Introduction to Natural Language Processing

Sequence Tagging

- Jurafsky, D. and Martin, J. H. (2019): Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Third Edition. Chapters 8 and 9: https://web.stanford.edu/~jurafsky/slp3/8.pdf, https://web.stanford.edu/~jurafsky/slp3/9.pdf
- 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
- Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In NAACL-HLT
- Simoes, G., Galhardas, H., & Coheur, L. (2004). Information Extraction tasks: a survey. Chicago

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Sequence tagging

- 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-PER`** `and/`**`O`** `Rose/`**`B-PER`** `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

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Parts-of-Speech (POS)

- 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. However:
  - Dionysious Thrax of Alexandria (100 B.C.) already pointed to these 8 POS tags
  - Some 2000+ years later a popular US broadcast made 8 educational videos on each

Conjunction Junction, what's your function?
Bob Dorough, *Schoolhouse Rock, 1973*

Parts-of-Speech (POS)

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

- 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

Language	Tagset Size
English	139
Czech	970
Estonian	476
Hungarian	401
Romanian	486
Slovene	1033

- 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

Penn Treebank POS tags

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	"	left quote	' or "
LS	list item marker	*1, 2, One*	TO	"to"	*to*	"	right quote	' or "
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

The key POS tagging challenge: to resolve grammatical ambiguity

Words often have more than one POS: back

- The back door = Adjective (JJ)
- On my back = Noun (NN)
- Win the voters back = Adverb (RB)
- Promised to back the bill = Verb (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? Ambiguity in the Brown corpus

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

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

This type of measurement is also applicable and useful for other sequence labelling tasks in NLP!

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Named Entity Recognition (NER)

- Find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

Named Entity Recognition (NER)

- Find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

Named Entity Recognition (NER)

Person (PER) **Location (LOC)** **Organization (ORG)** **Date**

- Find and classify names in text, for example:

  – The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

Named Entity Recognition (NER)

Person (PER) Location (LOC) Organization (ORG) Date

- **The uses**:
  - Named entities can be indexed, linked off, etc.
  - Sentiment can be attributed to companies or products
  - A lot of IE relations are associations between named entities
  - For question answering, answers are often named entities.

- **Concretely**:
  - Many web pages tag various entities, with links to bio or topic pages, Wikipedia (the entity linking task), etc.
    - Reuters' OpenCalais, AlchemyAPI, Yahoo's Term Extraction, …
  - Apple/Google/Microsoft/… smart recognizers for document content

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

Named Entity Recognition (NER)

Person (PER) **Location (LOC)** **Organization (ORG)** **Date**

Task: Predict entities in a text

Foreign	ORG
Ministry	ORG
spokesman	O
Shen	PER
Guofang	PER
told	O
Reuters	ORG
:	:

} Standard evaluation is per entity, *not* per token

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

Named Entity Recognition (NER)

Person (PER) Location (LOC) Organization (ORG) Date

BIO (a.k.a. **IOB**) notation: **B**egin (of an entity span), **I**nside (an entity span), **O**utside (an entity span)

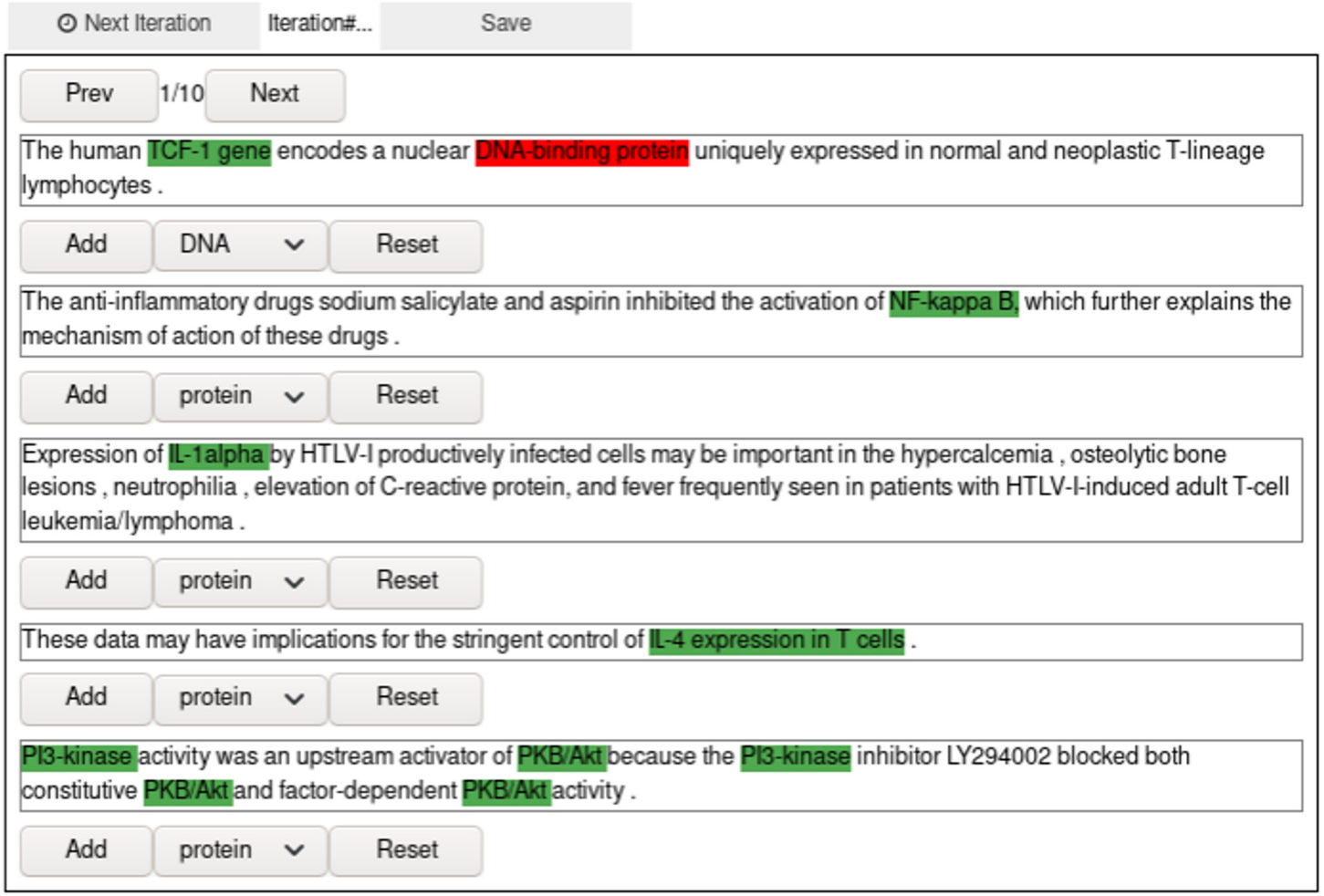
```
Foreign     B-ORG
Ministry    I-ORG
spokesman   O
Shen        B-PER
Guofang     I-PER
told        O
Reuters     B-ORG
:           :
```

Chris Manning and Daniel Jurafsky: NLP course at Stanford (2012 edition).

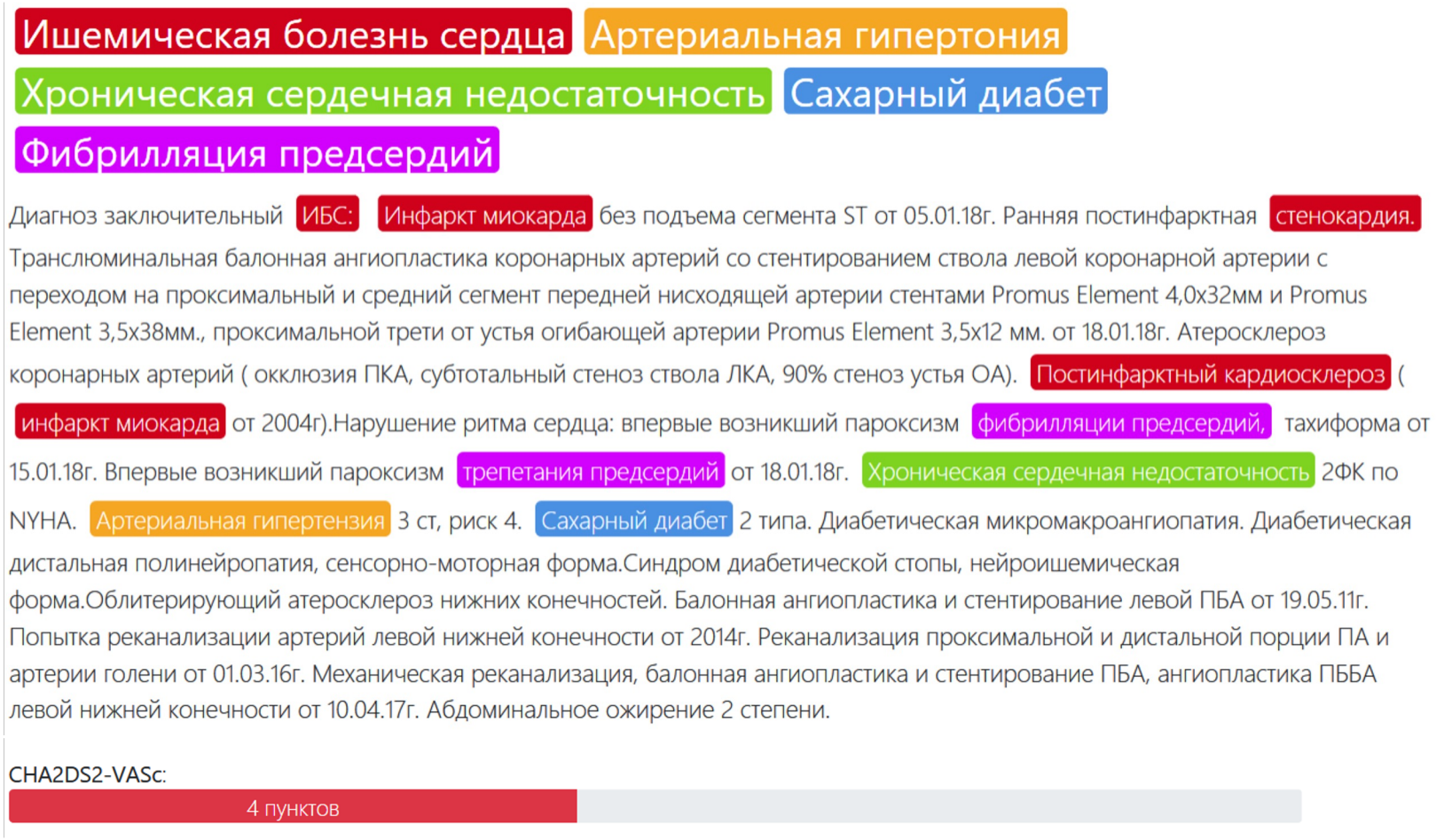
PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Other Tasks: Tagging Biomedical Entities (names of proteins, etc.)

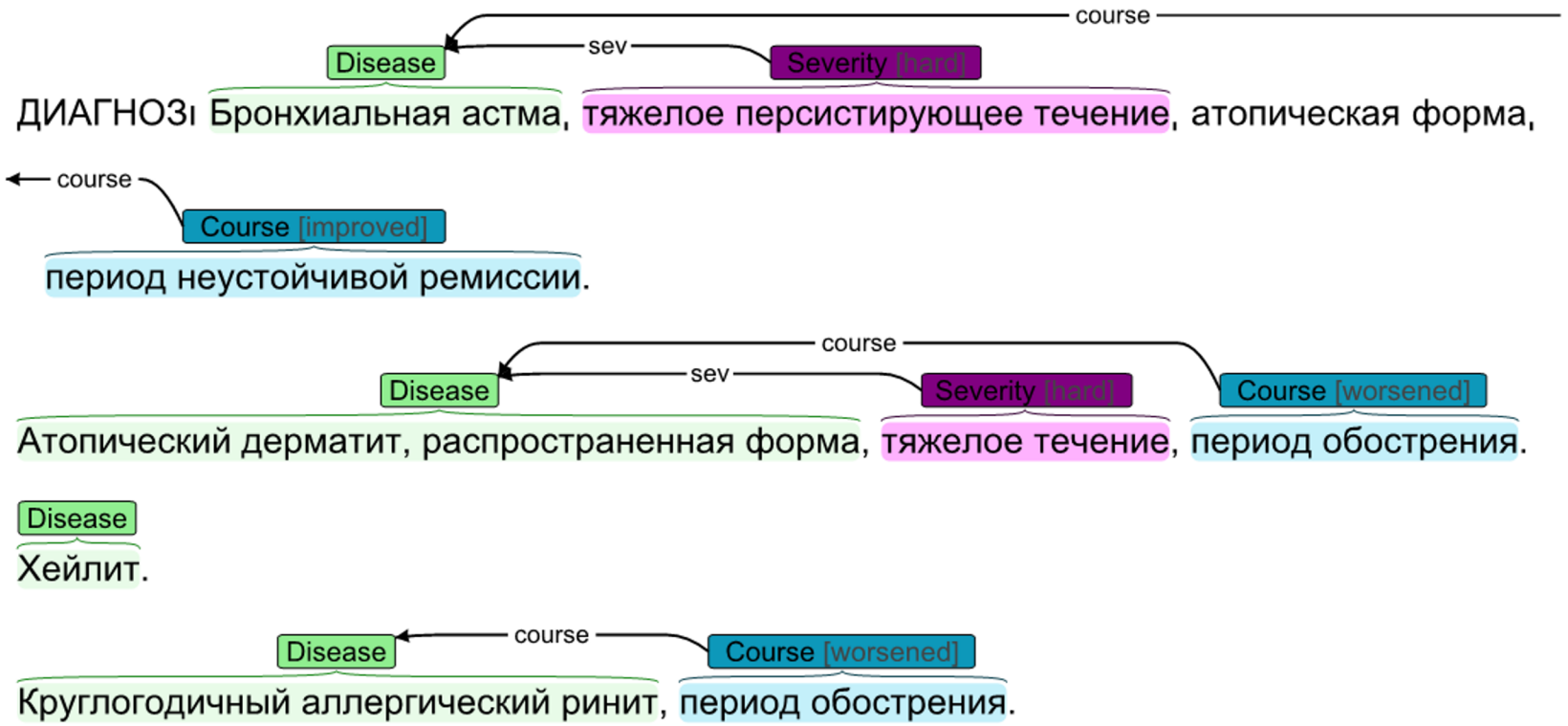


Other Tasks: Tagging Biomedical Entities (diseases, symptoms, etc.)

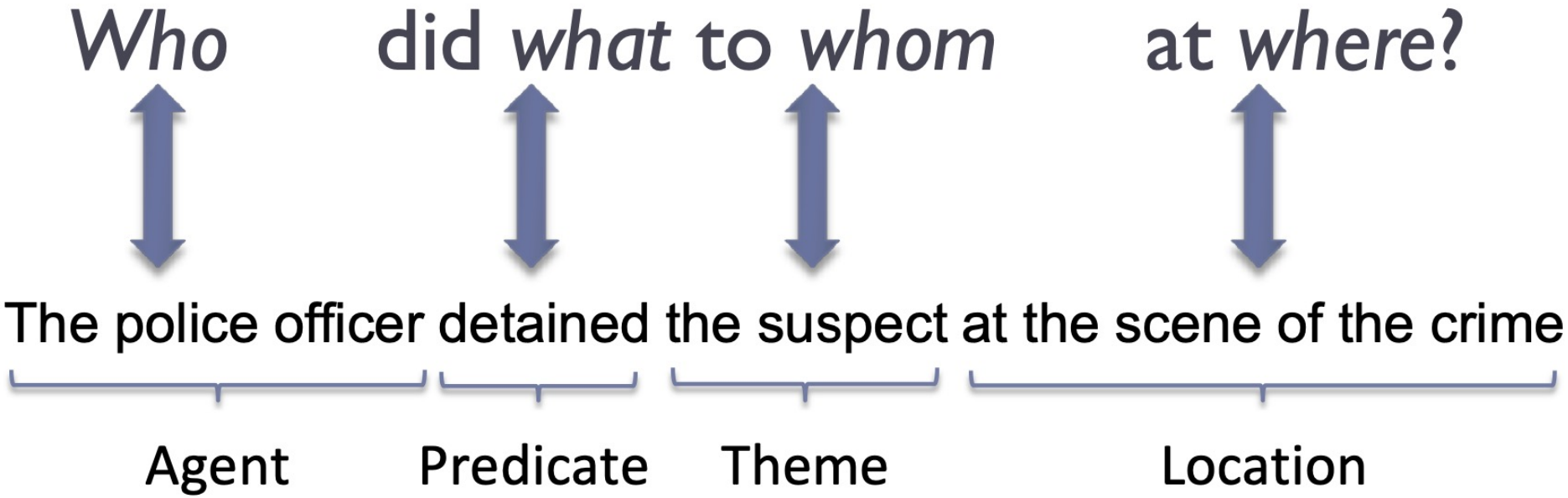


Other Tasks: Tagging Biomedical Entities (diseases, symptoms, etc.)

Entity detection can be used for detection of relations between entities → Information Extraction.



Other Tasks: Semantic Role Labeling, SRL (Thematic Roles)



Thematic Role	Definition	Example
AGENT	The volitional causer of an event	*The waiter* spilled the soup.
EXPERIENCER	The experiencer of an event	*John* has a headache.
FORCE	The non-volitional causer of the event	*The wind* blows debris from the mall into our yards.
THEME	The participant most directly affected by an event	Only after Benjamin Franklin broke *the ice*...
RESULT	The end product of an event	The city built a *regulation-size baseball diamond*...
CONTENT	The proposition or content of a propositional event	Mona asked *"You met Mary Ann at a supermarket?"*
INSTRUMENT	An instrument used in an event	He poached catfish, stunning them *with a shocking device*...
BENEFICIARY	The beneficiary of an event	Whenever Ann Callahan makes hotel reservations *for her boss*...
SOURCE	The origin of the object of a transfer event	I flew in *from Boston*.
GOAL	The destination of an object of a transfer event	I drove *to Portland*.

Other Tasks: Semantic Role Labeling, SRL (PropBank)

increase.01 "go up incrementally"

Arg0: causer of increase
Arg1: thing increasing
Arg2: amount increased by, EXT, or MNR
Arg3: start point
Arg4: end point

[$_{Arg0}$ Big Fruit Co.] increased [$_{Arg1}$ the price of bananas].
[$_{Arg1}$ The price of bananas] was increased again [$_{Arg0}$ by Big Fruit Co.]
[$_{Arg1}$ The price of bananas] increased [$_{Arg2}$ 5%].

https://web.stanford.edu/~jurafsky/slp3/slides/22_SRL.pdf

Other Tasks: Semantic Role Labeling, SRL (FrameNet)

- Semantic frames: COLLABORATION, SELF_MOTION, CONDUCT.
- Each frame has a set of specific roles, e.g. Goal, Manner, etc.

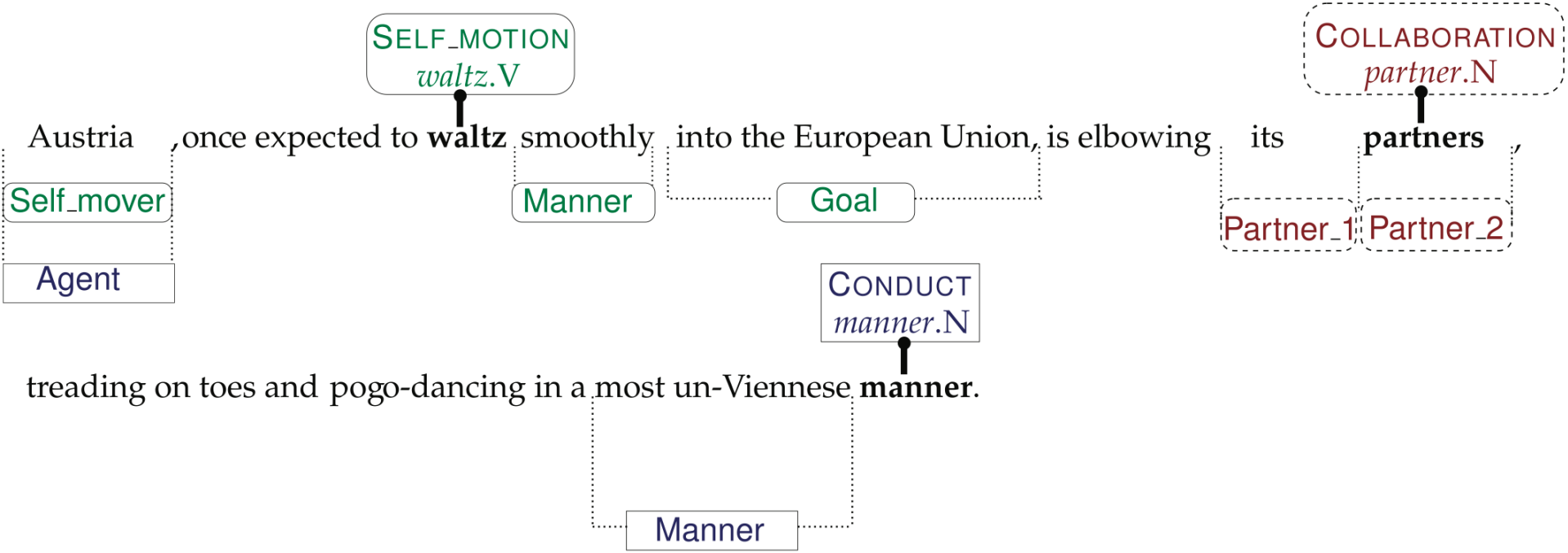
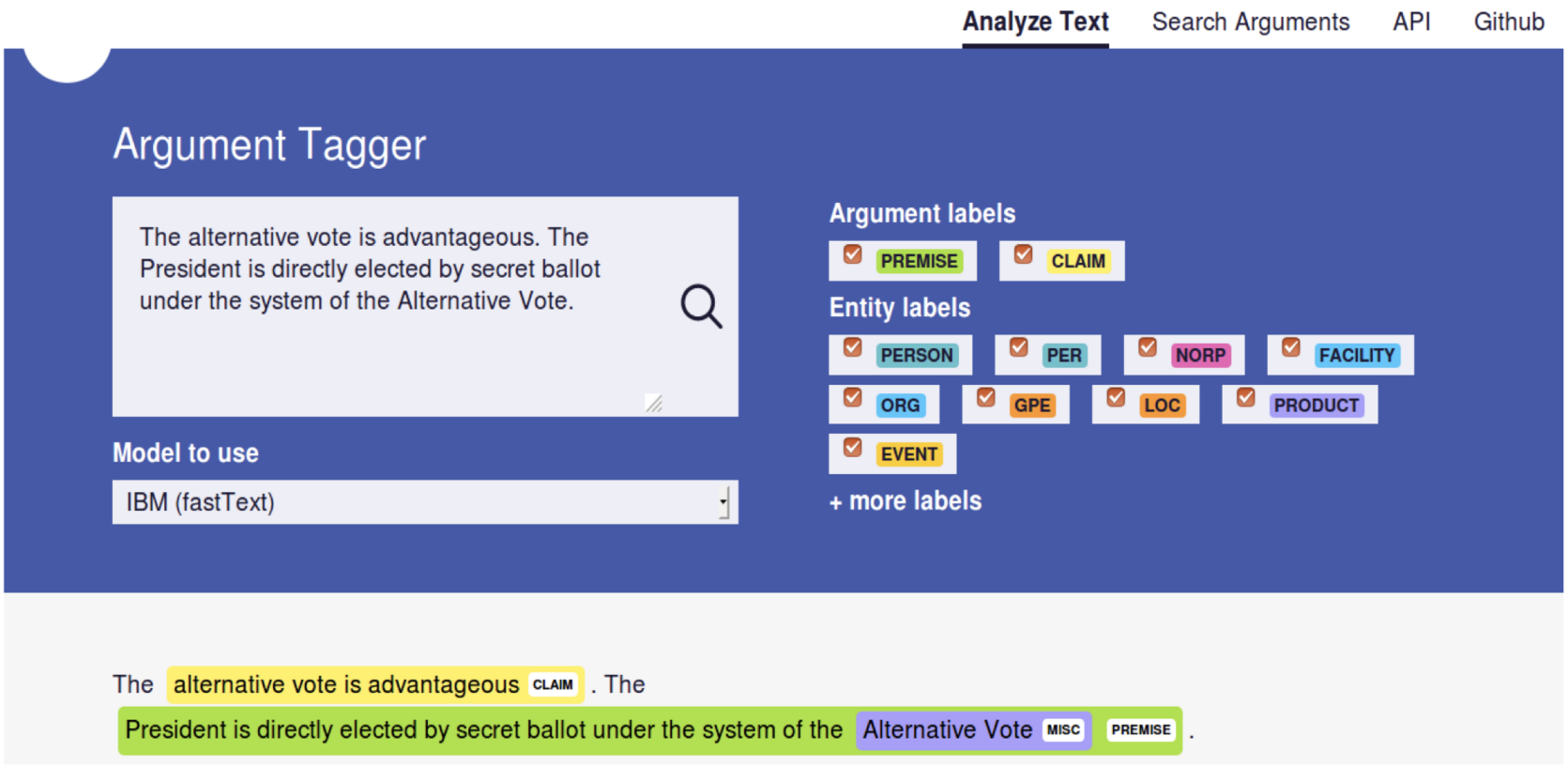


Figure from Das et al (2014)

https://web.stanford.edu/~jurafsky/slp3/slides/22_SRL.pdf

Other Tasks: Argument Mining. Tagging “Pro/Con” Discourse.

Argument mining provides algorithms for automatically extracting argument structures from text (in the simplest case the PREMISE-CLAIM BIO tagging).



http://ltdemos.informatik.uni-hamburg.de/targer/

Other Tasks: Comparative Argument Mining (a kind of SRL)

Object A	Object B	Sentence
ios	android	i find ios apps are way smoother and work better than android .
concrete	wood	hope concrete block is stronger and safer than wood .
michigan	virginia	likewise, michigan has seven centers to serve 10 million people and is more than twice the geographic size of west virginia .
python	ruby	i think that python is a lot easier to read than ruby or java.
plywood	steel	plywood is easier to rework than steel .
hockey	golf	adult hockey is really very safe, probably safer than golf or tennis and certainly safer than soccer .

Table 1: Examples of sentences from our dataset labelled with positions of compared objects, their aspects, as well as predicates used to compare them.

http://ltdemos.informatik.uni-hamburg.de/cam/

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

NER without training data

- Given a textual collection with no annotation
- Given a list of known named entities (e.g. a dictionary)
- Use the ”eyeballing” method to find some heuristic features:
  - Based on morphological and/or syntactic features
  - Based on lexical features and regular expressions
- Example for Russian: Natasha (based on Yargy parser)
  - https://natasha.github.io
- Example for other languages: UIMA, GATE (ANNIE), Unitex
  - http://services.gate.ac.uk/annie/
  - https://unitexgramlab.org

Adopted from slides by Ekaterina Artemova: Analysis of Unstructured Data. HSE.

Samples of the rules

1. Given a two-digit number and a name of the month → a date
   – 13 September, 18 November, etc.
2. Given one capital letter with a period and a word with a capital letter → a name
   – A. Panchenko, C. Biemann, etc.

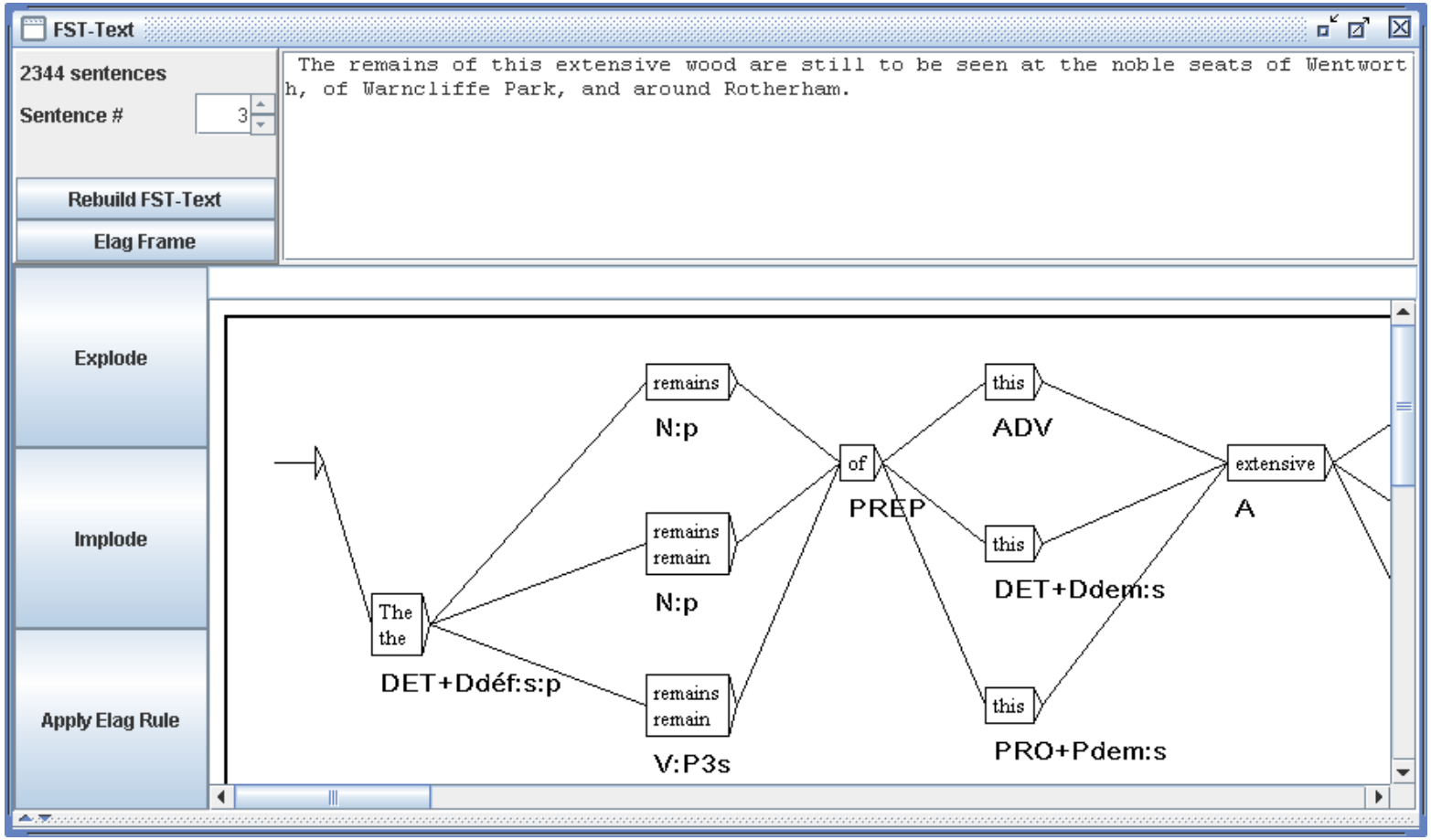
- Rules are usually not complete and may be not exact
- ... but it you have lots of them result can be still useful.

Adopted from slides by Ekaterina Artemova: Analysis of Unstructured Data. HSE.

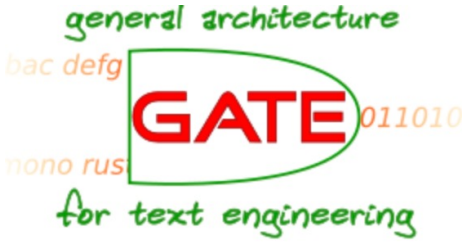
Samples of the rules

https://unitexgramlab.org

- Rules can be represented as FST graphs (here Unitex/Gramlab):



ANNIE tagger



To try the demo please enter some free text to process:

Toyota Motor Corporation is a Japanese multinational automotive manufacturer headquartered in Toyota, Aichi, Japan. In 2017, Toyota's corporate structure consisted of 364,445 employees worldwide and, as of December 2019, was the tenth-largest company in the world by revenue. Toyota is the largest automobile manufacturer in Japan, and the second-largest in the world behind Volkswagen, based on 2018 unit sales.[5] Toyota was the world's first automobile manufacturer to produce more than 10 million vehicles per year, which it has done since 2012, when it also reported the production of its 200 millionth vehicle.[6] As of July 2014, Toyota was the largest listed company in Japan by market capitalization (worth more than twice as much as number 2-ranked SoftBank)[7] and by revenue.[8][9]

↓ Process Text ↓

Toyota Motor Corporation is a Japanese multinational automotive manufacturer headquartered in Toyota, Aichi, Japan. In 2017, Toyota's corporate structure consisted of 364,445 employees worldwide and, as of December 2019, was the tenth-largest company in the world by revenue. Toyota is the largest automobile manufacturer in Japan, and the second-largest in the world behind Volkswagen, based on 2018 unit sales.[5] Toyota was the world's first automobile manufacturer to produce more than 10 million vehicles per year, which it has done since 2012, when it also reported the production of its 200 millionth vehicle.[6] As of July 2014, Toyota was the largest listed company in Japan by market capitalization (worth more than twice as much as number 2-ranked SoftBank)[7] and by revenue.[8][9]

- **Possible advantages**: speed (FSTs or CFGs), no training data, good precision
- **Disadvantages**: not the SOTA F1 measure (low recall), hard to write, maintain, and debug the rules

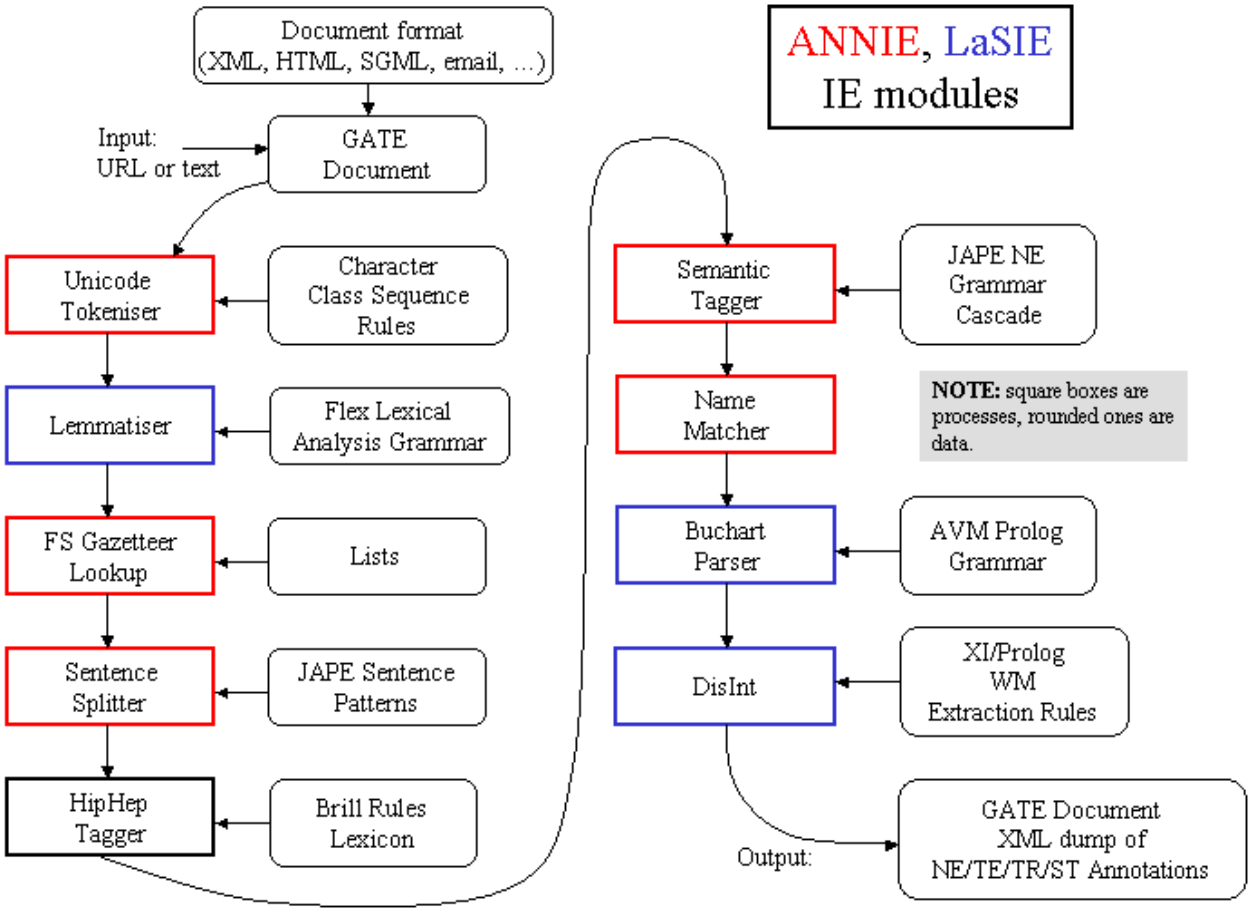
http://services.gate.ac.uk/annie/

ANNIE tagger: comparing to the manual Wikipedia tagging

Toyota Motor Corporation is a Japanese multinational automotive manufacturer headquartered in Toyota, Aichi, Japan. In 2017, Toyota's corporate structure consisted of 364,445 employees worldwide and, as of December 2019, was the tenth-largest company in the world by revenue. Toyota is the largest automobile manufacturer in Japan, and the second-largest in the world behind Volkswagen, based on 2018 unit sales.[5] Toyota was the world's first automobile manufacturer to produce more than 10 million vehicles per year, which it has done since 2012, when it also reported the production of its 200 millionth vehicle.[6] As of July 2014, Toyota was the largest listed company in Japan by market capitalization (worth more than twice as much as number 2-ranked SoftBank)[7] and by revenue.[8][9]

Toyota Motor Corporation (Japanese: トヨタ自動車株式会社, Hepburn: *Toyota Jidōsha KK*, IPA: [to↓jota], English: /tɔɪˈoʊtə/) is a Japanese multinational automotive manufacturer headquartered in Toyota, Aichi, Japan. In 2017, Toyota's corporate structure consisted of 364,445 employees worldwide[4] and, as of December 2019, was the tenth-largest company in the world by revenue. Toyota is the largest automobile manufacturer in Japan, and the second-largest in the world behind Volkswagen, based on 2018 unit sales.[5] Toyota was the world's first automobile manufacturer to produce more than 10 million vehicles per year, which it has done since 2012, when it also reported the production of its 200 millionth vehicle.[6] As of July 2014, Toyota was the largest listed company in Japan by market capitalization (worth more than twice as much as number 2-ranked SoftBank)[7] and by revenue.[8][9]

ANNIE tagger: architecture



PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

HMM Definition and Problems

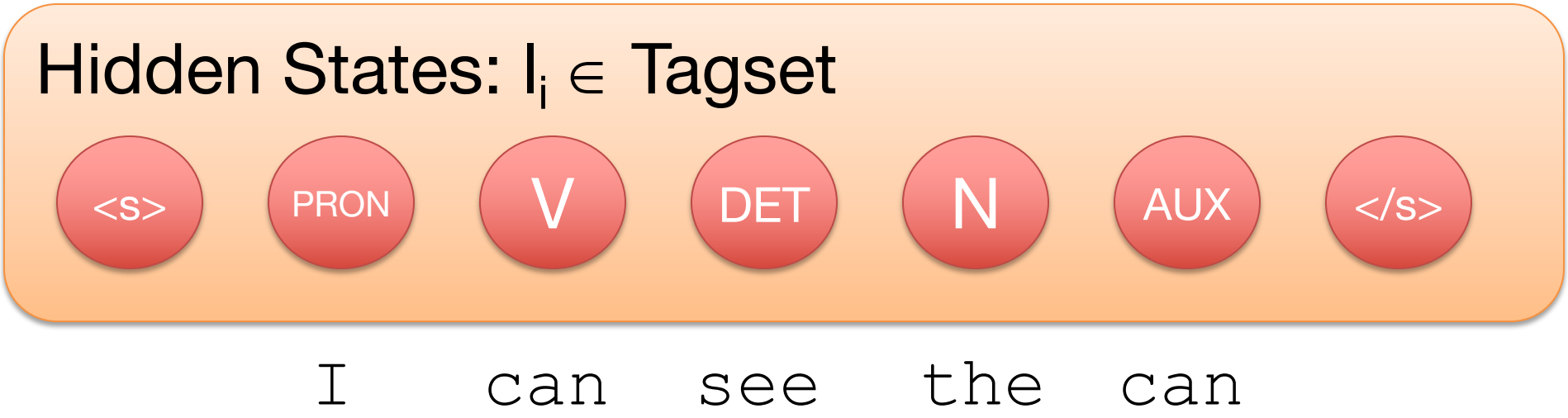
$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Tags can be viewed as hidden states

I can see the can

Tags: states of an HMM



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

Words: Emission from states

Hidden States: $l_i \in$ Tagset

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

I can see the can

Observations: $w_i \in$ Vocabulary

- States have possible words with probabilities

POS HMM training and application

Training:

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

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.

HMM simplifying assumptions

- **Markov assumption**: the probability of a particular state depends only on the previous state

$$P(q_i|q_1,...,q_{i-1}) = P(q_i|q_{i-1})$$

- **Output independence**: probability of an output observation o_i depends only on the state that produced the observation q_i and not on any other states or any other observations

$$P(o_i|q_1,\ldots q_i,\ldots,q_T,o_1,\ldots,o_i,\ldots,o_T) = P(o_i|q_i)$$

Training given labelled data: maximum likelihood estimation

- Data

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

Hidden: weather, **Observed**: # of eaten ice creams

- Priors $\pi_h = 1/3 \quad \pi_c = 2/3$

- State transitions

$$p(hot|hot) = 2/3 \quad p(cold|hot) = 1/3$$
$$p(cold|cold) = 2/3 \quad p(hot|cold) = 1/3$$

- Emissions

$$P(1|hot) = 0/4 = 0 \quad p(1|cold) = 3/5 = .6$$
$$P(2|hot) = 1/4 = .25 \quad p(2|cold = 2/5 = .4$$
$$P(3|hot) = 3/4 = .75 \quad p(3|cold) = 0$$

State transition matrix for a POS HMM

	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

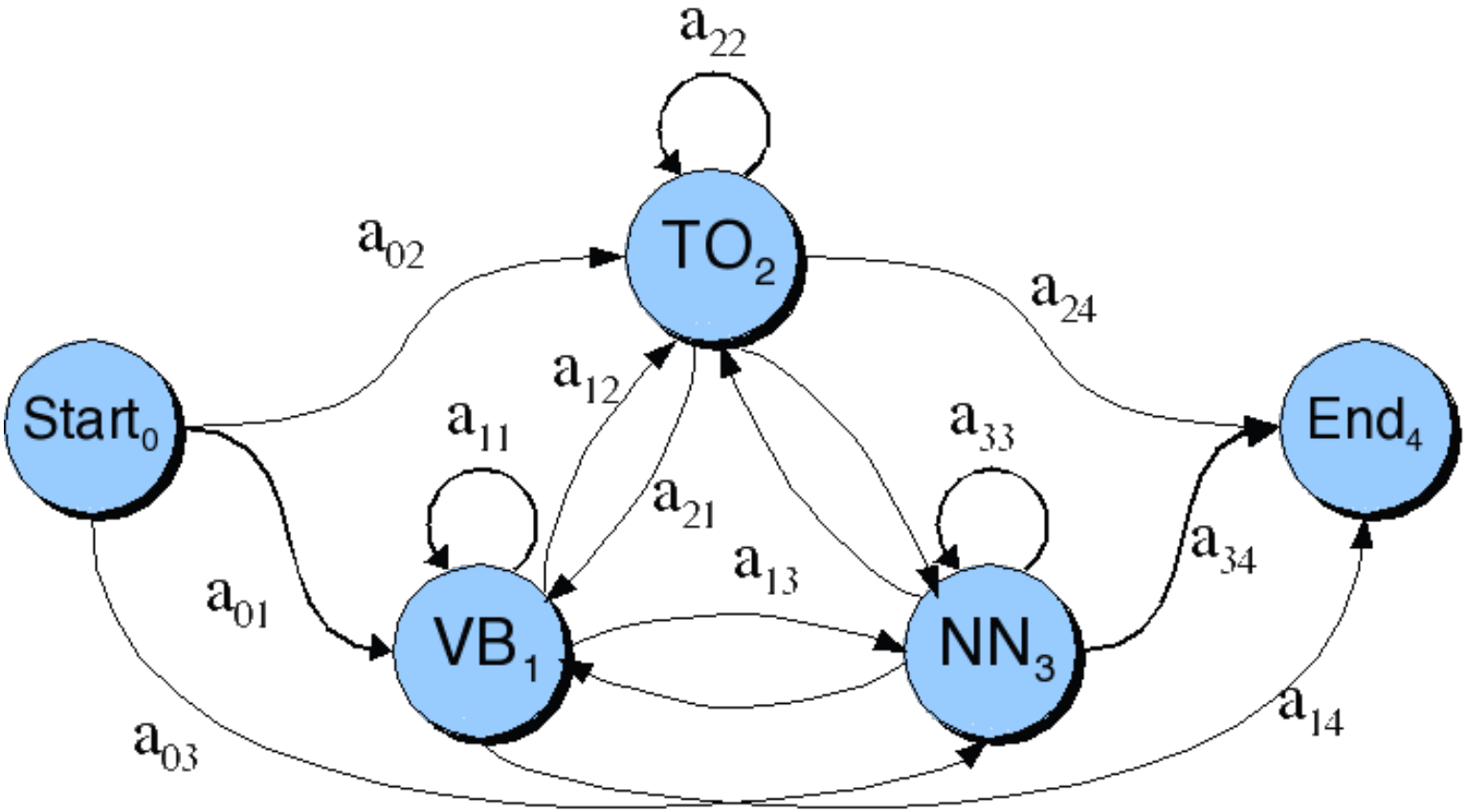
- Tag transition probabilities $p(l_i \mid l_j)$ computed from the 87-tag Brown corpus, without smoothing.
- Rows: Conditioning event, thus P(PPSS|VB) is 0.0070.

Tag-to-word probabilities for a POS HMM

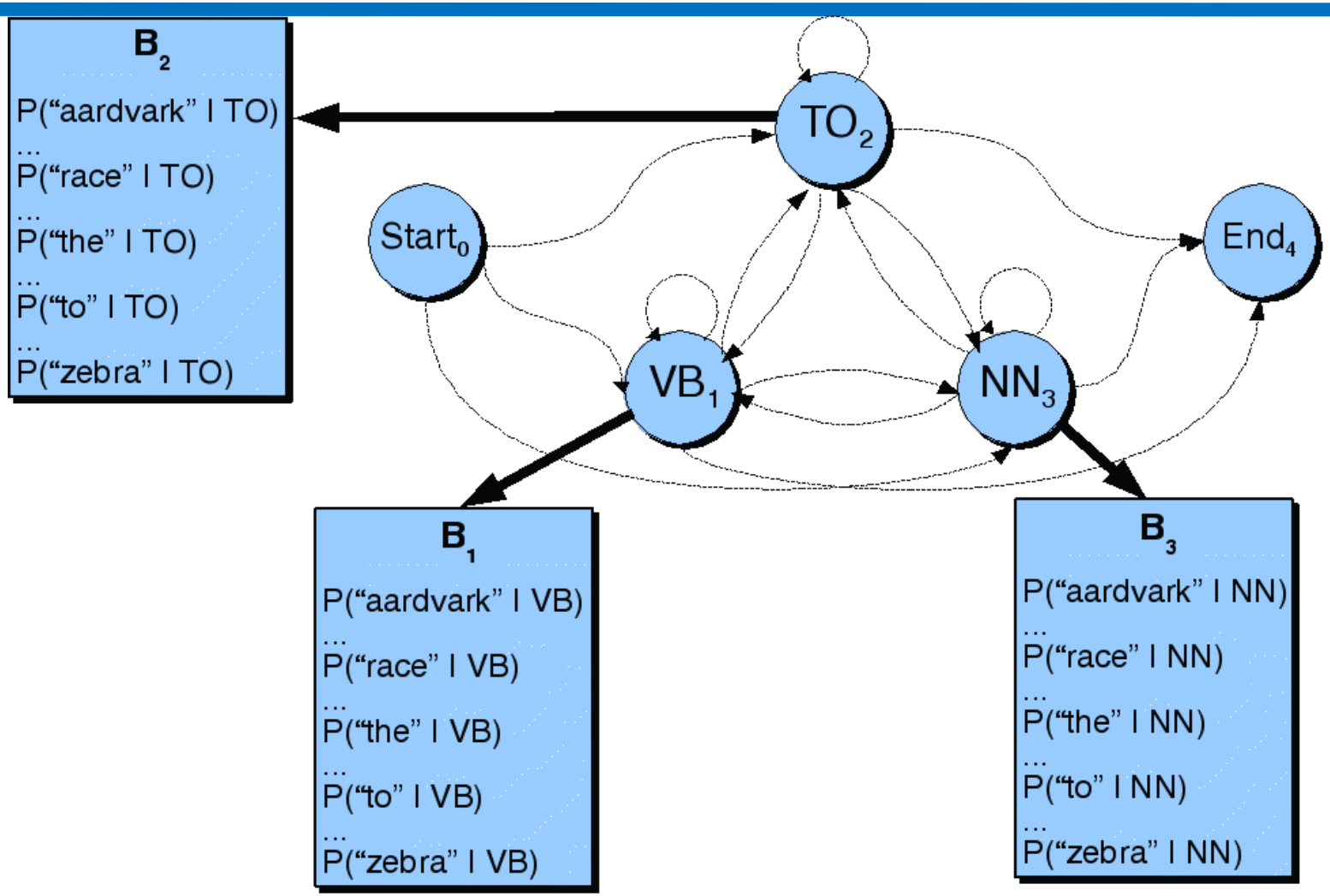
	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

- Observation likelihoods computed from the 87-tag Brown corpus without smoothing.
- This matrix is sparse. What do zeros mean?

Transitions between states with probabilities



Observation likelihoods: other probabilities



Decoding: Finding the best state sequence given observations

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n)$$

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} \frac{P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)}$$

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

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

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n) \approx \underset{t_1 \ldots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

Decoding: Finding the best state sequence given observations

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

- Use dynamic programming and define the probability of the best path to a state after t symbols.

- The iterative algorithm to solve this problem is called the Viterbi algorithm after Andrew Virerbi.

Citation	Field
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science

Viterbi Algorithm

Each cell of the lattice, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $q_1, ..., q_{t-1}$, given the HMM λ. The value of each cell $v_t(j)$ is computed by recursively taking the most probable path that could lead us to this cell. Formally,

Notation:

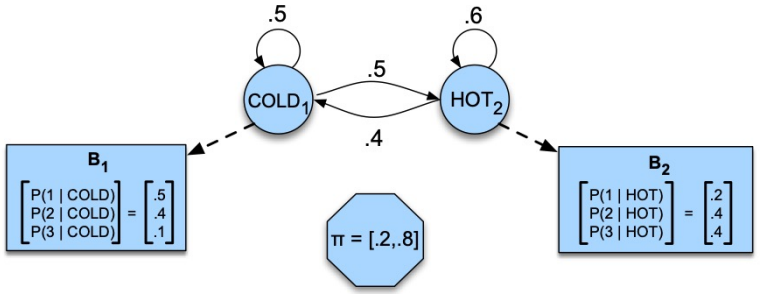
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Algorithm:

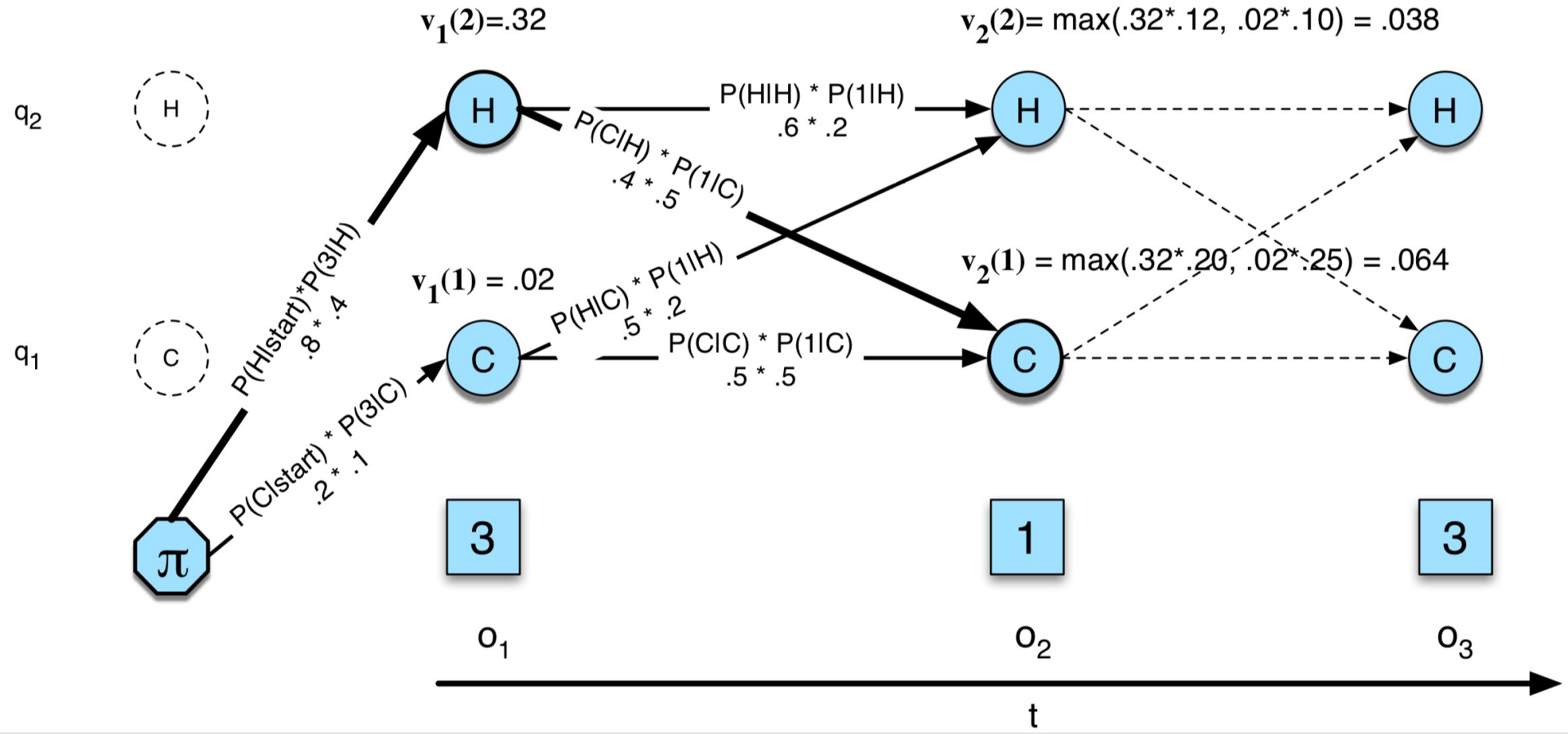
function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
$bestpath \leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

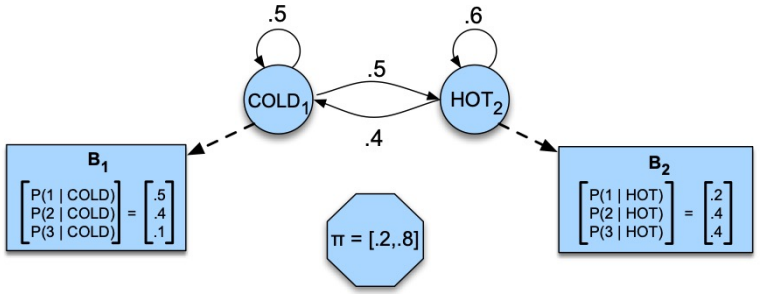
Viterbi Algorithm



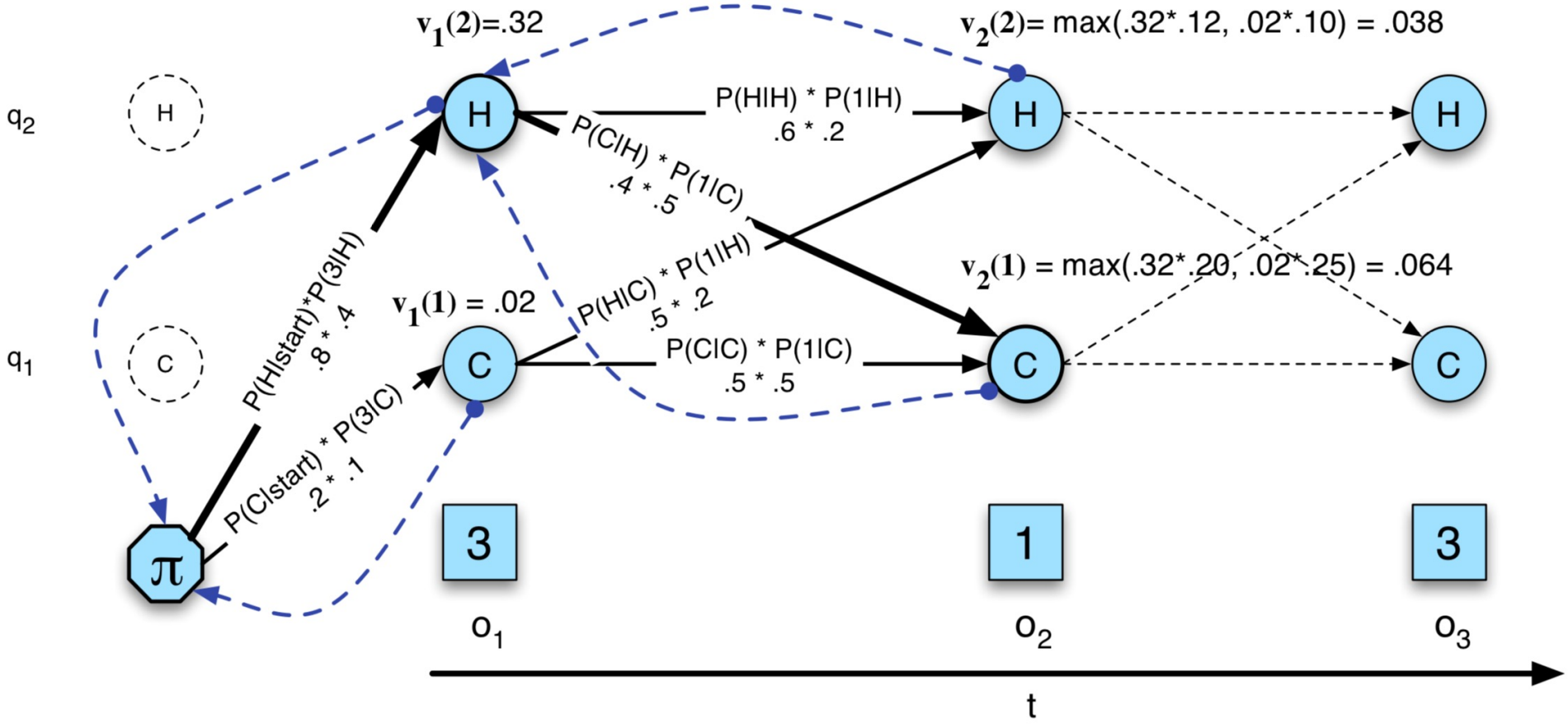
The Viterbi trellis for computing the best path through the hidden state space for "**3 1 3**":



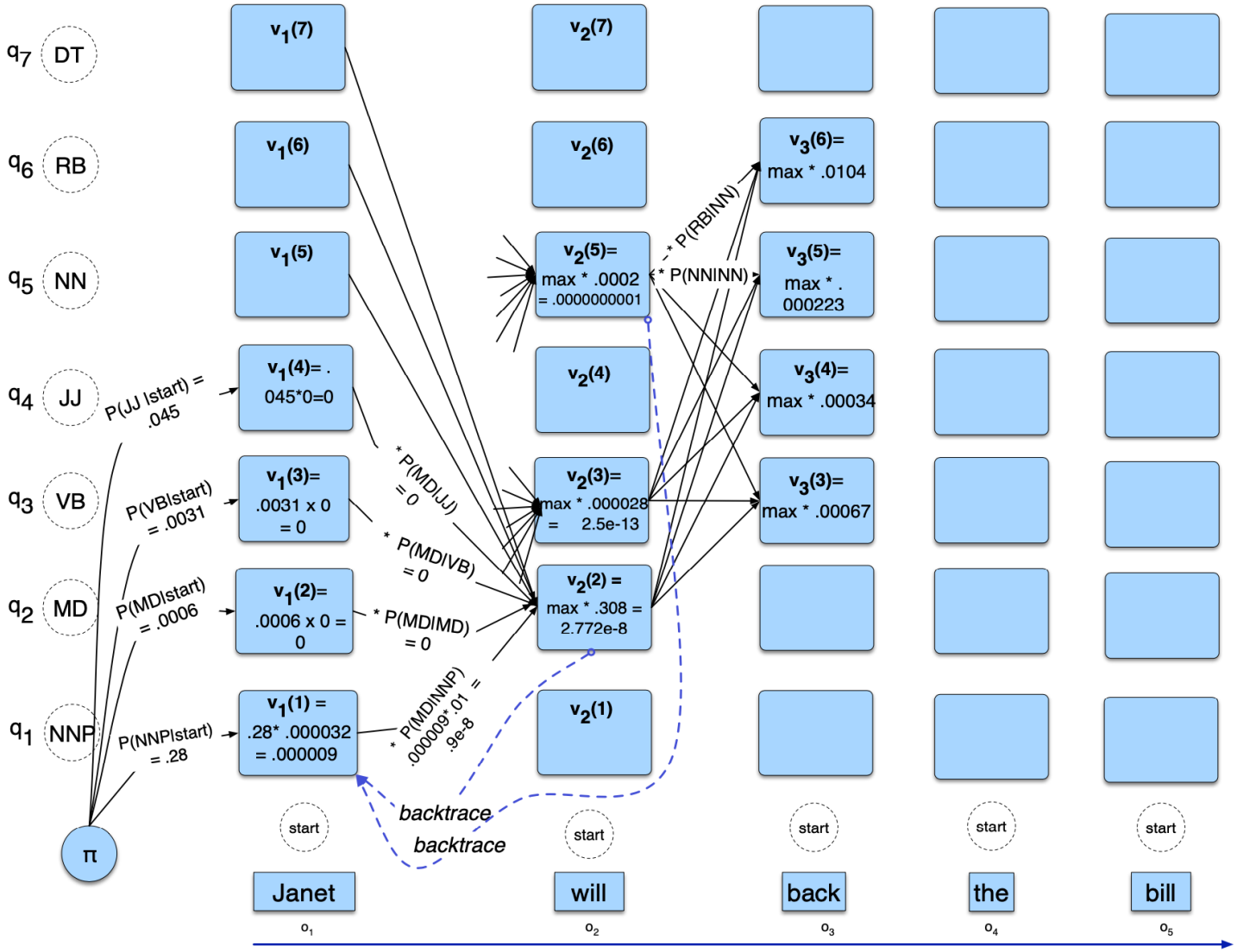
Viterbi Algorithm



The Viterbi backtrace. As we extend each path to a new state account for the next observation, we keep a backpointer to the best path that led us to this state.



For the case of POS tagging the table gets larger



Trigrams'n'Tags TNT POS tagger by Brants (2000)

- TnT uses second-order HMM for POS-tagging:

$$\arg\max[\prod_{j}[p(t_i|t_{i-1}, o_{t-2})p(w_i|t_i)]P(t_{T+1}|t_T)$$

- The probability of a POS-tag for a given word is computed as a linear interpolation of three LM's:

$$P(t_i|t_{i-1}, t_{i-2}) = l_1 * P(t_i) + l_2 * P(t_i|t_{i-1}) + l_3 * P(t_i|t_{i-1}, t_{i-2})$$

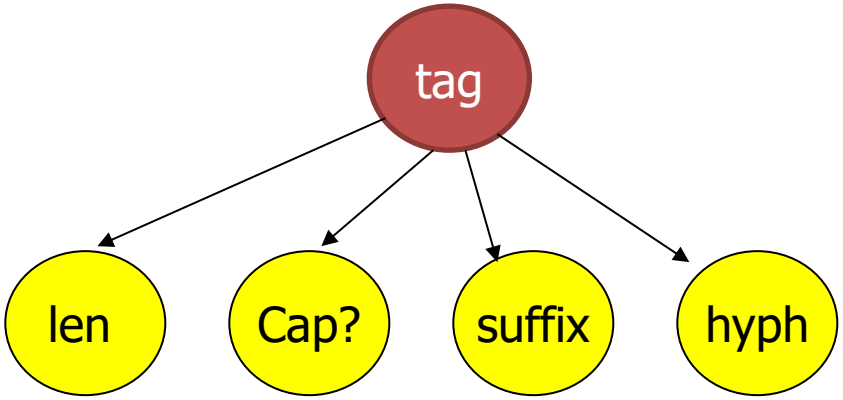
- Available in NLTK (English_HMM_POS_tagger)

  https://www.nltk.org/_modules/nltk/tag/tnt.html
  https://www.coli.uni-saarland.de/~thorsten/tnt/

Adopted from slides by Ekaterina Artemova's course on Analysis of Unstructured Data at HSE.

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)?

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Discriminative vs generative models

Generative model:

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} P(W|T)P(T) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})\end{aligned}$$

Discriminative model:

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(tag_i|word_i, tag_{i-1})\end{aligned}$$

Maximum Entropy Markov Model (MEMM)

Linear Regression: $y = w \cdot f$

Binary Logistic Regression: $$\ln\left(\frac{p(y=\text{true}|x)}{1-p(y=\text{true}|x)}\right) = w \cdot f$$

$$p(y=\text{true}|x) = \frac{e^{w \cdot f}}{1+e^{w \cdot f}}$$

Multinomial Logistic Regression:

$$p(c|x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} f_i\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^{N} w_{c'i} f_i\right)}$$

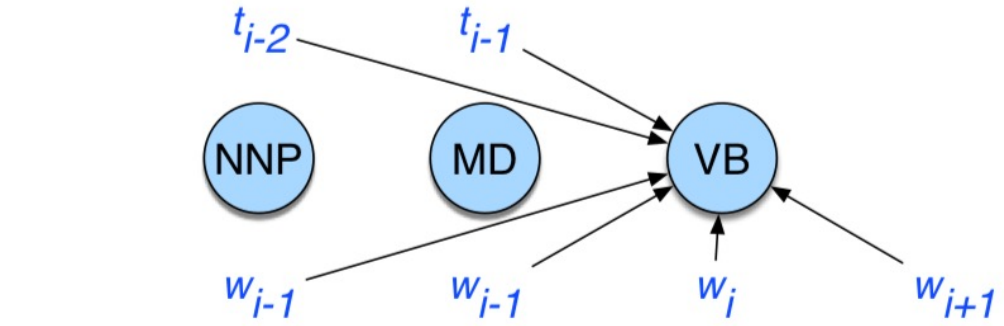
Maximum Entropy Markov Model (MEMM):

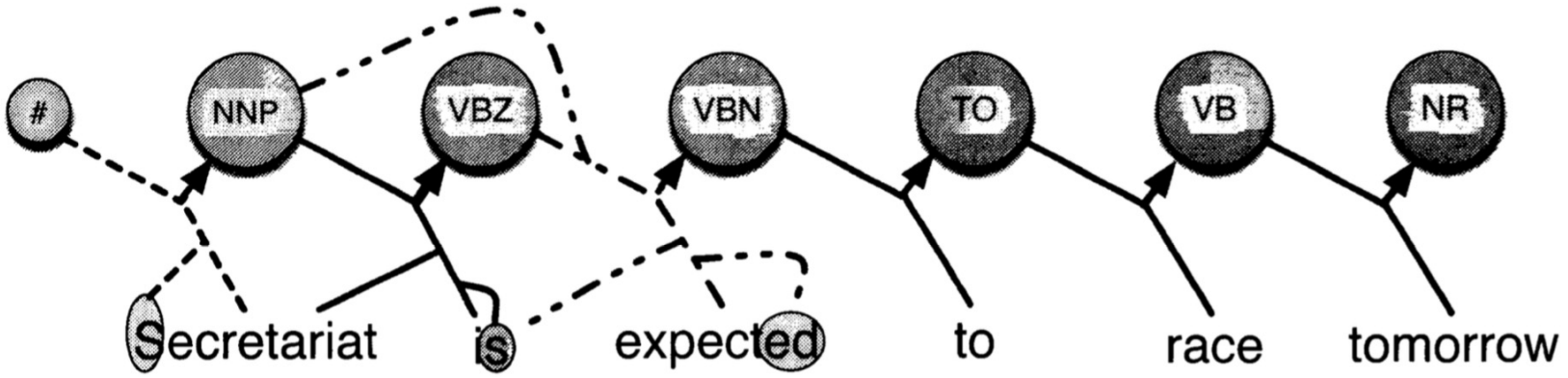
$$P(Q|O) = \prod_{i=1}^{n} P(q_i|q_{i-1}, o_i)$$

$$P(q|q', o) = \frac{1}{Z(o, q')} \exp\left(\sum_i w_i f_i(o, q)\right)$$

Maximum Entropy Markov Model (MEMM)

MEMMs for part-of-speech tagging showing the ability to condition on more features than the HMM could do:





Example of feature functions

Feature templates:

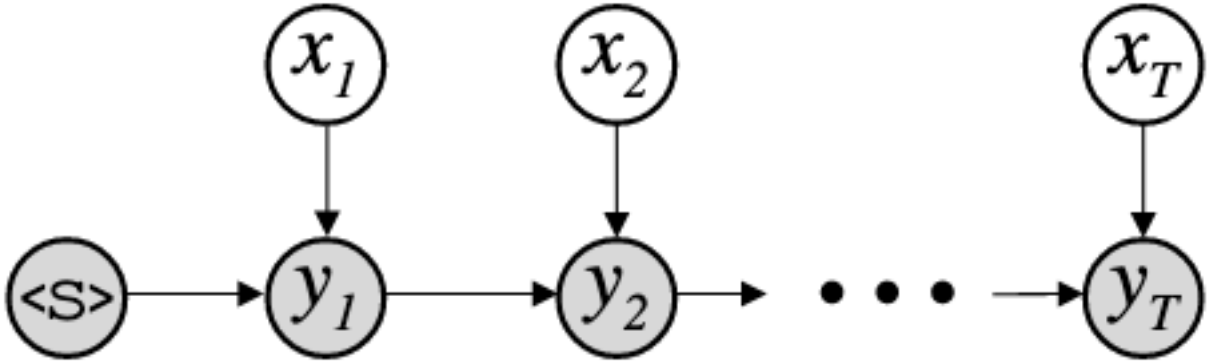
$$\langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle$$
$$\langle t_i, t_{i-1} \rangle, \langle t_i, t_{i-2}, t_{i-1} \rangle,$$
$$\langle t_i, t_{i-1}, w_i \rangle, \langle t_i, w_{i-1}, w_i \rangle \langle t_i, w_i, w_{i+1} \rangle,$$

Sample features:

t_i = VB and w_{i-2} = Janet
t_i = VB and w_{i-1} = will
t_i = VB and w_i = back
t_i = VB and w_{i+1} = the
t_i = VB and w_{i+2} = bill
t_i = VB and t_{i-1} = MD
t_i = VB and t_{i-1} = MD and t_{i-2} = NNP
t_i = VB and w_i = back and w_{i+1} = the

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

Maximum Entropy Markov Model (MEMM)

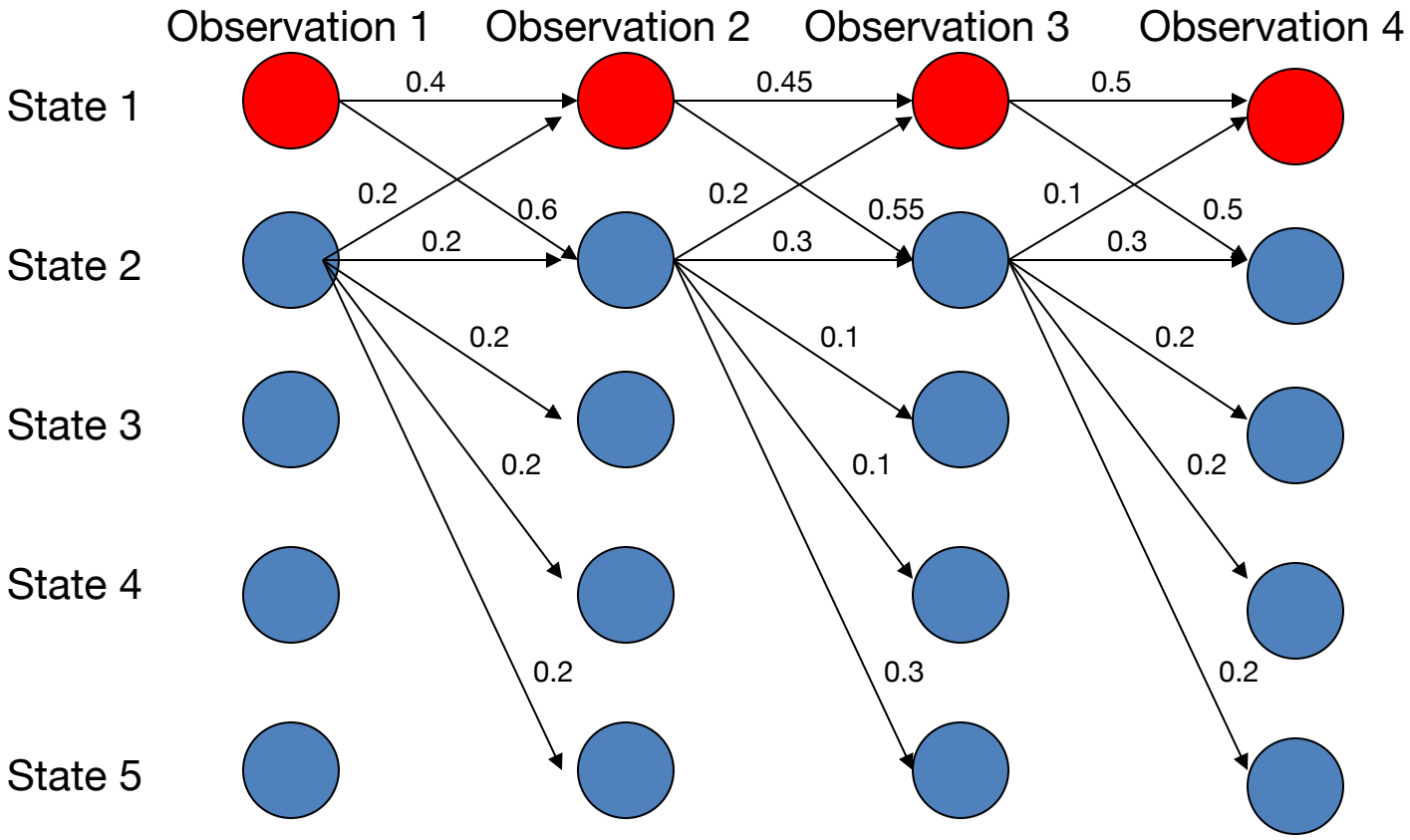


$$p(Y \mid X) = \prod_{t=1}^{T} p(y_t \mid x_t, y_{t-1}).$$

$$p(y_t \mid x_t, y_{t-1}) = \frac{e^{s(y_t, x_t, y_{t-1})}}{\sum_{i=1}^{c} e^{s(y_i, x_t, y_{t-1})}}$$

- Label y_t is conditionally independent of all previous observations and labels given x_t and y_{t−1}.
- The observations x_t are independent of one another.
- Normalized over the set of possible output labels at each time step, we say the model is *locally normalized*

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.

Reasons of label bias problem

https://awni.github.io/label-bias/
http://cseweb.ucsd.edu/~elkan/254spring02/gidofalvi.pdf

$$p(y_t \mid x_t, y_{t-1}) = \frac{e^{s(y_t, x_t, y_{t-1})}}{\sum_{i=1}^{c} e^{s(y_i, x_t, y_{t-1})}}$$

- 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**…
- In the extreme case, a state with a single outgoing transition effectively **ignores the observation.**

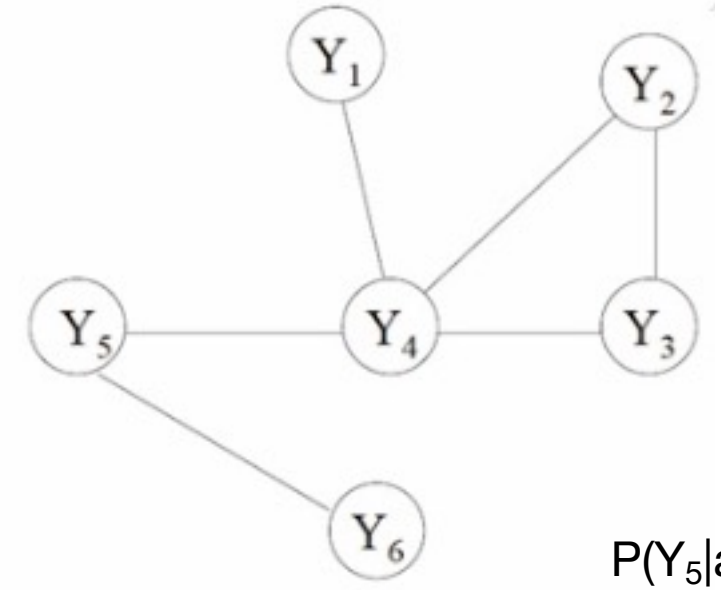
PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Random Fields: mutual dependence in undirected graphs

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

Example:



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

- Random fields use global normalization.
- Potential functions operate on cliques of the G.

CRF: Conditional Random Field

- In a CRF, we compute the posterior $p(Y|X)$ directly, training the CRF to discriminate among the possible tag sequences:

$$\hat{Y} = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X)$$

- Function F maps an entire input sequence X and an entire output sequence Y to a feature vector. Assuming K features, with a weight w_k for each feature F_k:

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

CRF: Conditional Random Field

- Denominator is commonly represented with function Z:

$$p(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)$$

$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)$$

- Functions F_k(X,Y) are called **global features**. In linear chain CRF **local features** are used::

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

$$\mathbb{1}\{x_i = \textit{the}, y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN}, x_{i+1} = \textit{Street}, y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB}, y_{i-1} = \text{AUX}\}$$

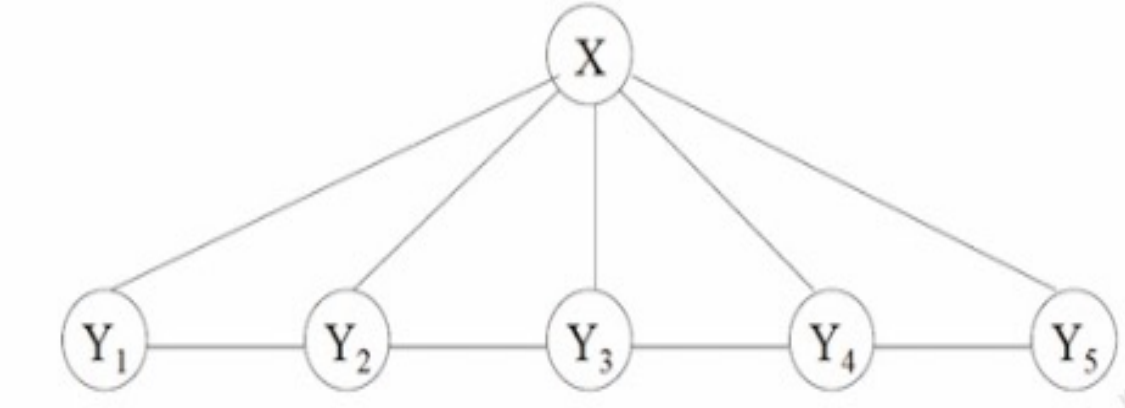
Properties of CRF

$$p(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)$$

$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)$$

- Like MEMM, it is a discriminative exponential model.
- 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**.
- 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



- 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
- **Most used in NLP!**

Decoding with CRFs

How do we find the best tag sequence $\hat{Y}$ for a given input X?

$$
\begin{aligned}
\hat{Y} &= \operatorname*{argmax}_{Y \in \mathcal{Y}} P(Y|X) \\
&= \operatorname*{argmax}_{Y \in \mathcal{Y}} \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \\
&= \operatorname*{argmax}_{Y \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)\right) \\
&= \operatorname*{argmax}_{Y \in \mathcal{Y}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \\
&= \operatorname*{argmax}_{Y \in \mathcal{Y}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)
\end{aligned}
$$

Decoding: Viterbi algorithm

- Filling an N x T array with the appropriate values, maintaining backpointers as we proceed.
- HMM update step:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i)\, P(o_t|s_j) \quad 1 \le j \le N, 1 < t \le T$$

- CRF update step:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t) \quad 1 \le j \le N, 1 < t \le T$$

Training CRFs

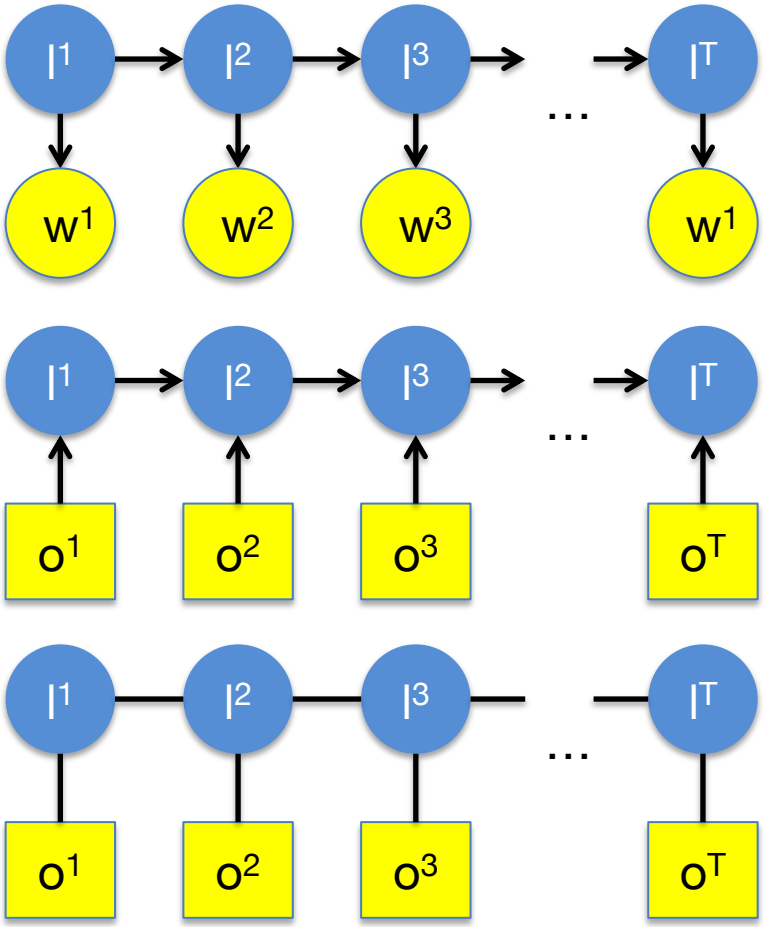
$$\underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)$$

- Mutual dependence between label random variables does not allow simple counting in MLE style.

Several methods for training CRFs (the same as for training a logistic regression):

- Gradient descent (fast)
  - Given a sequence of observations, feature functions, and corresponding outputs, we use stochastic gradient descent to train the weights to maximize the log-likelihood of the training corpus.
- Iterative scaling (slow)
- Conjugate Gradient method
- Limited Memory Quasi-Newton Methods

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

○ variable generated by model
□ observation

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.

PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

No Limited Horizon Assumption

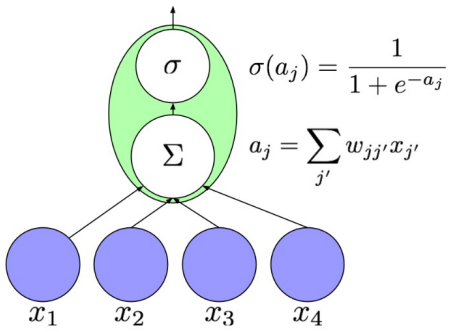
Reason for limited horizon: efficiency! Need a reasonably small number of discrete states for quadratic runtime of Viterbi.

Dropping this assumption requires:

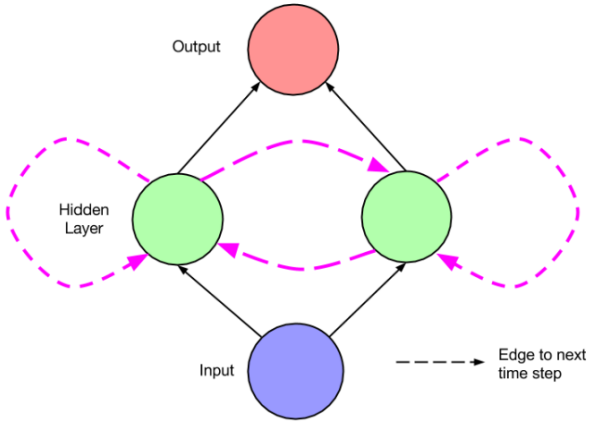
- Keeping the representation of states small: A binary activation layer of N neurons can represent 2^N discrete states, continuous activations even more.

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

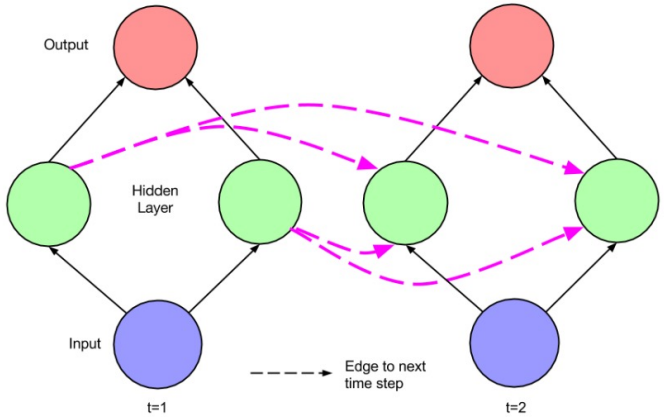
Recurrent Neural Networks (RNNs)



Artificial neuron



Simple recurrent network with 2 hidden units



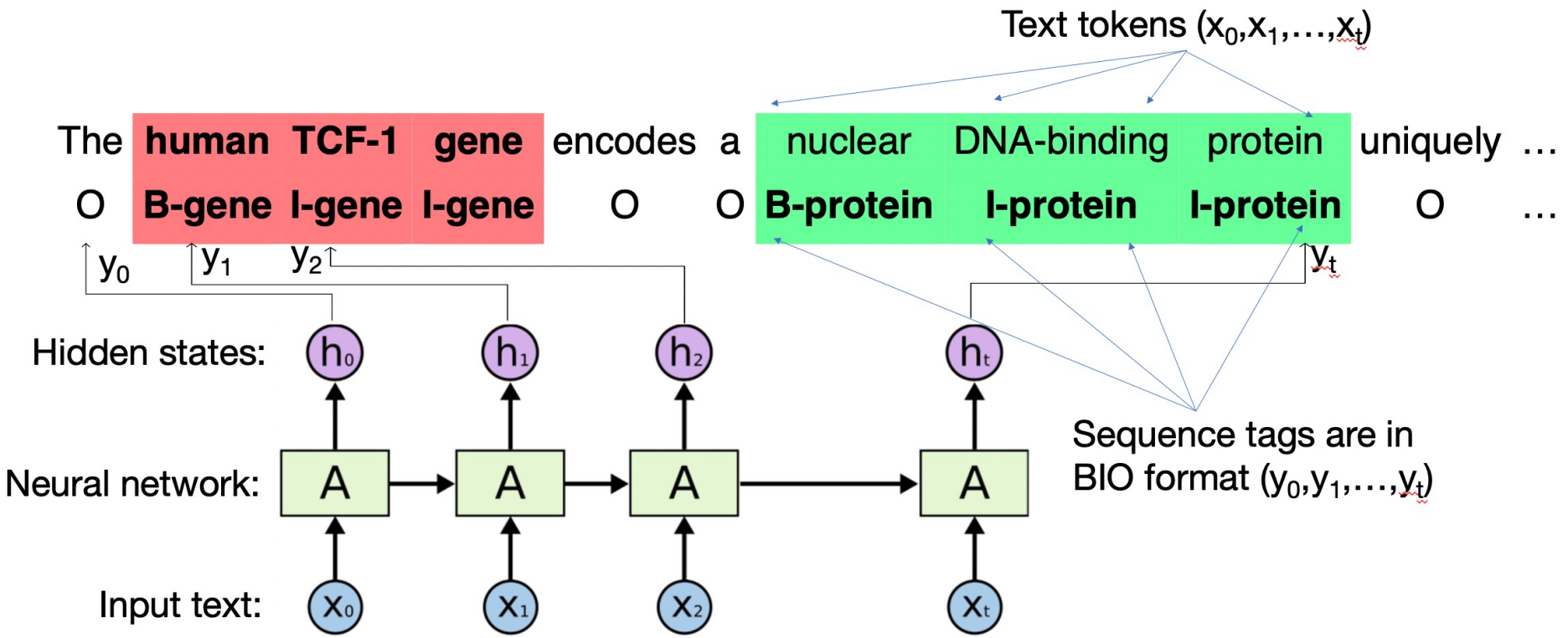
Same recurrent network unfolded over time

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b}$$

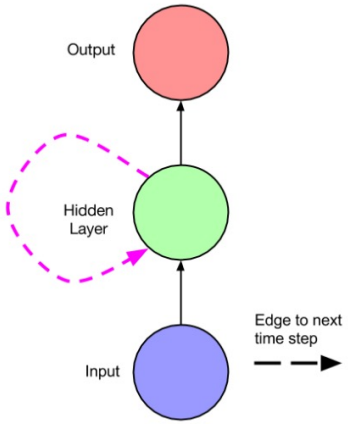
- Recurrent connections: input at time step comes from activation at time step (t-1)
- Recurrent connections introduce notion of sequence into the network
- Unfolding: Can view recurrent network like a deep network, can apply gradient-based training
- Training: backpropagation (through time, BPTT via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

Reminder on backprop: https://blog.yani.ai/backpropagation/

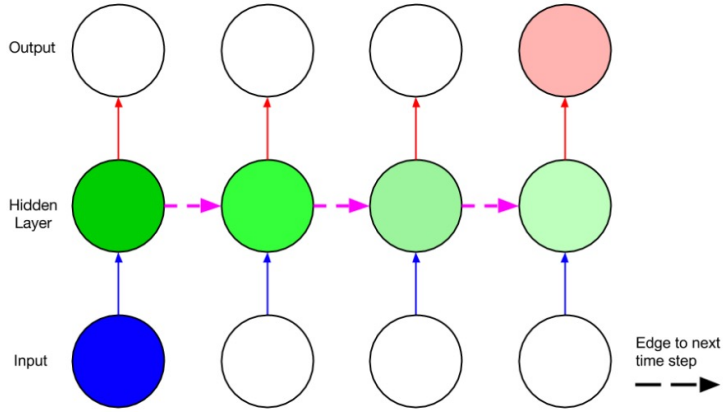
RNNs for tagging: an example from the JNLPBA (GENIA) corpus



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.
- In the same way a product of n real numbers can shrink to zero or explode to infinity, so does this product of matrices.

- Pascanu, R., Tomas M., and Yoshua B. On the difficulty of training recurrent neural networks. ICML 2013
- Graves A. Supervised sequence labelling with recurrent neural networks, Volume 385. Springer, 2012.

LSTM intuitions

Memory cell:

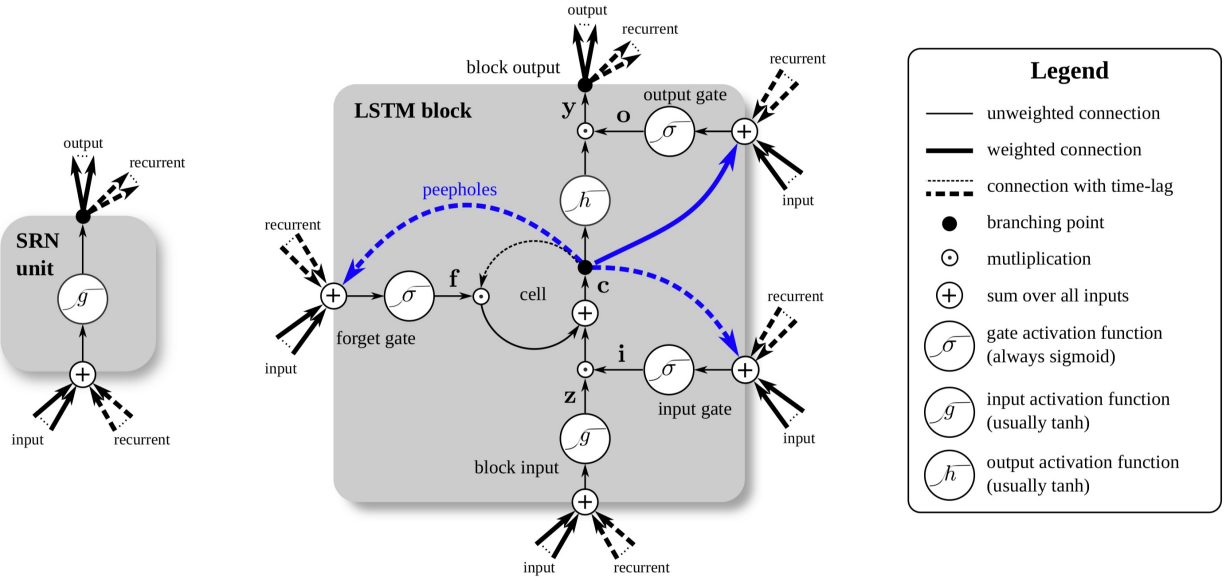
- Internal state serves as a memory
- Recurrent connection of weight 1: error can flow across time steps without vanishing or exploding

Gates:

- Pointwise multiplication in gates regulates how much is passed through, based on inputs. This way, for instance, an LSTM can learn:
  - when to let the input in, e.g. set the new grammatical subject
  - when to let the output out, e.g. predict verb that takes the subject
  - when to reset its memory, e.g. remove old subject once its taken

Long Short-Term Memory (LSTM)

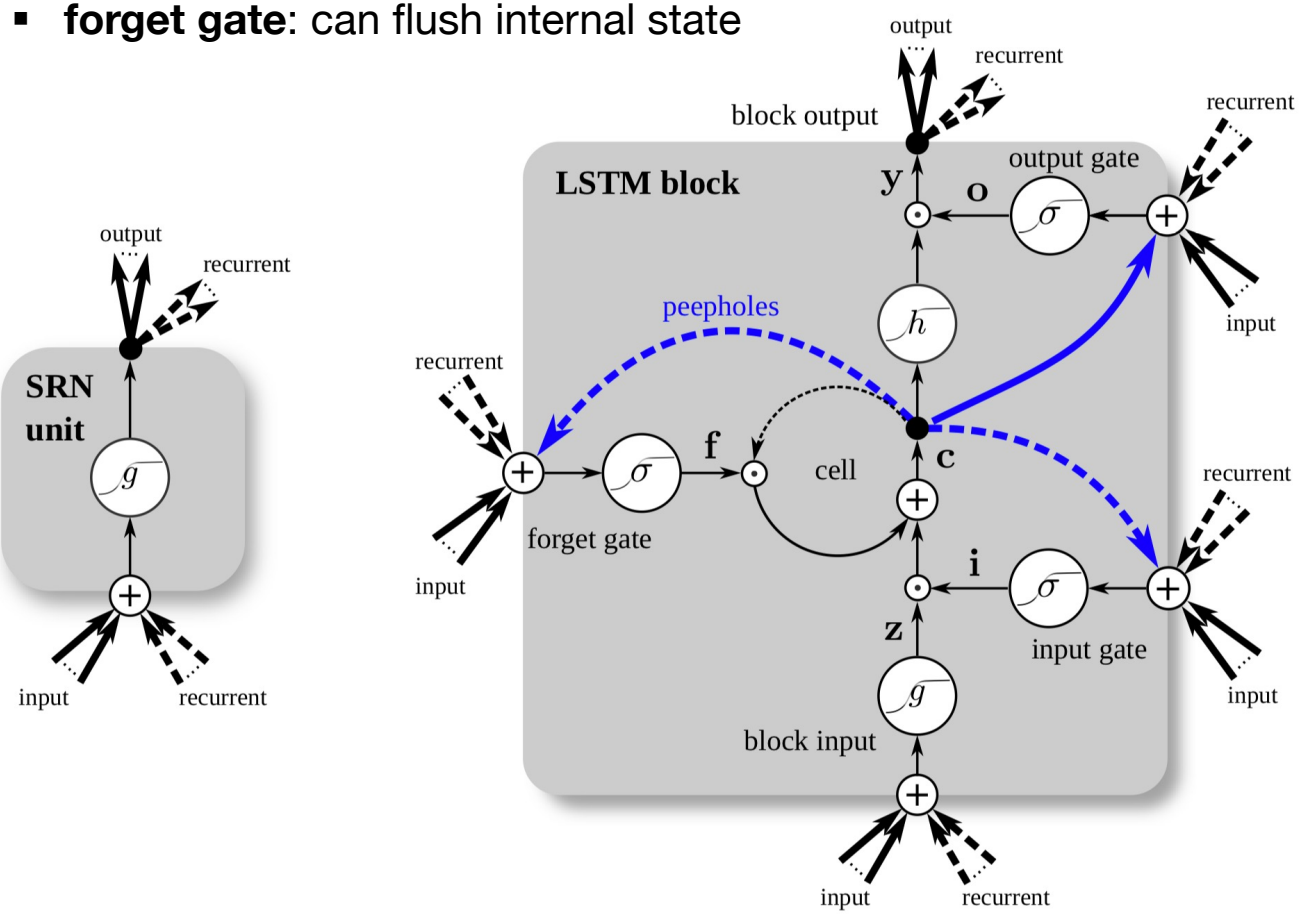
- LSTM resembles a standard RNN with a hidden layer
- Nodes in the hidden layer are replaced by a memory cell
- Memory cells contain a node with a self-connected recurrent edge of fixed weight 1 (no gradient issues):

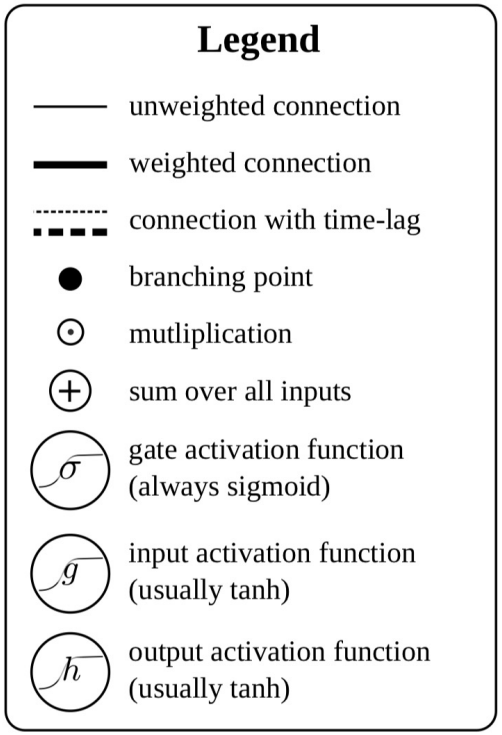


- Hochreiter S. and Schmidhuber H. (1997). Long short-term memory. Neural Computation, 9(8):1735–1780.
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE Trans. on neural networks and learning systems, 28(10)

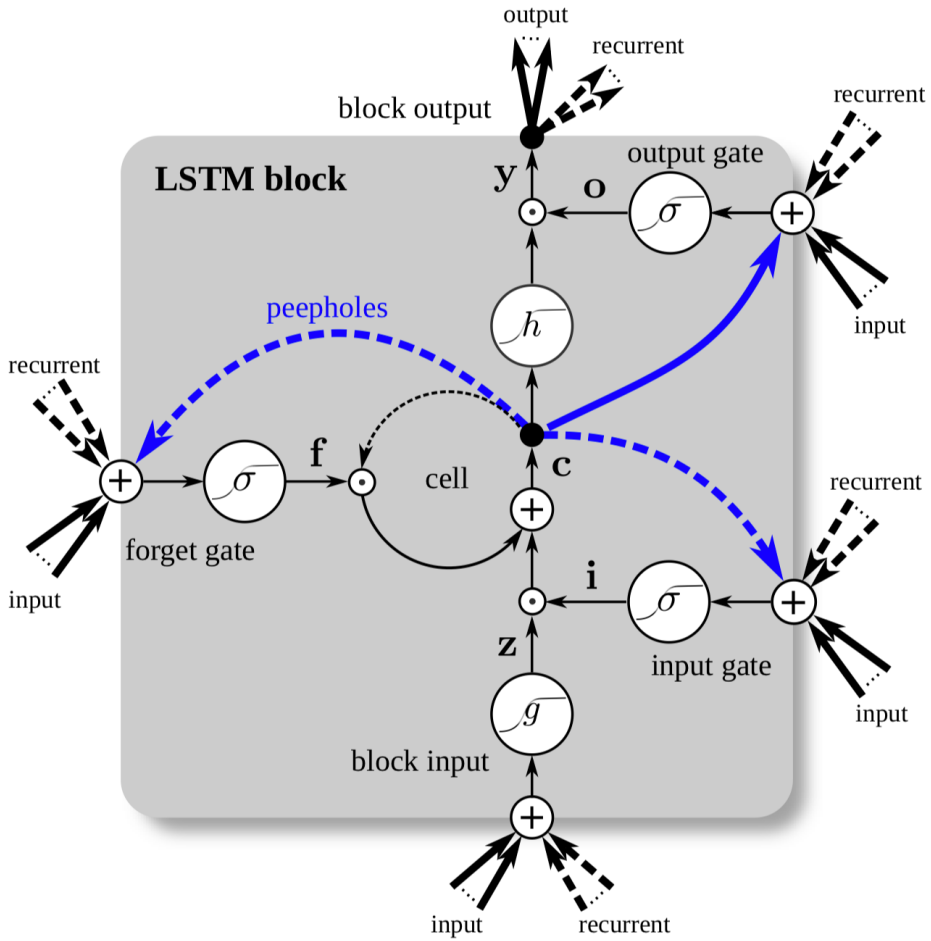
Long Short-Term Memory (LSTM)

- **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
- **recurrent link** with weight 1: “constant error carousel”.
- **forget gate**: can flush internal state





Memory Cell in LSTM



- Input weights: $\mathbf{W}_z$, $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o \in \mathbb{R}^{N \times M}$
- Recurrent weights: $\mathbf{R}_z$, $\mathbf{R}_i$, $\mathbf{R}_f$, $\mathbf{R}_o \in \mathbb{R}^{N \times N}$
- Peephole weights: $\mathbf{p}_i$, $\mathbf{p}_f$, $\mathbf{p}_o \in \mathbb{R}^{N}$
- Bias weights: $\mathbf{b}_z$, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o \in \mathbb{R}^{N}$

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1} + \mathbf{b}_z$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t) \qquad \textit{block input}$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t) \qquad \textit{input gate}$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t) \qquad \textit{forget gate}$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t \qquad \textit{cell}$$

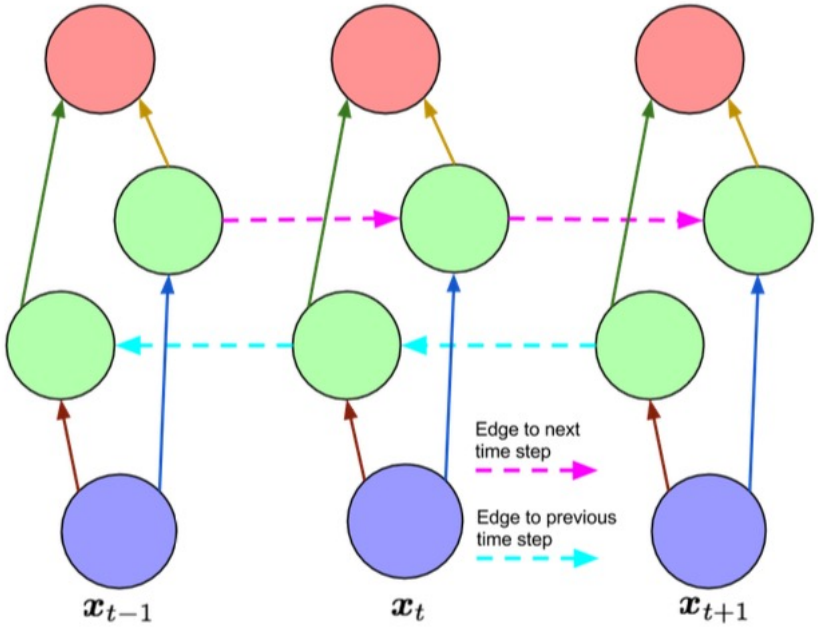
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t) \qquad \textit{output gate}$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t \qquad \textit{block output}$$

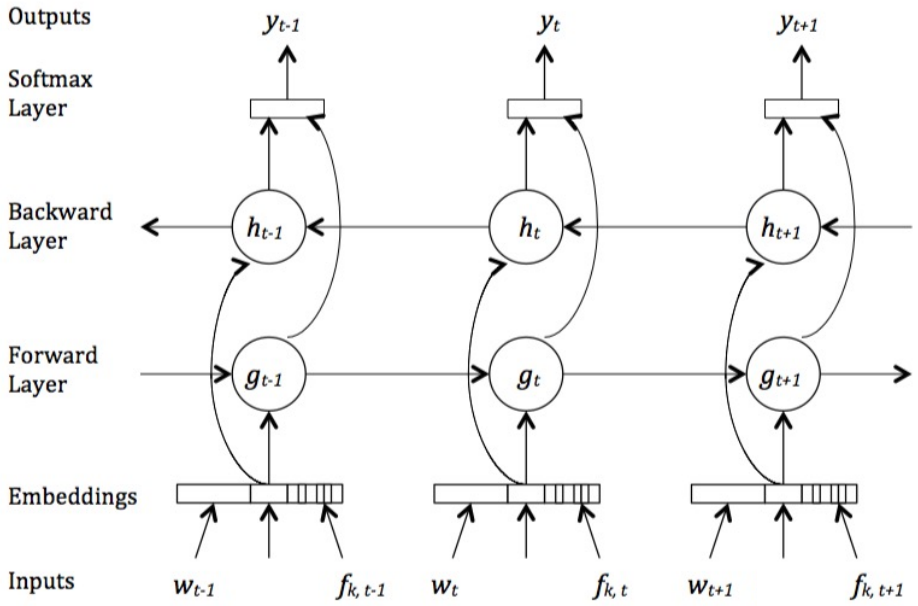
Bidirectional LSTMs (Bi-LSTMs)

- Idea: if we are tagging whole sentences, we can use context representations from the ‘past’ and from the ‘future’ to predict the ‘current’
- 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



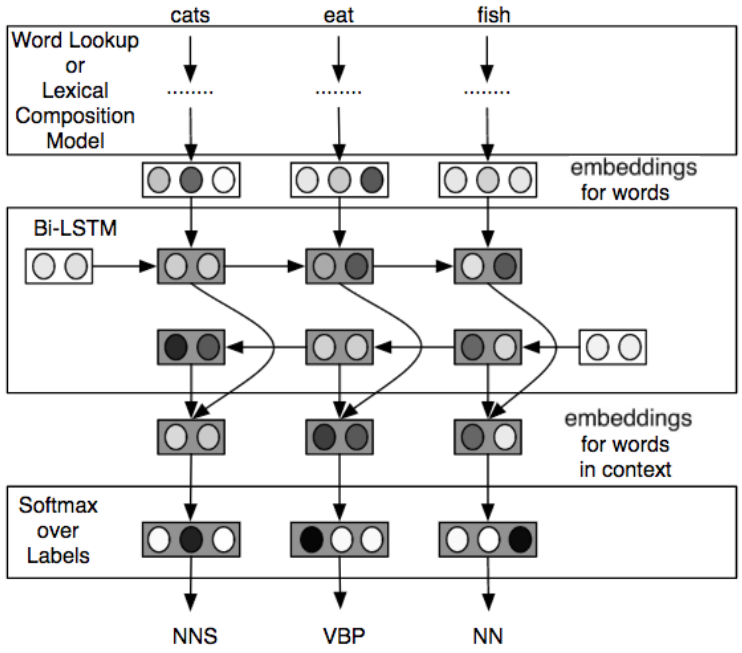
- Input: Word embeddings, additional word features
- Combine two directions: usually concatenation
- Output: 1-hot-encoding over labels (softmax)

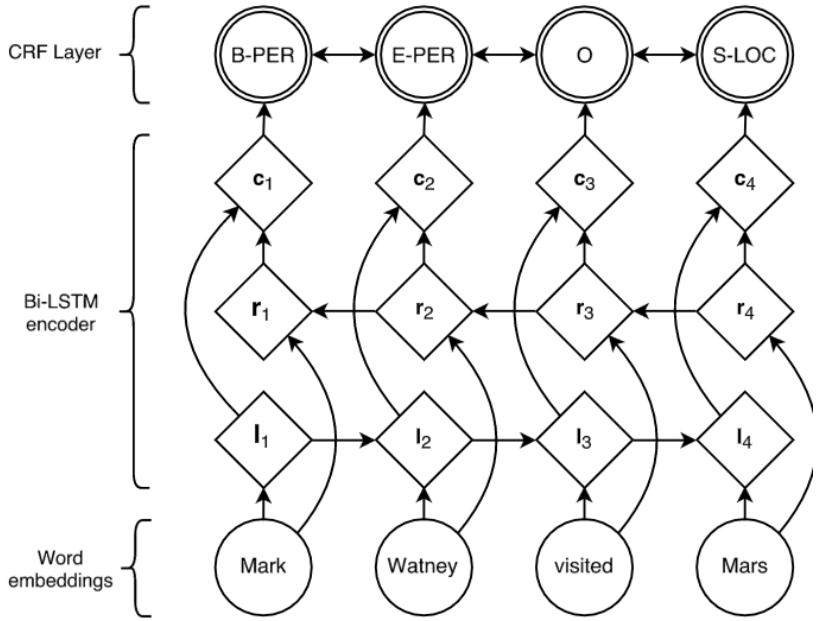
Note:

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

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

Bi-LSTM for POS tagging: variants

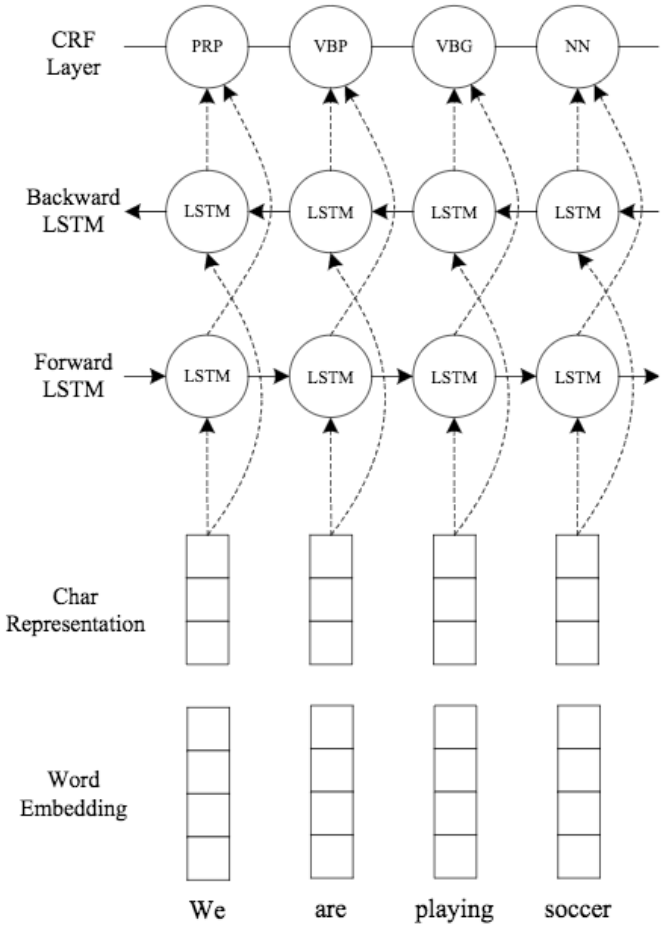




- Compose words from **character embeddings** to addess unseen words
- Use combined outputs as features in **CRF layer**, better making use of neighboring labels

- 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
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016): Neural Architectures for Named Entity Recognition. Proceedings of NAACL.

Bi-LSTM for NER



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**

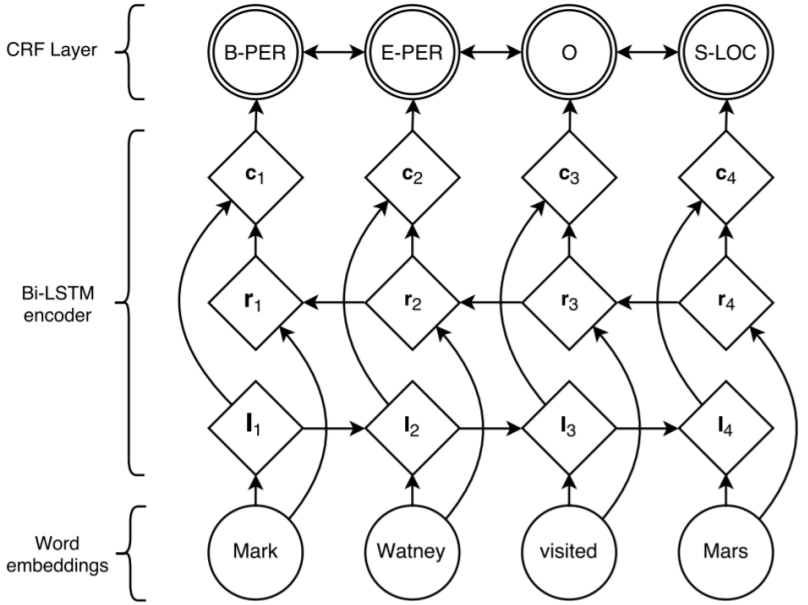
‡ marks the neural models

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

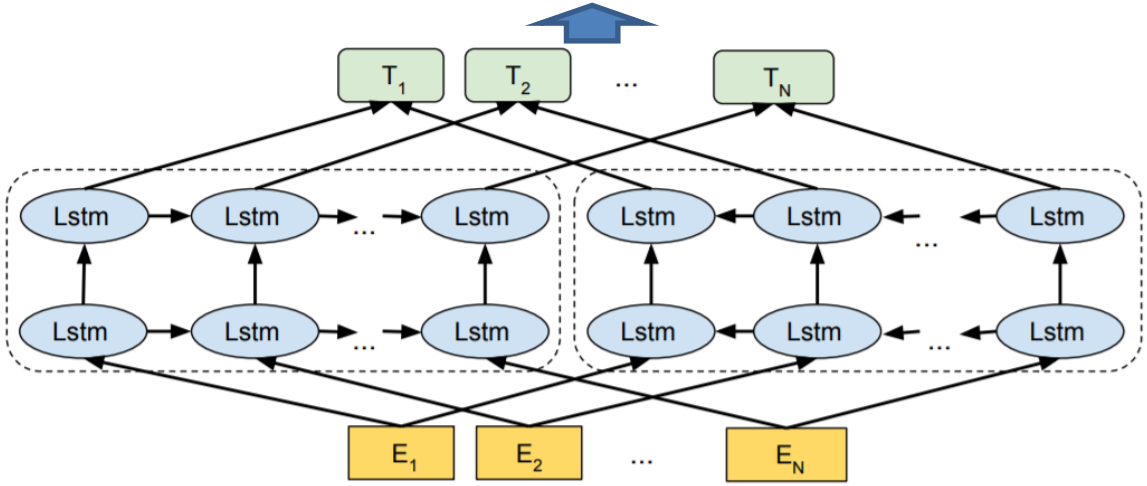
ELMo: deep contextualized word representations for NER

BiLSTM-CRF (Ma and Hovy, 2016) and (Lample et al., 2016)

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%



ELMo (Peters et al., 2018)



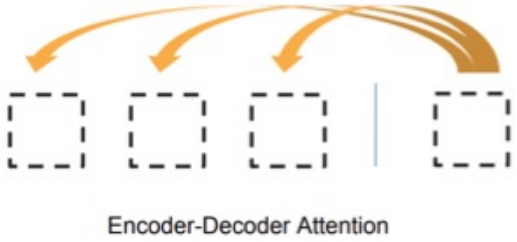
PLAN OF THE LECTURE

- Sequence Tagging Tasks: Part-of-Speech Tagging, Named Entity Recognition, etc.
- Rule-based Taggers
- Generative Taggers: HMMs
- Discriminative Taggers: MEMMs and CRFs
- Neural Taggers: RNNs and Transformers

Transformer: let's get rid of RNNs

- RNNs are slow, because not parallelizable over timesteps
- Attention is parallelizable + have shorter gradient paths
  - Sequence transduction w/o RNNs/CNNs (attention+FF)
  - SOTA on En→Ge WMT14, better than any single model on En→Fr WMT14 (but worse than ensembles)
  - Much faster than other best models (base/big: 12h/3.5d on 8GPUs

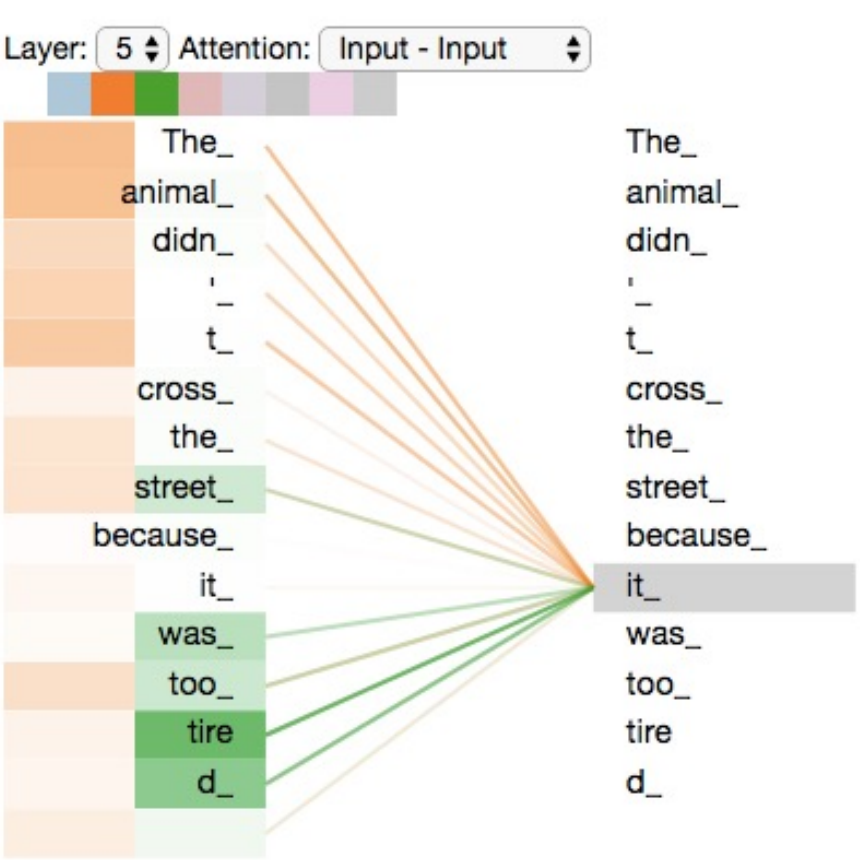
Vaswani et al. (2017) Attention is all you need.

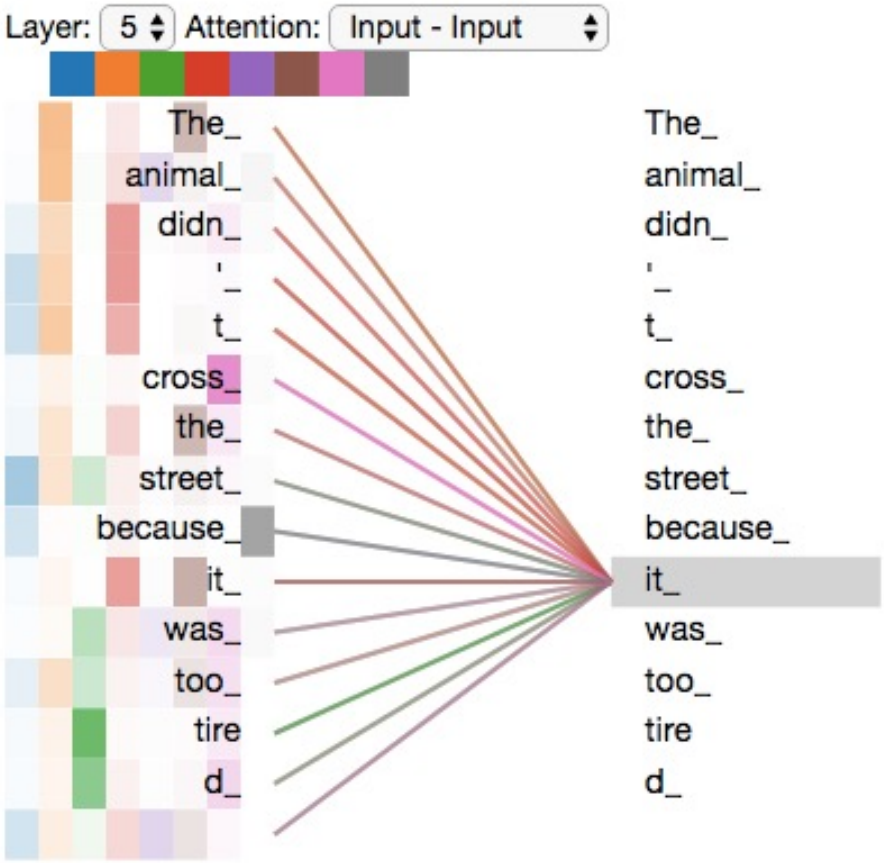


Queries = **K**eys = **V**alues = word embeddings $A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/lectures/lecture12.pdf

Multihead self-attention in encoder

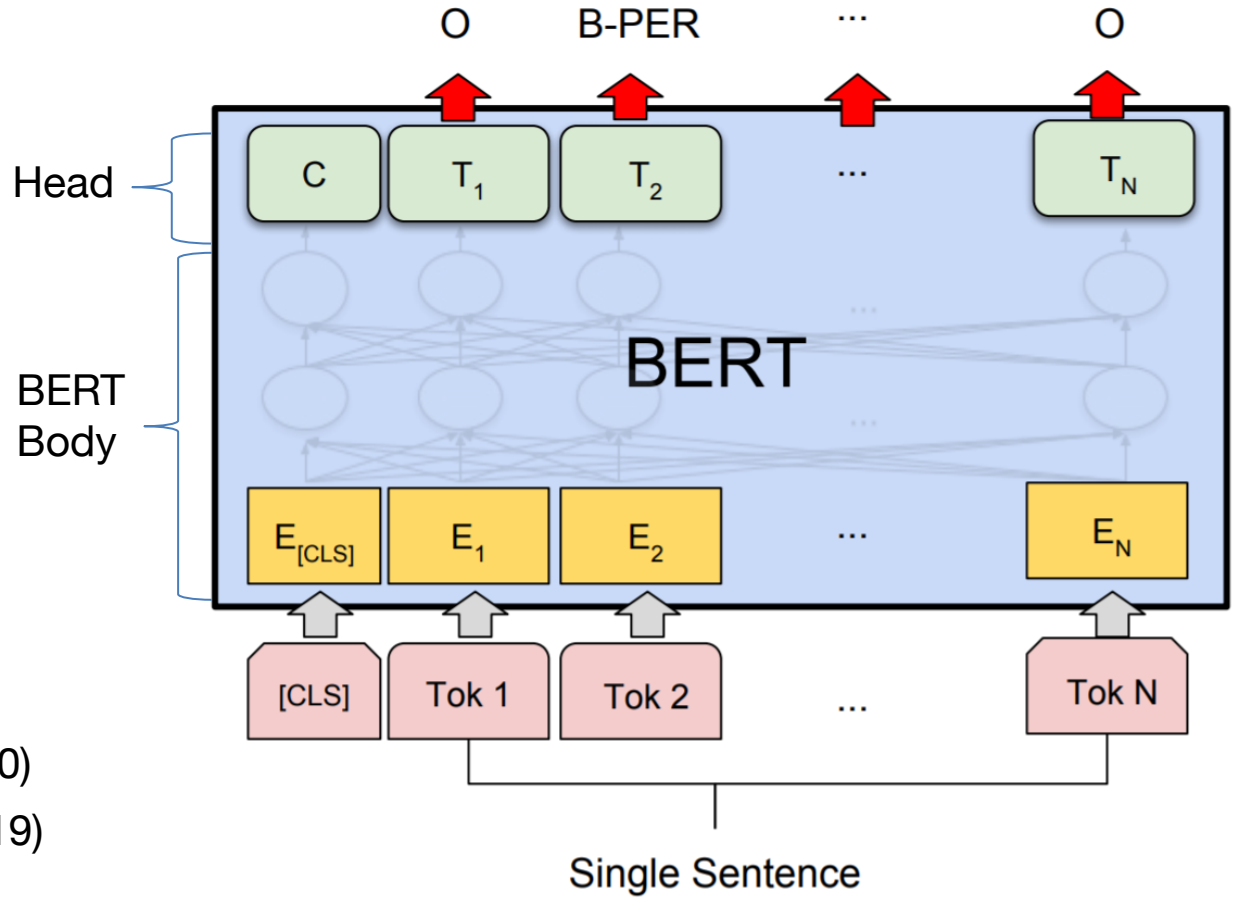




http://jalammar.github.io/illustrated-transformer

BERT (Devlin et al., 2019) for NER

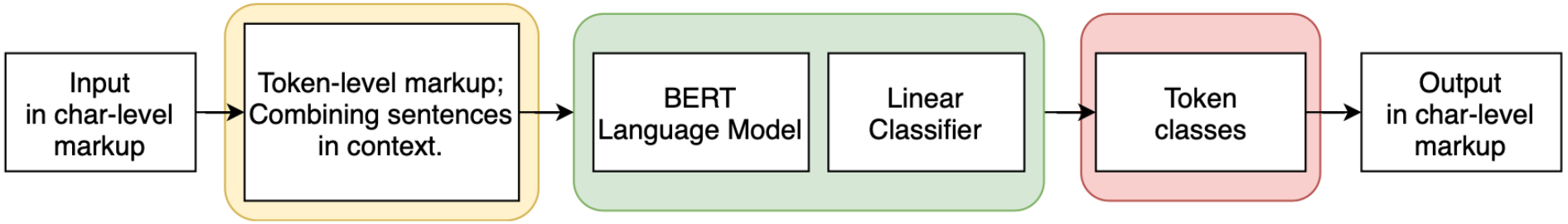
- Training a final fully-connected layer for sequence labelling



Other models:

- ELECTRA (Clark et al., 2020)
- DistilBERT (Sanh et al., 2019)

BERT-based tagger implementation: a linear layer on top of embeddings



class **transformers.BertForMaskedLM** *(config)* [SOURCE]

Bert Model with a *language modeling* head on top.

This model inherits from `PreTrainedModel`. Check the superclass documentation for the generic methods the library implements for all its model (such as downloading or saving, resizing the input embeddings, pruning heads etc.)

This model is also a PyTorch torch.nn.Module subclass. Use it as a regular PyTorch Module and refer to the PyTorch documentation for all matter related to general usage and behavior.

```
class BertForTokenClassification(BertPreTrainedModel):

    authorized_unexpected_keys = [r"pooler"]

    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels

        self.bert = BertModel(config, add_pooling_layer=False)
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        self.classifier = nn.Linear(config.hidden_size, config.num_labels)

        self.init_weights()
```

Summary on Sequence Tagging

- Simplest NER can be done using rule-based approaches
- 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
- MEMM is an alternative that make it easier to incorporate arbitrary features
  - discriminative model: can use any classifier, e.g. MaxEnt, SVM, LR, etc.
  - per-state normalization leads to the label bias problem
- CRF subsumes the advantages of HMMs and MEMMs:
  - per sequence normalization: no label bias problem
  - discriminative: arbitrary features possible
  - training is slow
- Bi-LSTM is a improvment over the CRFs
  - less feature engineering: induce features and representations
  - operates on continuous word representations
  - hyperparameter tuning is costly, makes training even slower
- Transformer-based taggers is the recent state-of-the-art model