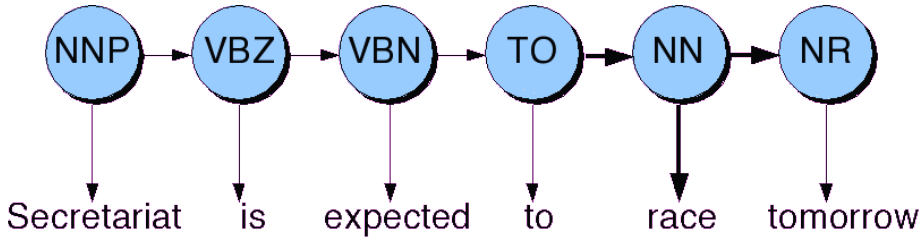
GENERATIVE VS. DISCRIMINATIVE MODELS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

We want: $\underset{L}{\text{argmax}}\ P(L|W)$

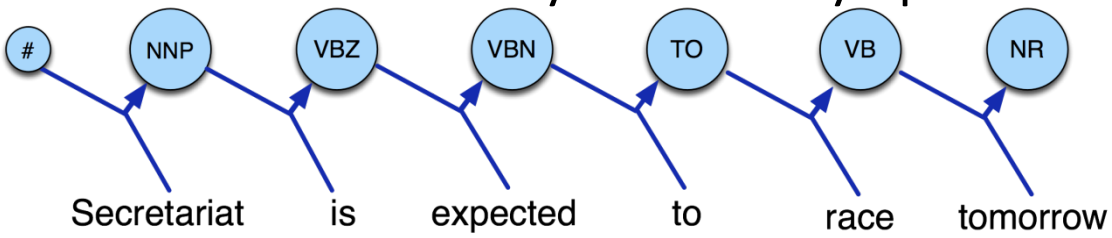
Generative models: Bayes rule is used to turn the dependencies around:

$$\underset{L}{\text{argmax}}\ P(L|W) = \underset{L}{\text{argmax}}\ P(L,W)P(W) = \underset{L}{\text{argmax}}\ P(L,W) = \underset{L}{\text{argmax}}\ P(W|L)P(L)$$



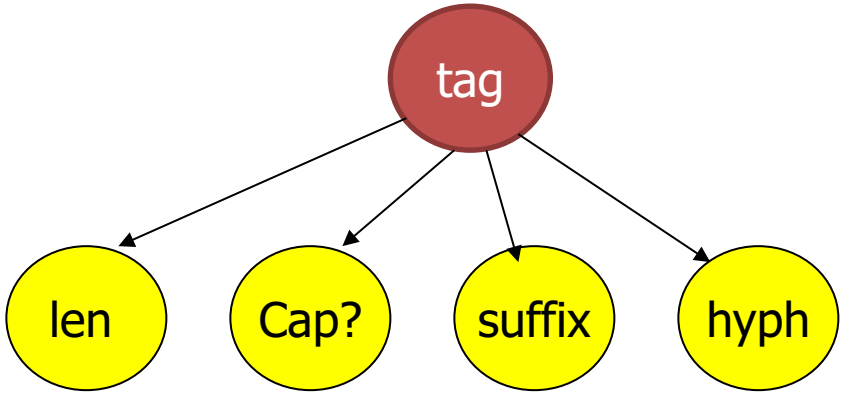
what we model is the joint probability of labels and observations P(L,W).

Discriminative models: Why not directly optimize P(L|W)?



DIFFICULTIES WITH HMMS

- Need a richer representation of words: Describe words with features instead of enumerating
- Example features:
  - capitalization
  - word ending
  - word length
  - hyphenation



Split states to account for features, e.g. NN -> (NN,Cap) and (NN,noCap) ?
Introduce layers? All this increases the number of parameters.

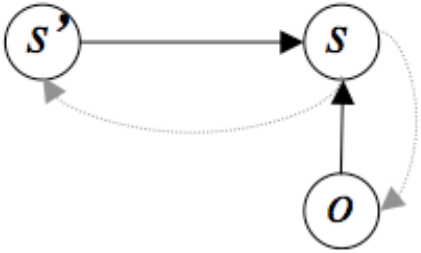
- Since we have given fixed word observations for tagging, why model the joint probability of observations and labels instead of P(L|W)?

CONDITIONAL MARKOV MODEL (CMM), MAXIMUM ENTROPY MARKOV MODEL (MEMM)

- CMM: Attempt to turn the HMM into a discriminative model by conditioning the label on the observation **o**, which subsumes current and surrounding words, as well as features computed on them.

$$L_{\max}^{CMM} \approx \underset{I^1,\ldots I^T}{\operatorname{argmax}} \prod_{t=1}^{T} P(I^t \mid I^{t-n+1}\ldots I^{t-1}, o^t)$$

evidence →
cause →

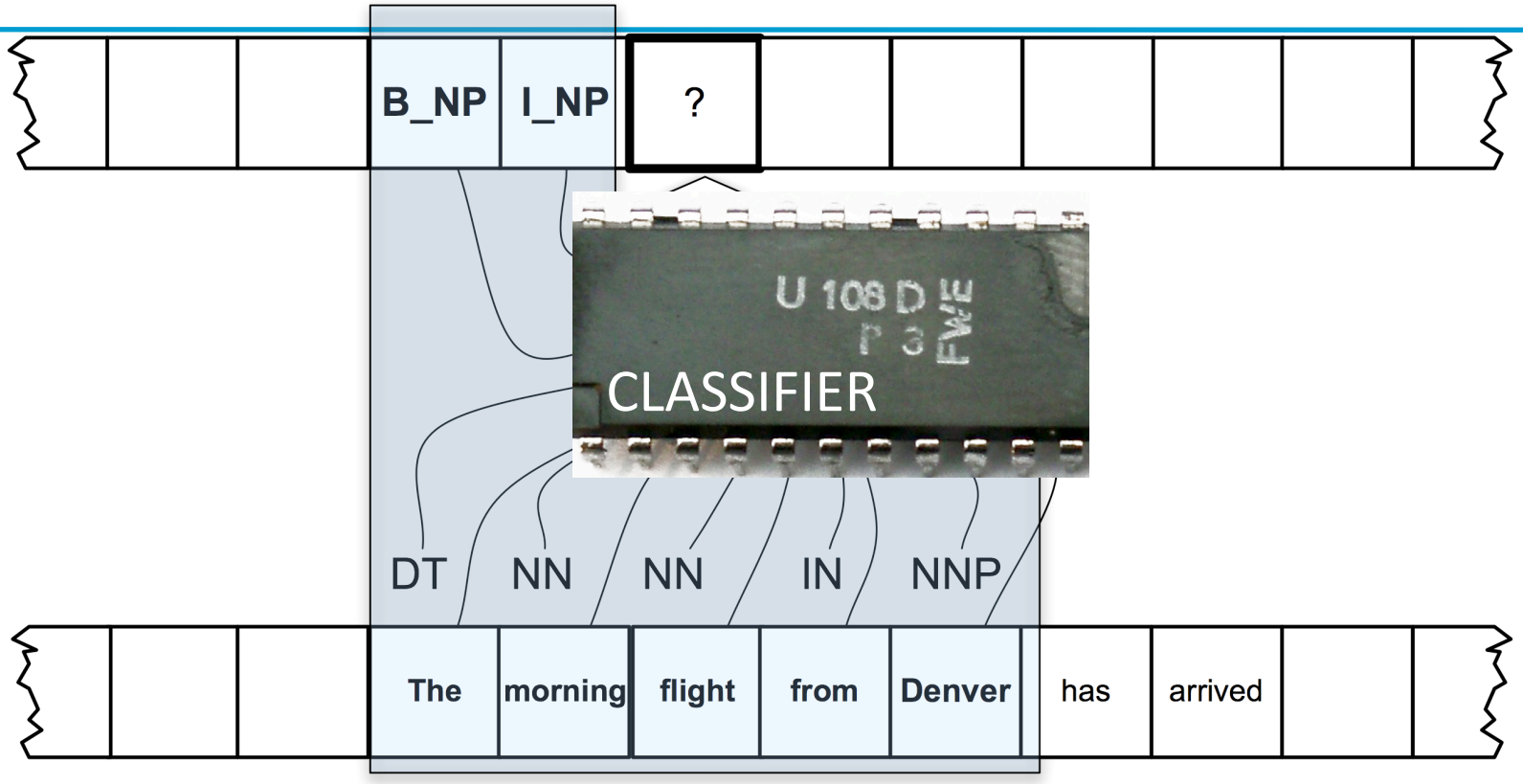


- MEMM: use a MaxEnt classifier (Logistic regression) on current and previous observations

$$L_{\max}^{MEMM} \approx \underset{I^1,\ldots I^T}{\operatorname{argmax}} \prod_{t=1}^{T} P(I^t \mid I^{t-1}, o^t)$$

$$= \underset{I^1,\ldots I^T}{\operatorname{argmax}} \prod_{t=1}^{T} \frac{\exp(\mathbf{w}^{\mathsf{T}}\mathbf{f}(I^t, I^{t-1}, o^t))}{Z(I^{t-1}, o^t)}$$

SEQUENCE CLASSIFICATION WITH MEMMS



Corresponding feature representation

Label

`The, DT, B_NP, morning, NN, I_NP, flight, NN, from, IN, Denver, NNP, I_NP` `I_NP`

HOW TO READ THE MEMM DEFINITION

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

$$L_{\max}^{MEMM} \approx \underset{I^1,\ldots I^T}{\operatorname{argmax}} \prod_{t=1}^{T} \frac{\exp(\mathbf{w}^{\mathbf{T}}\mathbf{f}(I^t, I^{t-1}, o^t))}{Z(I^{t-1}, o^t)}$$

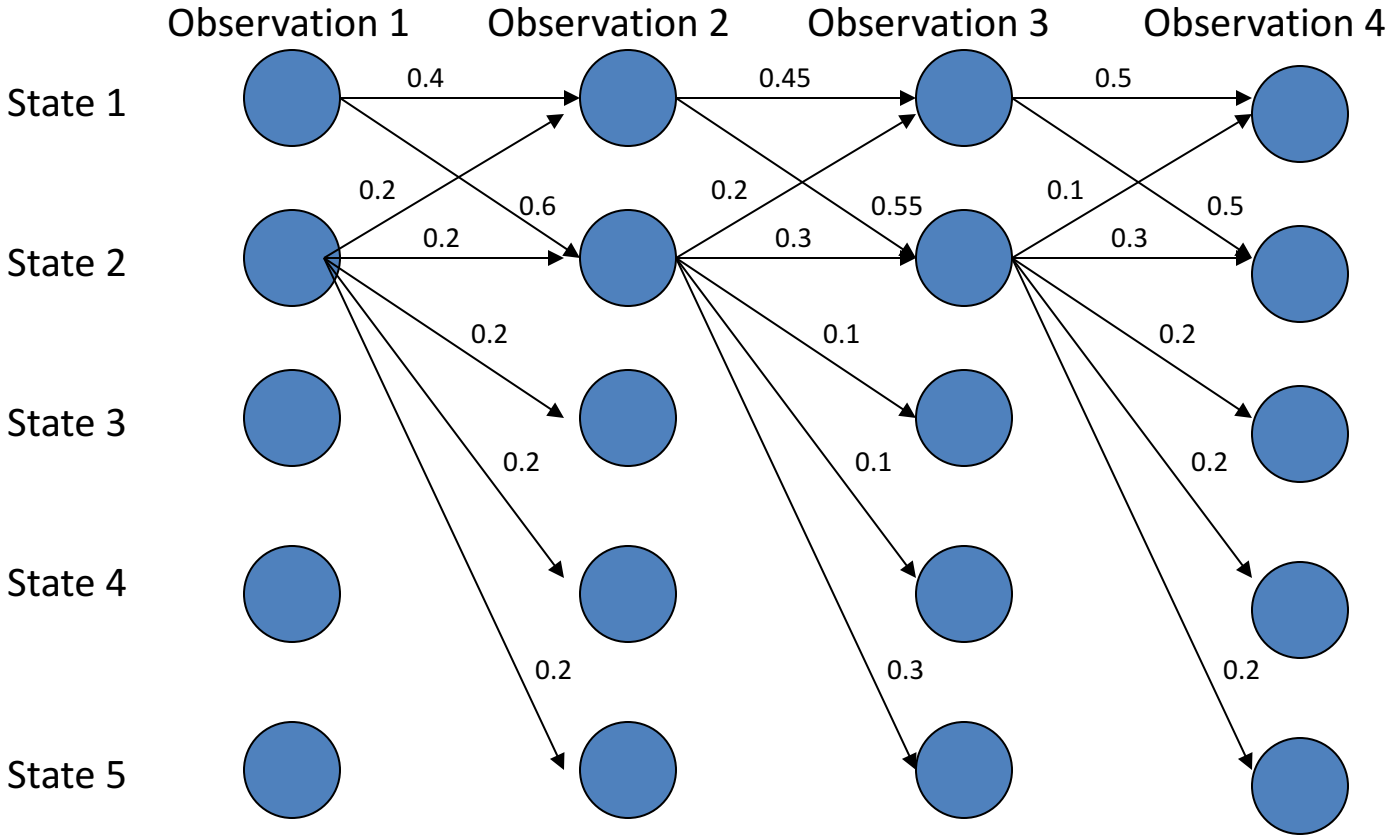
$$\exp(\mathbf{w}^{\mathbf{T}}\mathbf{f}(I^t, I^{t-1}, o^t)) = \exp(\sum_{i=1}^{k} \lambda_i \cdot f_i(I^t, I^{t-1}, o^t))$$

- $\mathbf{w}^{\mathbf{T}}$: transposed vector of feature weights λ_i
- **f**: feature vector
- f_i: function that computes feature *i* from observations and new label
- o^t: observation at time t
- Z: normalization factor
- λ_i: parameters for lin. combination
- Σ: this is a linear combination of features, weighted by λ_i
- exp(): this linear combination is performed in exponential space

The MEMM is normalized *per state*.

PROBLEM WITH MEMMS: LABEL BIAS

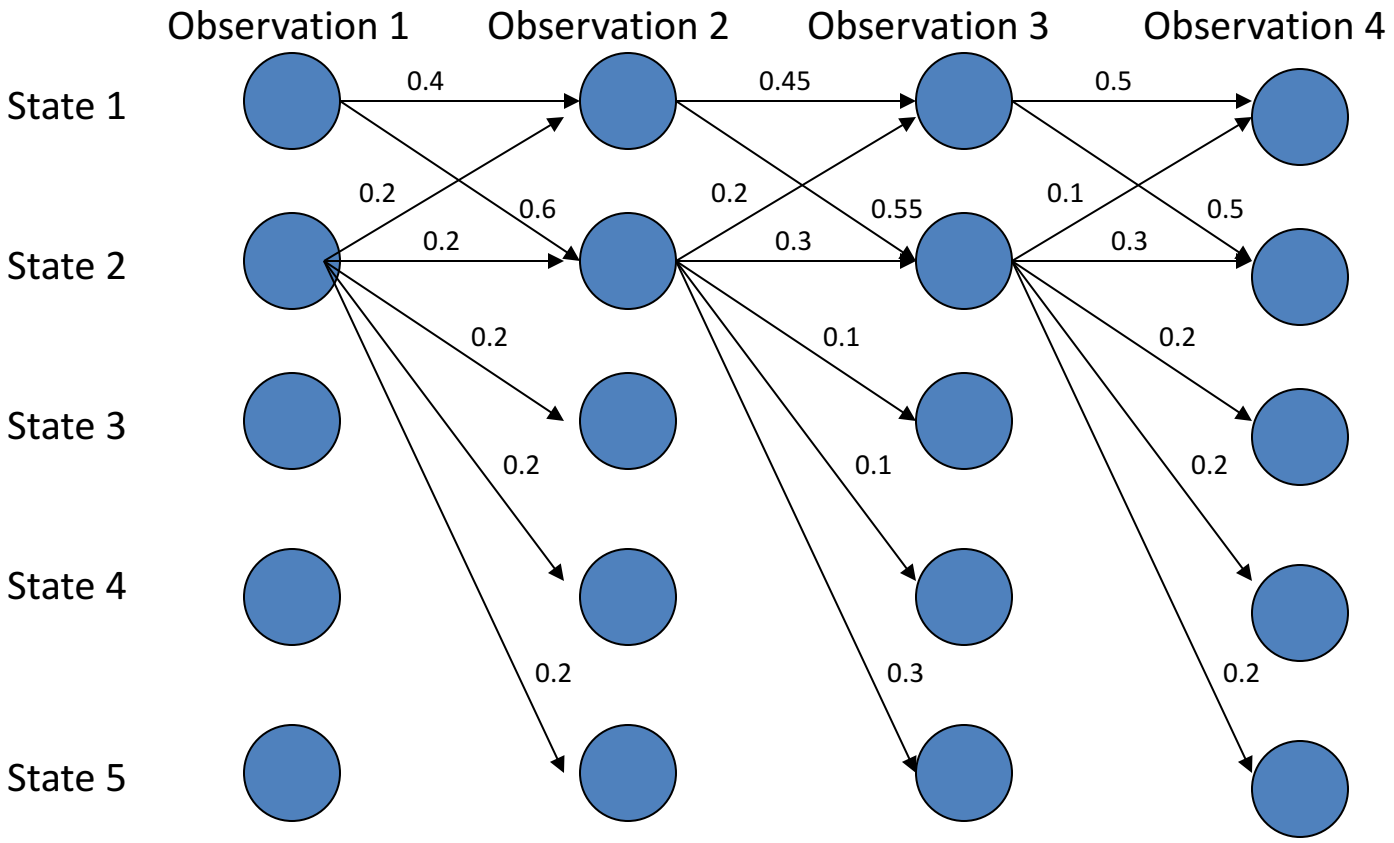
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



What the local transition probabilities say:

- State 1 almost always prefers to go to state 2
- State 2 almost always prefers to stay in state 2

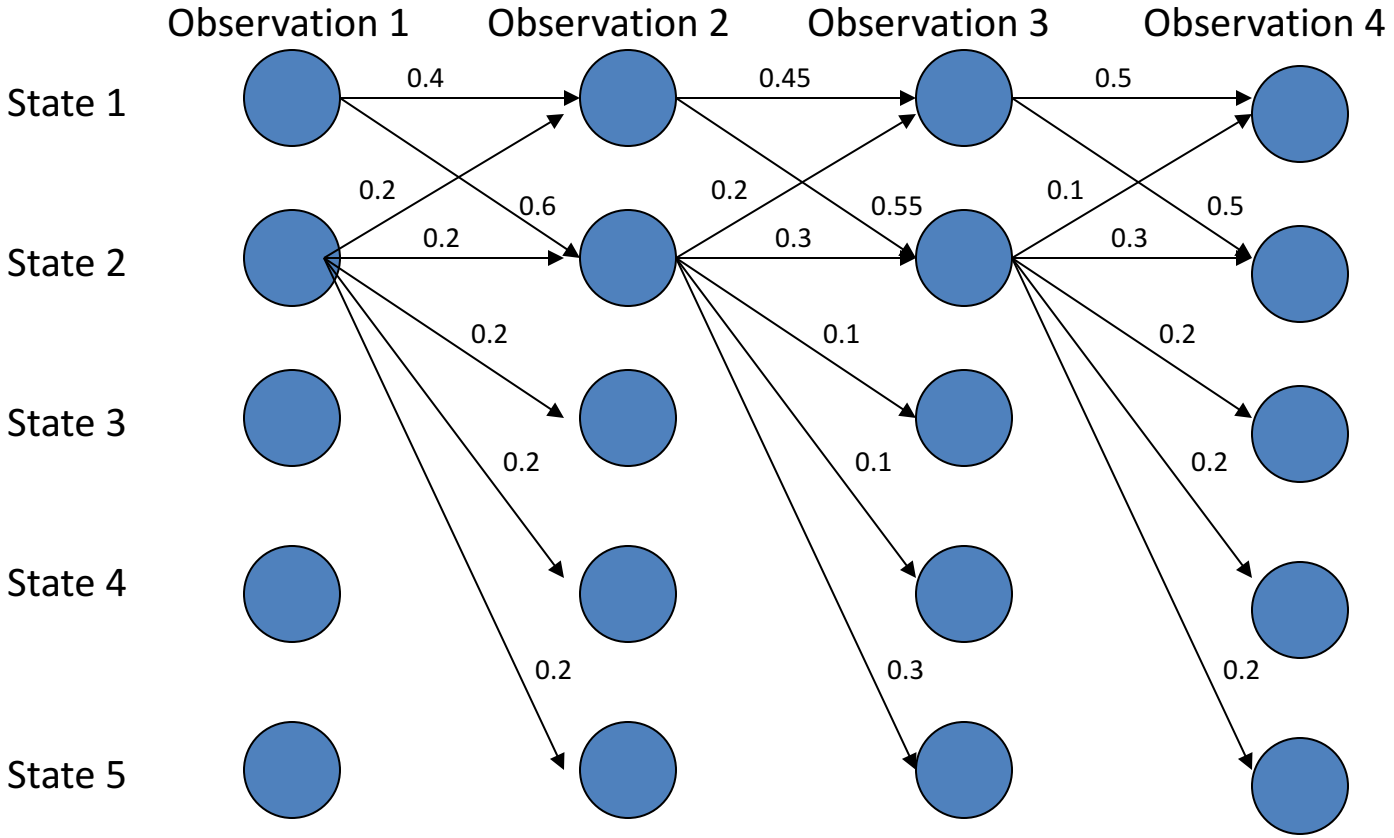
LABEL BIAS PROBLEM



Probability of path 1-> 1-> 1-> 1:

- 0.4 x 0.45 x 0.5 = 0.09

LABEL BIAS PROBLEM



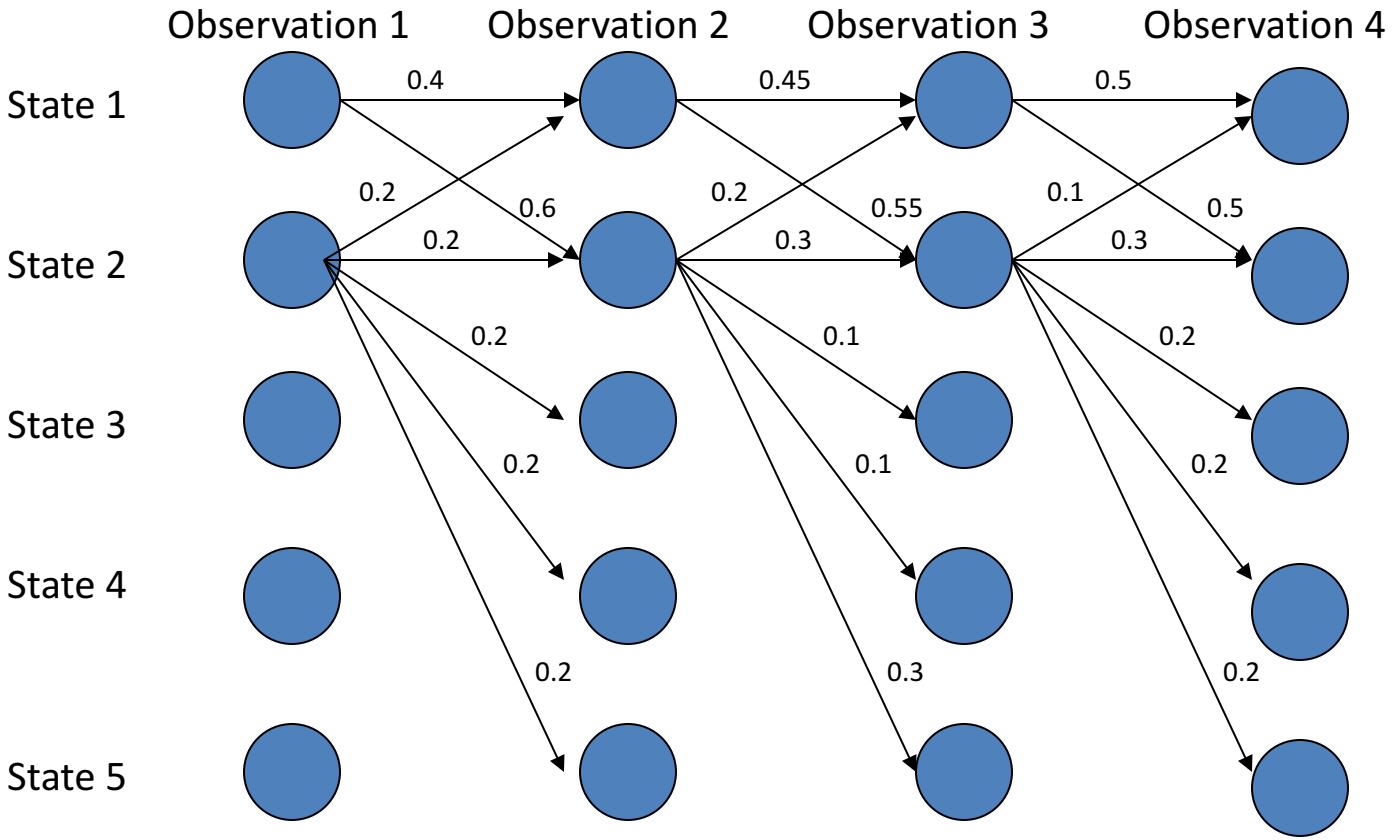
Probability of path 2-> 2-> 2-> 2:

- 0.2 x 0.3 x 0.3 = 0.018

Other paths:

- 1->1->1->1: 0.09

LABEL BIAS PROBLEM



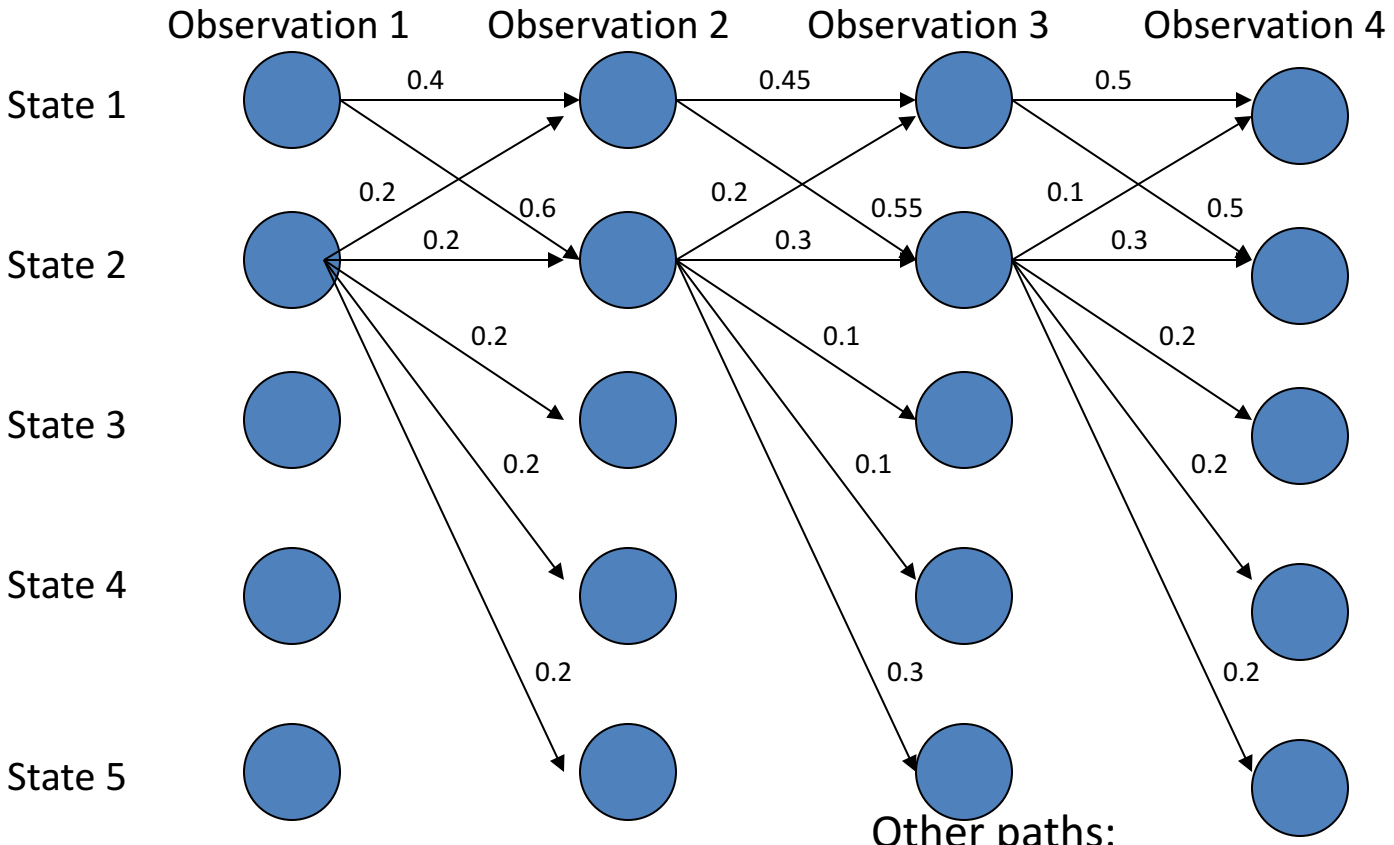
Probability of path 1-> 2-> 1-> 2:

- 0.6 x 0.2 x 0.5 = 0.06

Other paths:

- 1->1->1->1: 0.09
- 2->2->2->2: 0.018

LABEL BIAS PROBLEM



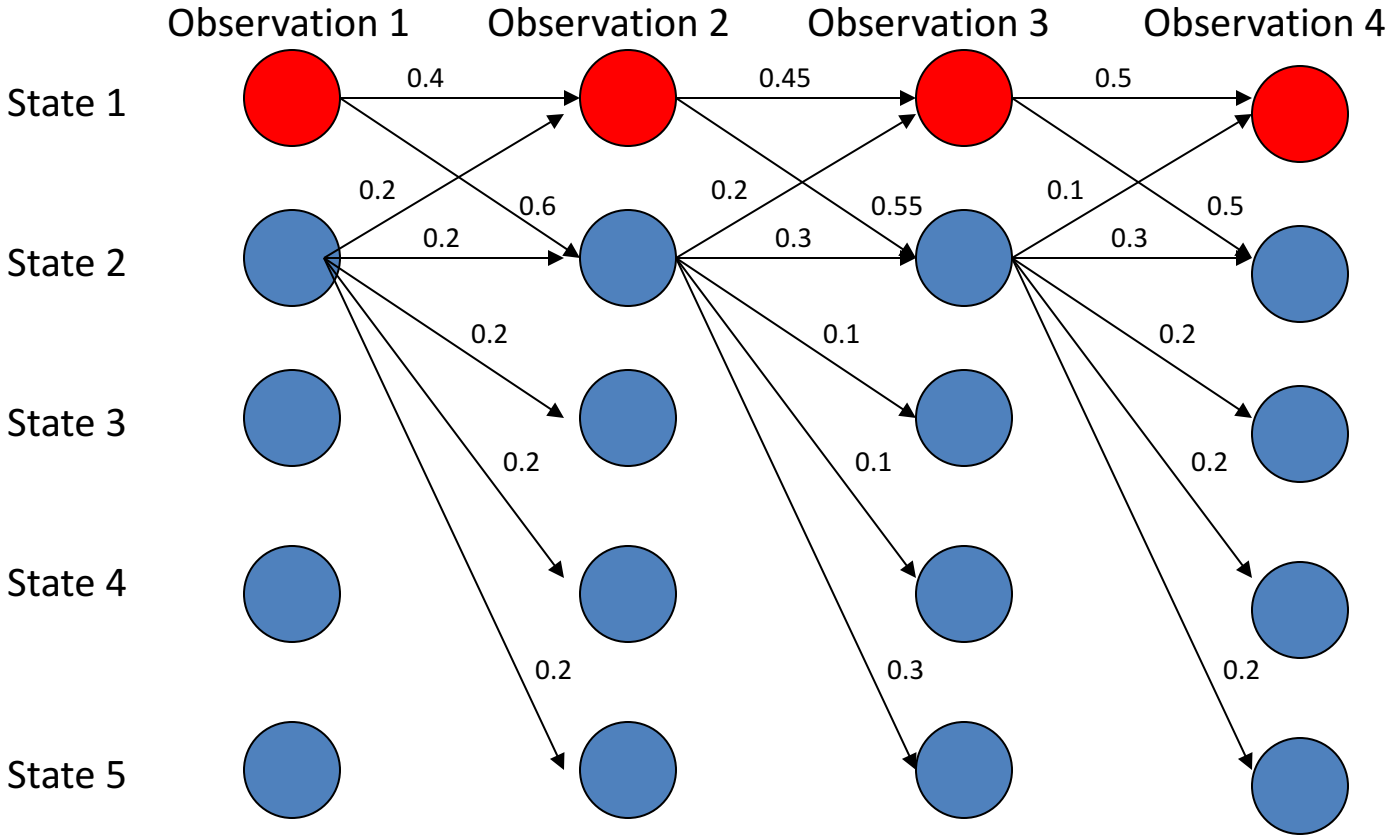
Probability of path 1-> 1-> 2-> 2:

- 0.4 x 0.55 x 0.3 = 0.066

Other paths:

- 1->1->1->1: 0.09
- 2->2->2->2: 0.018
- 1->2->1->2: 0.06

LABEL BIAS PROBLEM

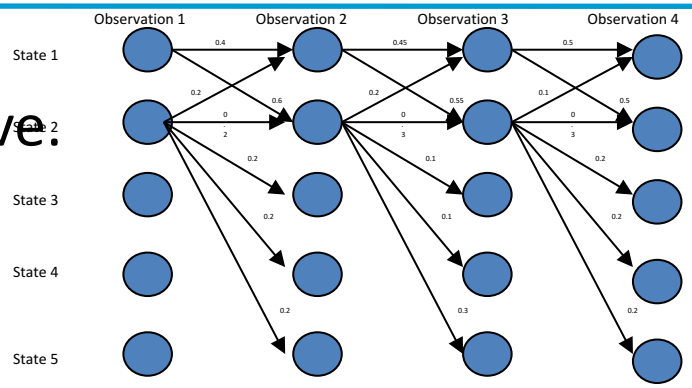


Most likely path 1-> 1-> 1-> 1!

Although it seems that state 1 wants to go to state 2 and state 2 wants to stay in state 2. Why?

REASONS FOR THE LABEL BIAS PROBLEM

- In the example: state 1 has only two possible successors, state 2 has five.
- This means: On average, state 2 has a lower average transition probability



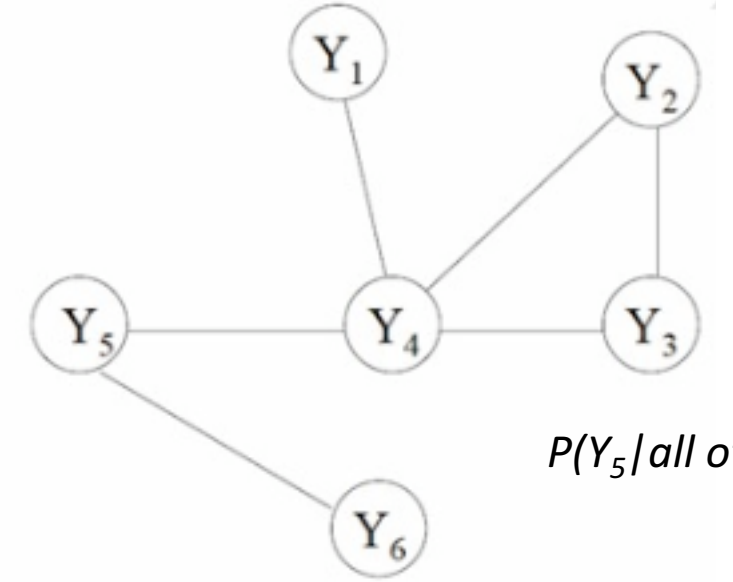
- Label bias problem: States with a lower entropy are preferred
- This results from the per-state normalization in MEMMs as well as from the independence of labels from subsequent labels. Example: unknown word at the beginning
- In theory, a dysfunctional model can stumble from state to state with high conditional probability but completely ignorant of the observation sequence…

RANDOM FIELDS: MUTUAL DEPENDENCE IN UNDIRECTED GRAPHS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Let *G(V,E)* be a graph where each vertex Y_v is a random variable. Suppose $P(Y_v|all\ other\ Y) = P(Y_v|\ neighbors(Y_v)\ )$, then G is a **random field**.

Example



$P(Y_5|all\ other\ Y) = P(Y_5|Y_4,Y_6\ )$

Random fields use *global normalization*.

CONDITIONAL RANDOM FIELD

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Let *X* be a random variable over sequences of observations to be labeled. Let *Y* be a random variable over corresponding label sequences.

Let *G=(V,E) be a graph such that* $Y=(Y_v)$, $v \in V$, so that *Y* is indexed by the vertices of G. Then *(X,Y)* is a **conditional random field** in case, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: Y_v are only dependent on their neighbors in the graph and the corresponding X.

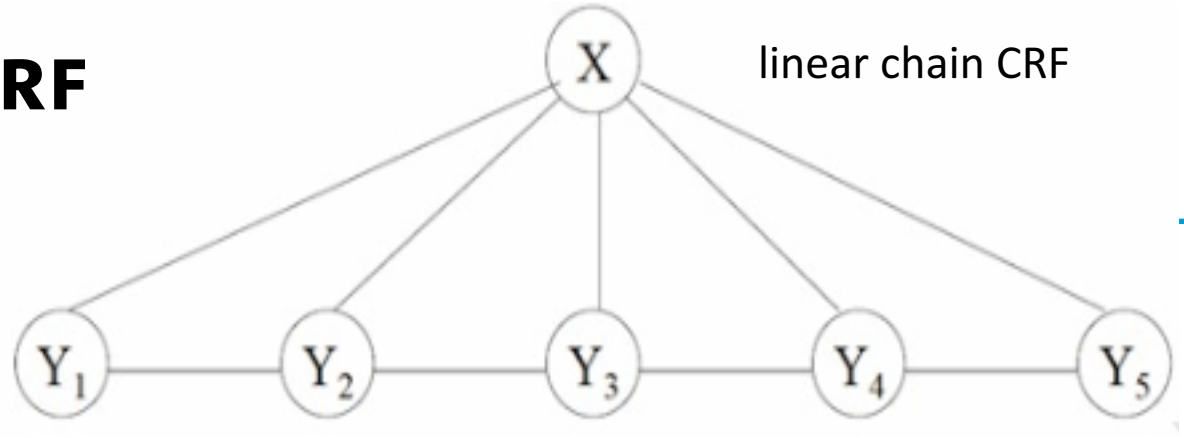
$$L_{\max}^{CRF} \approx \underset{I^1,\ldots I^T}{\operatorname{argmax}} \frac{1}{Z(o^1..o^T)} \prod_{t=1}^{T} \exp(\mathbf{w}^{\mathsf{T}}\mathbf{f}(I^t, I^{t-1}, o^t)) = \underset{I^1,\ldots I^T}{\operatorname{argmax}} \exp(\sum_{t=1}^{T}\sum_{i=1}^{k} \lambda_i f_i(I^t, I^{t-1}, o^t))$$

PROPERTIES OF CRF

$$L_{\max}^{CRF} \approx \underset{l^1,\ldots l^T}{\operatorname{argmax}} \frac{1}{Z(o^1..o^T)} \prod_{t=1}^{T} \exp(\mathbf{w}^\intercal \mathbf{f}(l^t, l^{t-1}, o^t)) = \underset{l^1,\ldots l^T}{\operatorname{argmax}} \exp(\sum_{t=1}^{T}\sum_{i=1}^{k} \lambda_i f_i(l^t, l^{t-1}, o^t))$$

- Idea: Allow some transitions to vote more strongly than others, depending on the observations
- CRF solves the label bias problem by **normalizing over the whole observation sequence**
- like MEMM, it is a **discriminative exponential** model
- the marginal probability of the observation sequence is not modeled.
- it is straightforward to implement features on the observation sequence, this includes modeling of dependencies on previous and future observations

LINEAR CHAIN CRF



linear chain CRF

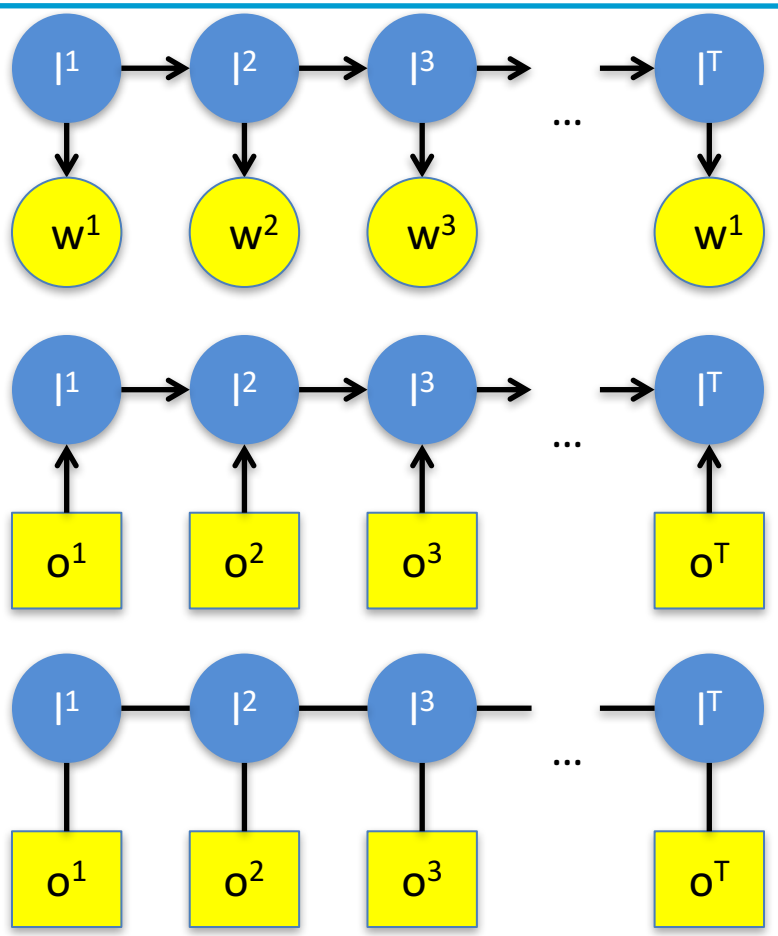
- Simplest form of CRF: every hidden state has two neighbors
- Markov property in linear chain CRFs: use a variant of Viterbi decoding for computing the optimal label sequence, conditioned on the observed features. This makes efficient decoding possible
- Linear chain CRF subsumes HMM but is more expressive, since it allows arbitrary dependencies on the observation sequence

$$P(\mathbf{L} \mid o^1 \ldots o^T) \propto \exp\left(\sum_{e \in E,k} \lambda_k f_k(e, \mathbf{L}|_e, o^1 \ldots o^T) + \sum_{v \in V,k} \mu_k g_k(v, \mathbf{L}|_v, o^1 \ldots o^T) \right)$$

$|_S$ is the operator that returns the set of components associated with S.

f and *g* are Boolean feature functions, λ and μ are model parameters playing a similar role as $p(l^t \mid l^{t-1})$ and $p(w^t \mid l^t)$ in the HMM.

GRAPHICAL REPRESENTATIONS: HMM, MEMM AND CRF



- HMM:
  - previous tag labels generate next tag label
  - tag label generates word
- CMM/MEMM
  - next tag label is dependent on previous tag labels and current observation
  - observation: features on words
- linear chain CRF
  - mutual dependence between labels and observations
  - observation: features on words

TRAINING CRFS IS HARD AND SLOW

- Mutual dependence between label random variables does not allow simple counting in MLE style
- Baum-Welch (as in HMMs) did not take features on observations into account and is not defined for undirected relations

Several methods for training CRFs:

- iterative scaling (slow)
- gradient descent (accelerated)
- Conjugate Gradient method
- Limited Memory Quasi-Newton Methods

EVALUATING THE CRF WITH ARTIFICIAL DATA: THE LABEL BIAS PROBLEM

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Training data: Output of simple HMM. Each state generates its designated symbol with probability 29/32 and three other symbols with prob. 1/32 each
- Experiment: Train MEMM and CRF with the same topologies and reconstruct generating state sequence
- A run consists of 2'000 training examples and 500 test examples, trained to convergence using iterative scaling
- CRF error: 4.6%, MEMM error: 42%

➔ MEMM fails to discriminate between branches, CRF solves label bias problem

EVALUATION ON POS TAGGING

model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

- First order models (bigrams), 45-tagset Penn Treebank, 50% train/test
- With no additional features, HMM and CRF are about equal, and much better than MEMM
- Using additional features, CRF is much better than MEMM.

NO LIMITED HORIZON ASSUMPTION

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

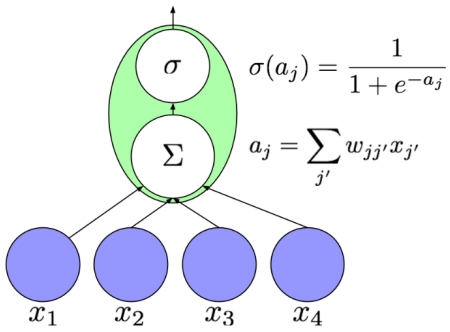
Reason for limited horizon: efficiency! Need a reasonably small number of discrete states for quadratic runtime of Viterbi.

Dropping this assumption requires:

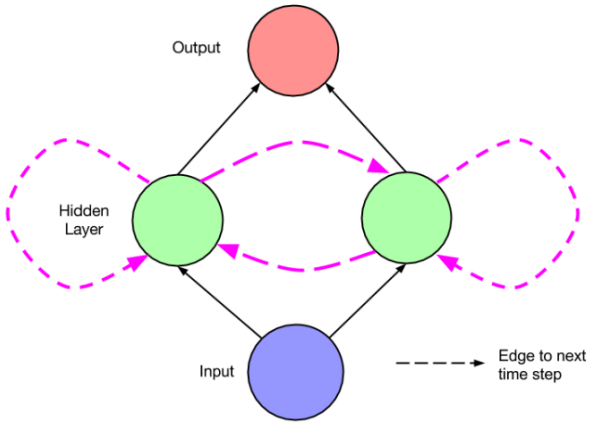
- Keeping the representation of states small:
  A binary activation layer of N neurons can represent 2^N discrete states, continuous activations even more

- Recurrent units:
  State is stored in a context history, which can retain traces of the entire input

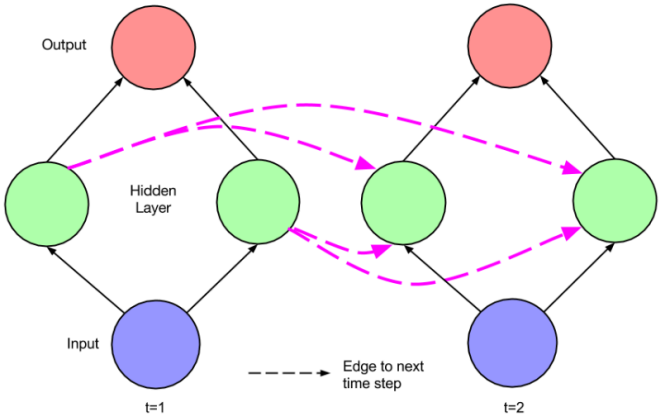
RECURRENT NEURAL NETWORKS



Artificial neuron



Simple recurrent network with 2 hidden units



Same recurrent network unfolded over time

- Recurrent connections: input at time step comes from activation at time step (t-1)
- Recurrent connections introduce notion of sequence into the network
- Unfolding: Can view recurrent network like a deep network, can apply gradient-based training

GRADIENT-BASED TRAINING: BACKPROPAGATION

- Remember stochastic gradient-based training from neural LM:

$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log \hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1})}{\partial \theta}$$

- Backpropagation:
  1. Propagate new training example through the network to get activations v_i at each node and an output value $\hat{y}$ at the topmost layer.
  2. Compute update towards output node using loss function

$$\delta_k = \frac{\partial \mathcal{L}(\hat{y}_k, y_k)}{\partial \hat{y}_k} \cdot l'_k(a_k)$$

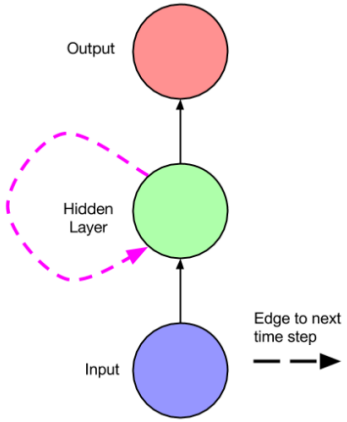
  3. Propagate loss/error through the network layers

$$\delta_j = l'(a_j) \sum_k \delta_k \cdot w_{kj}$$

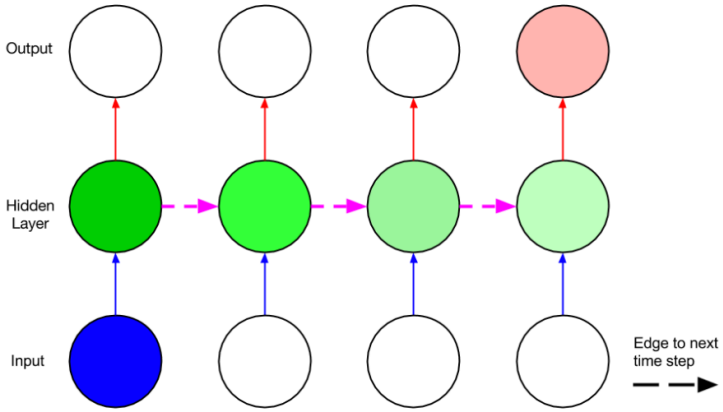
  4. Change weights according to learning rate

With minor adaption, can use backpropagation for training recurrent networks

VANISHING / EXPLODING GRADIENTS



Simple recurrent network



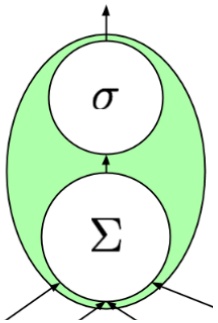
Unfolded network, visualizing the vanishing gradient

- Gradients vanish (explode) exponentially across time steps when the recurrent connection is <1 (>1)
- Problem is connected to the fact that it is always THE SAME connection weight

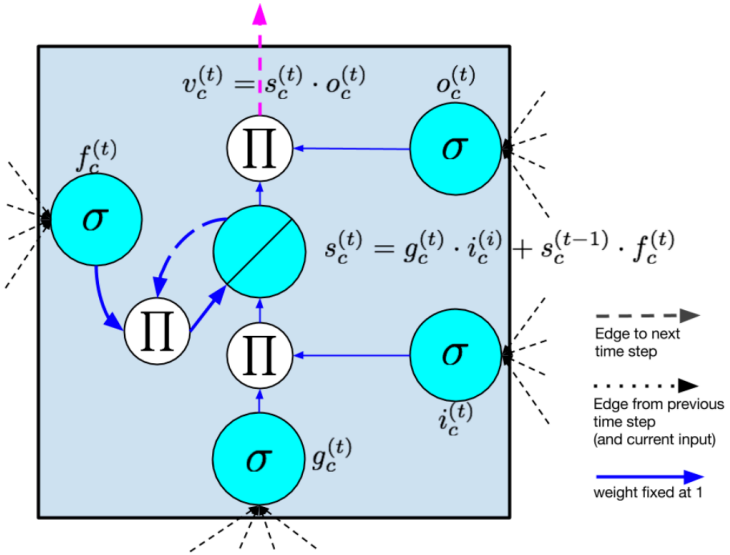
Alex Graves. Supervised sequence labelling with recurrent neural networks, Volume 385. Springer, 2012.

LONG SHORT-TERM MEMORY (LSTM)

- LSTM resembles a standard recurrent neural network with a hidden layer
- Nodes in the hidden layer are replaced by a memory cell
- Memory cells contain a node with a self-connected recurrent edge of fixed weight 1 (no gradient issues)

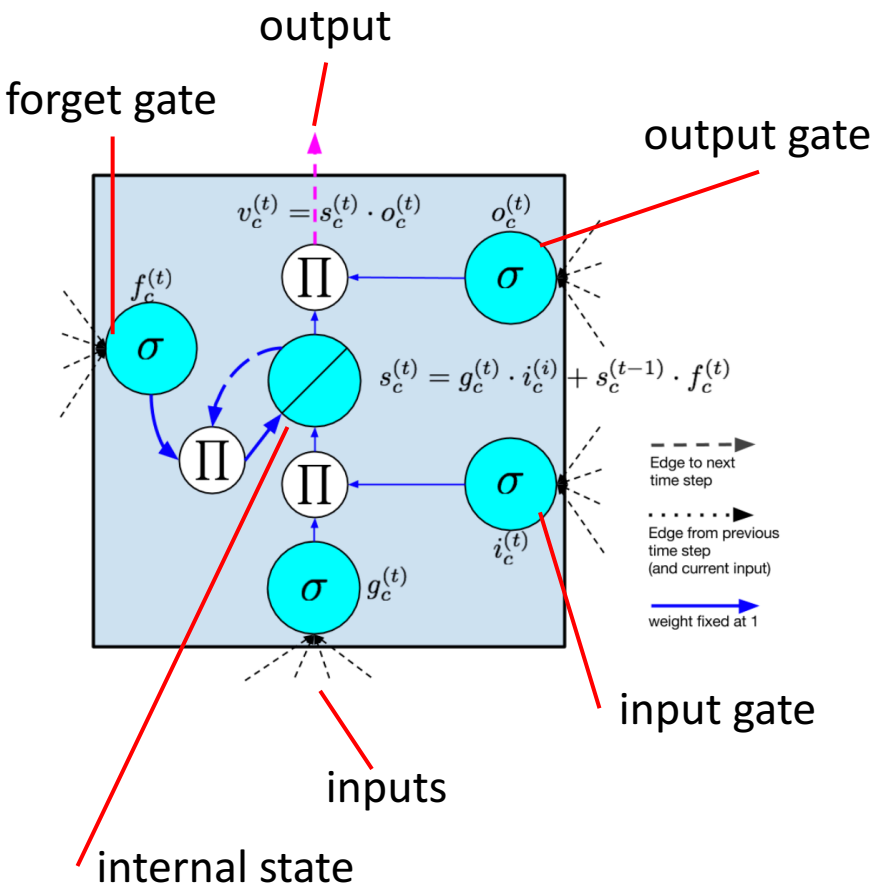


vs.



Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

MEMORY CELL IN LSTM



- inputs: from sequence and from other memory cells
- input gate: regulates whether to take input into account
- output gate: regulates whether to output the internal state
- forget gate: can flush internal state
- recurrent link with weight 1: “constant error carousel”.

LSTM INTUITIONS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- "long short-term memory": standard NNs have
  - long-term memory in the weights
  - short-term memory in the activations

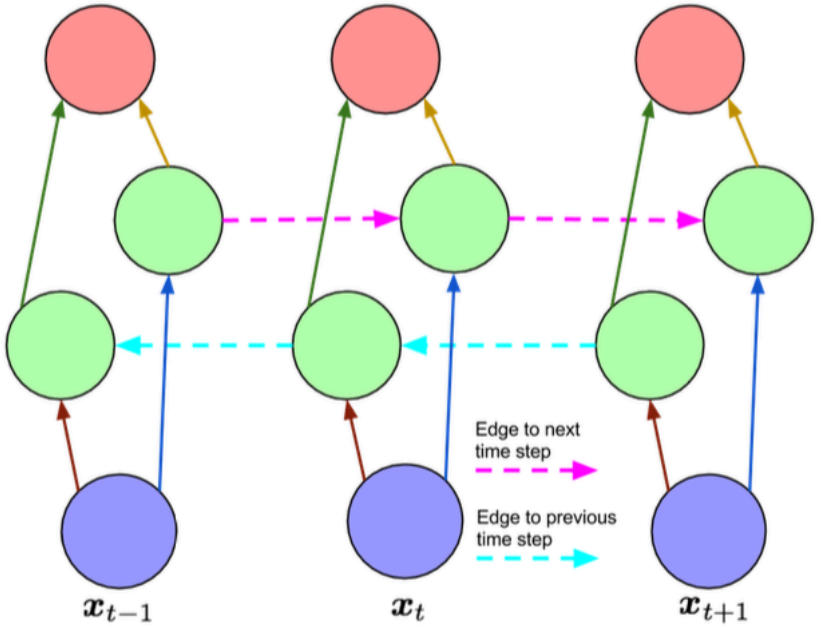
  LSTM mixes both notions

- Gate: pointwise multiplication regulates how much is passed through, based on inputs
- Internal state serves as a memory
- Recurrent connection of weight 1: error can flow across time steps without vanishing or exploding

- LSTM can learn:
  - when to let the input (and error) in — *e.g. set the new grammatical subject*
  - when to let the output (and error) out — *e.g. predict verb that takes the subject*
  - when to reset its memory — *e.g. remove old subject once its taken*

BIDIRECTIONAL LSTMS

- Idea: if we are tagging whole sentences, we can use context representations from the 'past' and from the 'future' to predict the 'current' label
- Not applicable in an online incremental setting.
- LSTM cells and bidirectional networks can be combined into Bi-LSTMs



Bidirectional recurrent network, unfolded in time

BI-LSTM FOR SEQUENCE TAGGING

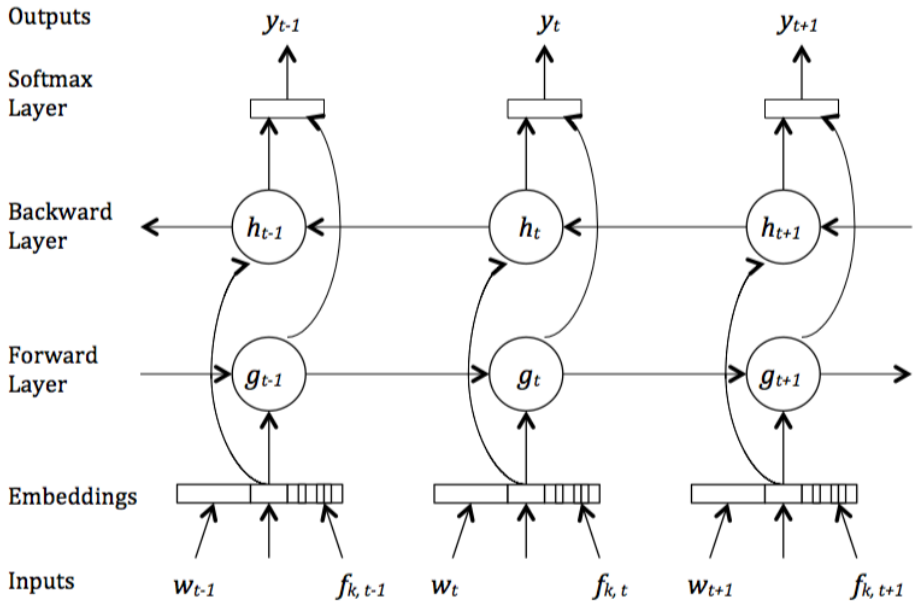


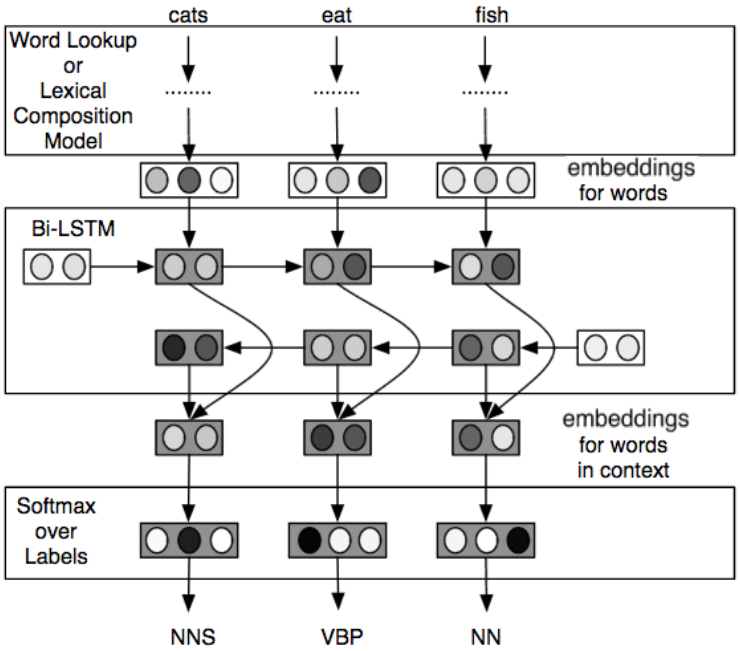
Fig. from: Zayats, V., Ostendorf, M., Hajishirzi, H. (2016): Disfluency Detection using a Bidirectional LSTM. Proceedings of Interspeech 2016

- Input: Word embeddings, additional word features
- Combine two directions: usually concatenation
- Output: 1-hot-encoding over labels (softmax)

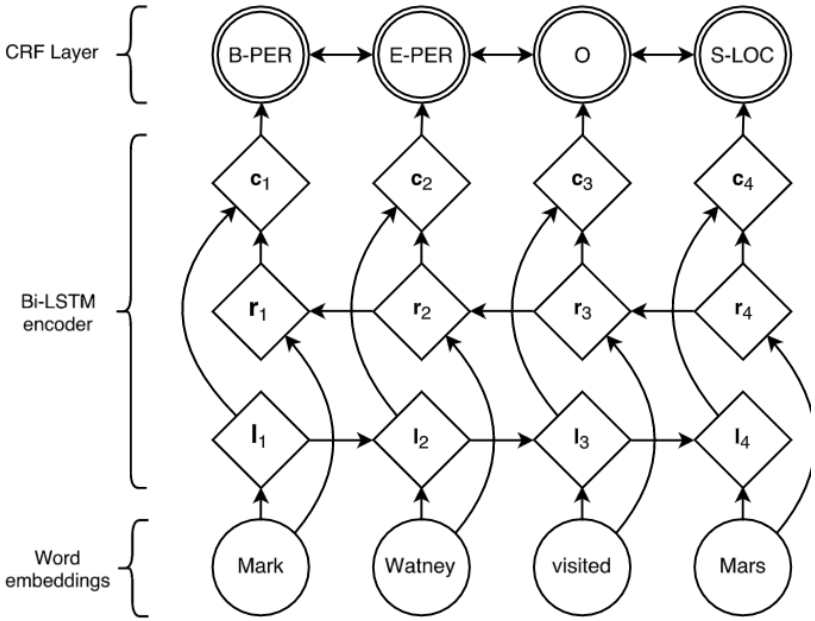
Note:

- State size: there are many 'parallel' LSTM cells in each layer
- LSTM layers can be stacked for deeper networks

BI-LSTM FOR POS TAGGING - VARIANTS



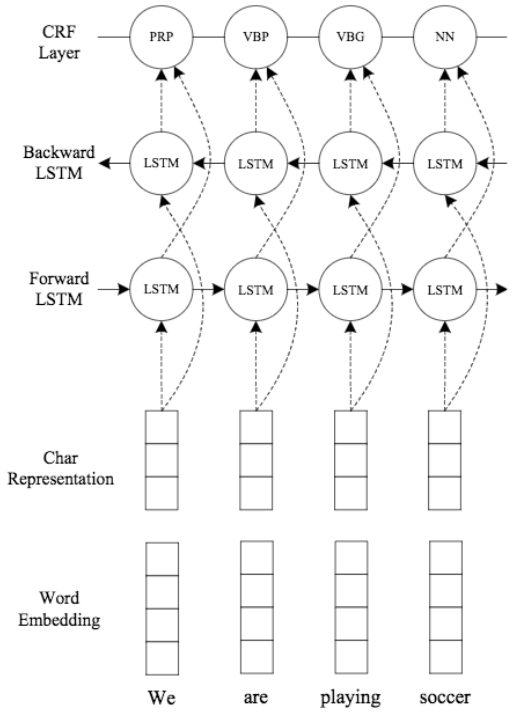
- Compose words from character embeddings to address unseen words



- Use combined outputs as features in CRF layer, better making use of neighboring labels

Ling, W., Dyer, C., Black, A.W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L. and Luis, T. (2015): Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation, Proceedings of EMNLP, pp. 1520-1530, Lisbon, Portugal.
Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016): Neural Architectures for Named Entity Recognition. Proceedings of NAACL, San Diego, CA, USA.

2016 STATE OF THE ART IN POS TAGGING AND NAMED ENTITY RECOGNITION



Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

Table 1: Hyper-parameters for all experiments.

Further parameters:

- which kind of word embeddings
- early stopping after 50 epochs

Model	Acc.
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

Model	F1
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)‡	89.59
Huang et al. (2015)‡	90.10
Chiu and Nichols (2015)‡	90.77
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Passos et al. (2014)	90.90
Lample et al. (2016)‡	90.94
Luo et al. (2015)	91.20
This paper	**91.21**

‡ marks the neural models

One of the first papers that has state-of-the-art performance with end-to-end approach on standard text processing

Ma, X. and Hovy, E. (2016): End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. Proceedings of ACL 2016, pp. 1064-1074, Berlin, Germany

SUMMARY ON STATISTICAL SEQUENCE TAGGING

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- HMM is the classic generative model used for sequence tagging
  - tag labels are modeled as hidden states
  - hidden states are observable for labeled training text
  - when applying to unlabeled text, max path through hidden state sequence is the tag sequence
- CMMs are an alternative that make it easier to incorporate arbitrary features on the observations
  - discriminative model: can use any classifier, e.g. MaxEnt, SVM, NN etc.
  - per-state normalization leads to the label bias problem
- CRF subsumes the advantages of HMM and CMM:
  - per sequence normalization: no label bias problem
  - discriminative: arbitrary features possible
  - training is slow
- Bi-LSTM is a recent neural approach
  - Less feature engineering: induce features and representations
  - Operates on continuous word representations
  - hyperparameter tuning very costly, makes training even slower

➔ CRFs are the framework of choice for modern sequence taggers, rivaled by Bi-LSTMs. Combination of handcrafted features with neural approaches still brings best performance.

Chunking, Syntax trees, Parsing, PCFGs

SYNTAX PROCESSING