CS918: LECTURE 8

Sequence Classification and Part-Of-Speech Tagging

Arkaitz Zubiaga, 29th October, 2018

RECAP: WHAT IS TEXT CLASSIFICATION?

WARWICK

- Having as input:
  - A **text document d**
  - A **set of categories** $C=\{c_1, ..., c_m\}$
- The **text classification** task **outputs**:
  - Predicted class **c*** that **document d belongs to**.

$$c^* \in C$$

RECAP: TEXT CLASSIFICATION APPROACHES

- **Rule-based classifiers**, e.g. if email contains ‘viagra’ → spam
  - Significant **manual effort** involved.

- **Supervised classification**:
  - **Given**: a hand-labeled set of **document-class pairs** $(d_1,c_1), (d_2, c_j), \ldots, (d_m,c_2) \rightarrow$ classified into $C=\{c_1, \ldots, c_j\}$
  - The classifier **learns a model** that can **classify new documents into C**.

RECAP: EVALUATION

- Binary classification (k = 2):
  - Precision, recall and F1 score.

- Multiclass classification (k > 2):
  - Accuracy, ratio of correct predictions, can be problematic when labels are skewed.
  - Better, precision, recall and F1 score per class, then:
    - Micro-averaged evaluation.
    - Macro-averaged evaluation.

RECAP: ERROR ANALYSIS

- **Error analysis:** can help us find out where our classifier can do better.
- No magic formula for performing error analysis.
  - Look **where we are doing wrong**, what labels particularly.
  - Do our errors **have some common characteristics**? Can we infer a new feature from that?
  - Could our **classifier be favouring one of the classes** (e.g. the majority class)?

LECTURE 8: CONTENTS

WARWICK

- Sequence Classification
- Sequence Classifiers:
  - Hidden Markov Models (HMM).
  - Maximum Entropy Markov Models (MEMM).
  - Conditional Random Fields (CRF).
- Using Sequence Classifiers for Part-of-Speech (POS) Tagging.

SEQUENCE CLASSIFICATION

WARWICK

- Sometimes, **classification of items in a sequence is dependent on other sequence items**:
  - **Part-of-Speech (POS) tagging:**
    Assigning **categories to words**, e.g. adjective, noun or verb

    Example:

The	man	saw	a	cat
DET	NN	VB	DET	NN

DET: determiner
NN: noun
VB: verb

SEQUENCE CLASSIFICATION

- **Why** is classification **dependent** on other sequence items?

  In our example, "The man saw a cat":

  'saw' can be:

  a **noun** (if it refers to the tool for cutting)
  a **verb** (if it's simple past of 'see')

  we can't classify whether 'saw' is verb or noun by looking at the word alone → need to look at context, the sequence

SEQUENCE CLASSIFICATION

The	man	saw	a	cat
???	???	???	???	???

- The first item, **‘the’, is easy to classify**, it’s always a determiner:

The	man	saw	a	cat
DET	???	???	???	???

SEQUENCE CLASSIFICATION

The	man	saw	a	cat
DET	???	???	???	???

- Now **'man' is ambiguous:** (1) **noun**, i.e. male person, or (2) **verb**, i.e. 'take charge of'? We can look at:
  - **P('man$_{NN}$') vs P('man$_{VB}$')** → probability of the word as noun or verb
  - **P(NN | DET) vs P(VB | DET)** → probability of a noun or a verb to be preceded by a determiner

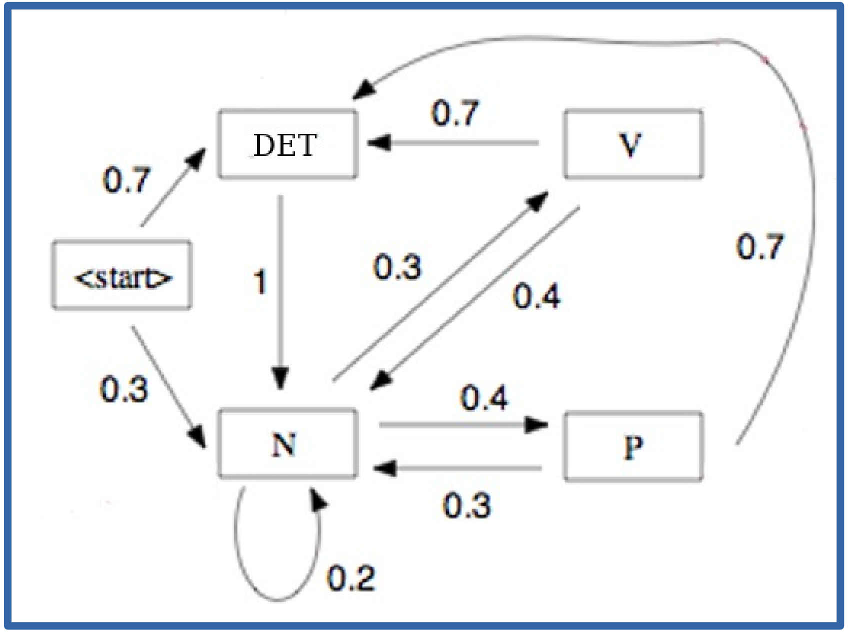
SEQUENCE CLASSIFICATION

WARWICK

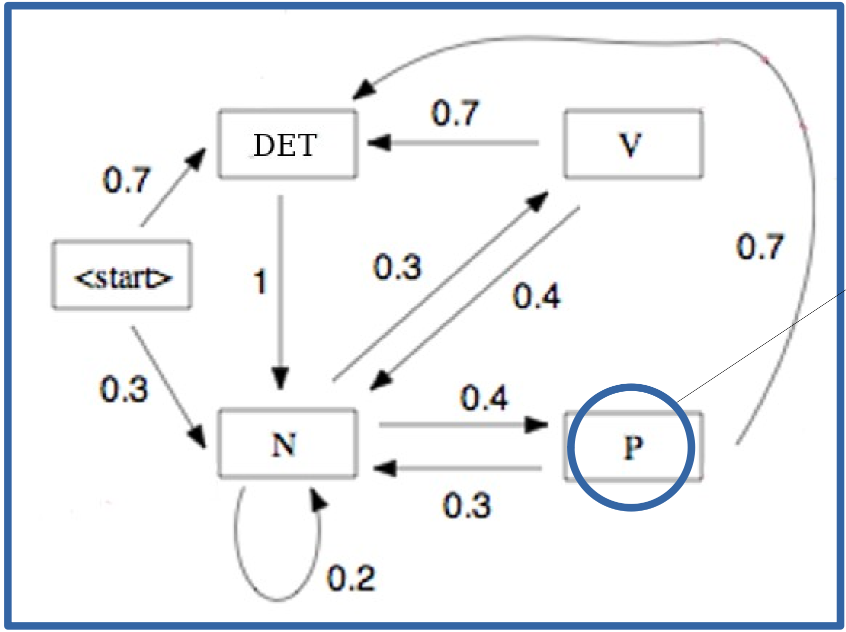
- **Two probabilities** to determine POS of a word:
  [1] P('manNN') vs P('manVB')
  & [2] P('manNN' | DET) vs P('manVB' | DET)

- Classifiers we have seen so far (NB, MaxEnt, SVM) can handle [1].
  - But they can't handle [2].

- Using training data, we can learn of word category POS_i being preceded by POS_j.

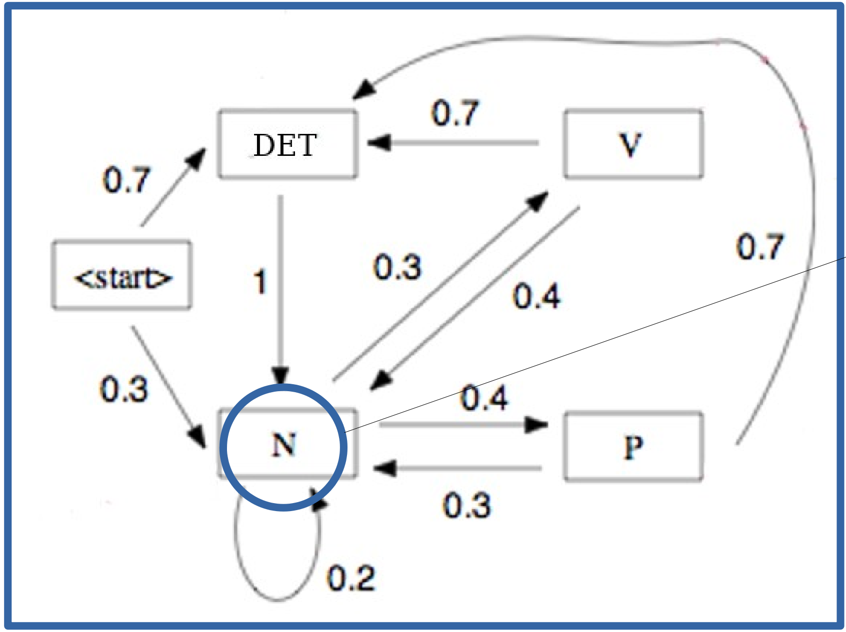


SEQUENCE CLASSIFICATION



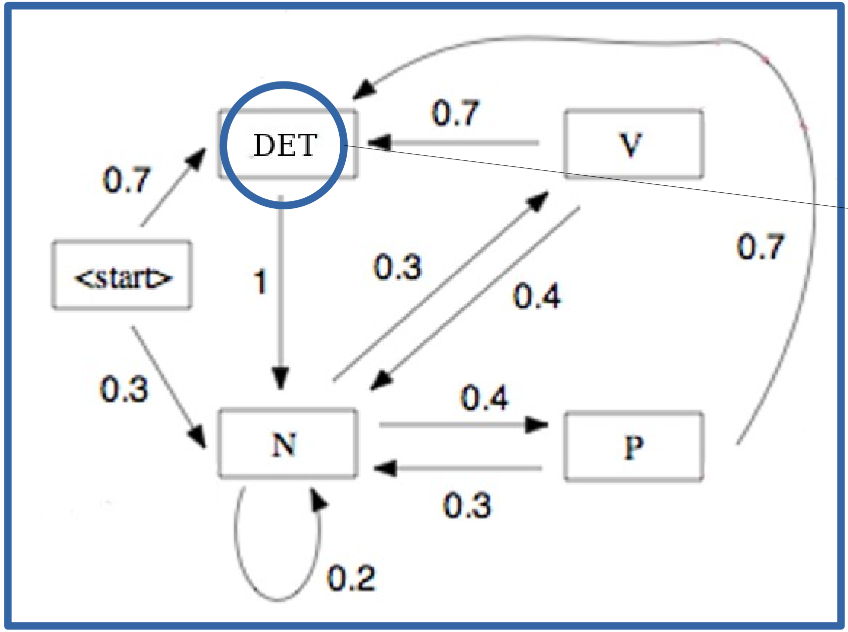
Prepositions can be followed by nouns (.7) or determiners (.3), never by verbs.

SEQUENCE CLASSIFICATION



Nouns can be followed by verbs, prepositions, or another noun.

SEQUENCE CLASSIFICATION



Determiners are ALWAYS followed by a noun.

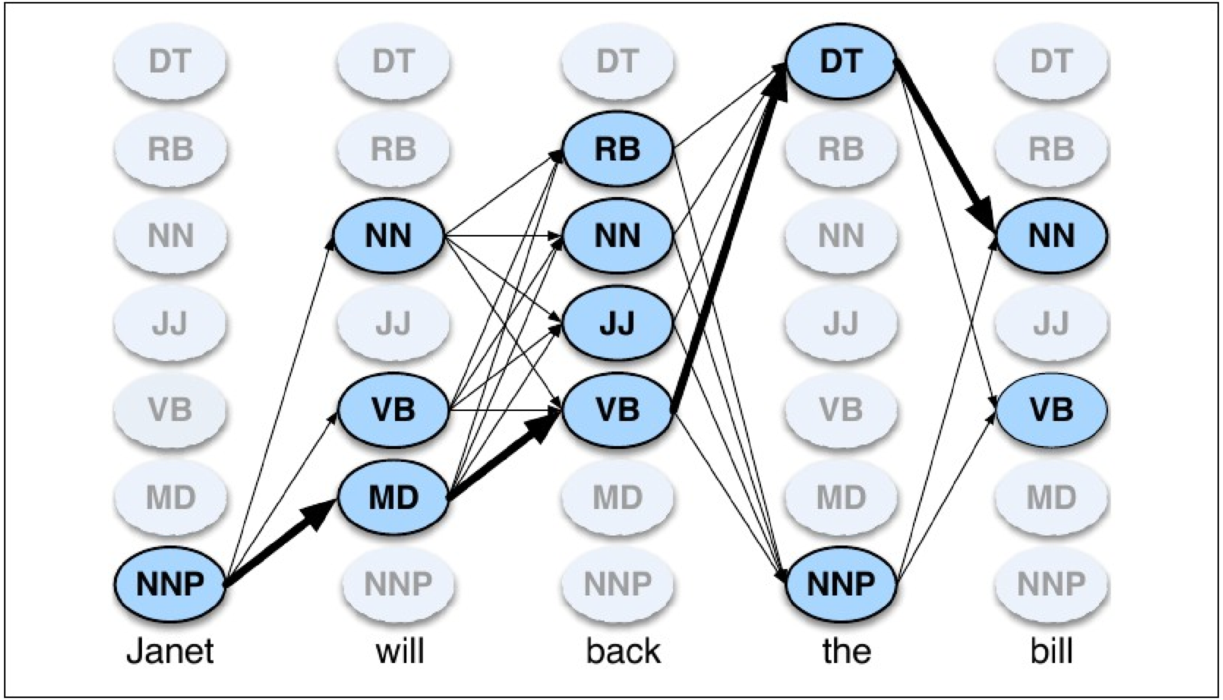
[1] P('manNN') vs P('manVB')

[2] P(NN | DET) vs P(VB | DET) → 0

The	man	saw	a	cat
DET	???	???	???	???

No matter the outcome of [1], we know it's NN thanks to [2].

FINDING THE MOST LIKELY SEQUENCE



SEQUENCE CLASSIFICATION

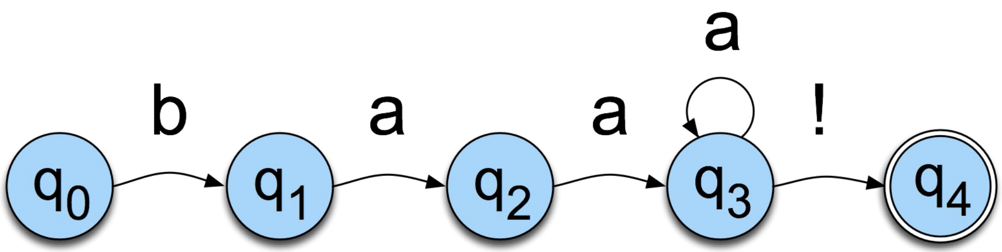
- OK, and how do we achieve this?
  - Hidden Markov Models (HMM)
  - Maximum Entropy Markov Models (MEMM)
  - Conditional Random Fields (CRF)
  - Deep Learning:
    - Recurrent Neural Networks (RNN).
    - Long/Short-Term Memory Networks (LSTM).
    - Convolutional Neural Networks (CNN).

- OK, and how do we achieve this?
  - **Hidden Markov Models (HMM)**
  - **Maximum Entropy Markov Models (MEMM)**
  - **Conditional Random Fields (CRF)**
  - Deep Learning:
    - Recurrent Neural Networks (RNN).
    - Long/Short-Term Memory Networks (LSTM).
    - Convolutional Neural Networks (CNN).

HIDDEN MARKOV MODELS

WHAT IS A MARKOV CHAIN?

- A Markov chain:
  - Set of **nodes connected with probabilities**, where weights on all edges leaving a node sum to 1.



MARKOV CHAINS

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations

MARKOV CHAINS

- In a First order Markov Chain, the probability of a particular state depends ONLY on the previous state.

$$\textbf{Markov Assumption:} \quad P(q_i|q_1 \ldots q_{i-1}) \approx P(q_i|q_{i-1})$$

- **Remember:** Lecture 3 on language models, Markov assumption: P(library | I found two pounds in the) ≈ P(library | the)

WHAT IS A HIDDEN MARKOV MODEL (HMM)?

WARWICK

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	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
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

HMM: TWO KINDS OF PROBABILITIES

- **Transition probabilities:** pairwise probabilities of tags being preceded/followed by another tag.
  e.g. determiner likely followed by noun or adjective.

- **Emission probabilities:** probability for a particular tag to be a particular word, e.g. verb is very likely to be “is”.

TRANSITION PROBABILITIES: EXAMPLE

	NNP	MD	VB	JJ	NN	RB	DT
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

EMISSION PROBABILITIES: EXAMPLE

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

WARWICK

TWO IMPORTANT ASSUMPTIONS IN HMM

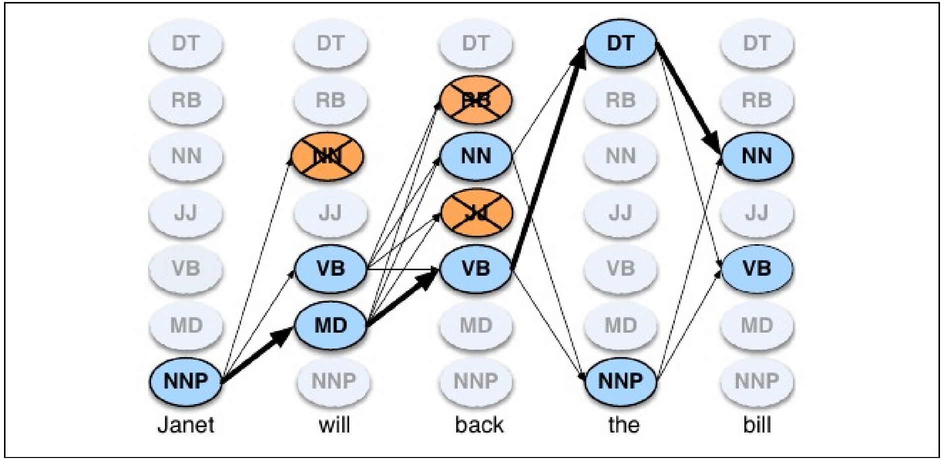
- Again, the Markov assumption, i.e. dependence only on the previous state.

$$\textbf{Markov Assumption:} \quad P(q_i|q_1 \ldots q_{i-1}) \approx P(q_i|q_{i-1})$$

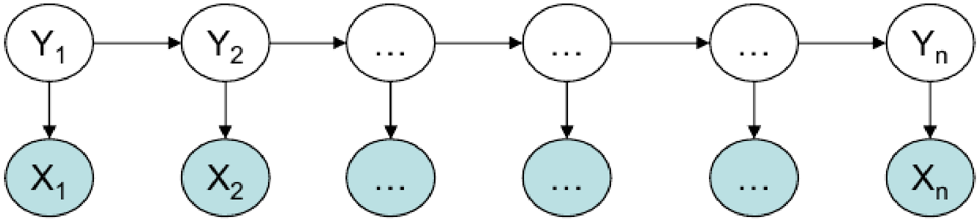
- There must be a dependency between sequence items, e.g.:
  -  Weather (dependency): this morning’s weather may determine the probability of this afternoon’s weather.
  -  Independent events: my laptop having been broken doesn’t determine the probability of my next laptop breaking.

OPTIMISING WITH BEAM SEARCH

- Beam search: only consider top β options for each word.
  - Here β = 2:



LIMITATIONS OF HMM



- Limitations of HMM:
  - Models dependencies between each state and only its corresponding observation.
  - Learns joint distribution of states and observations P(Y, X), but not the conditional probability P(Y|X).

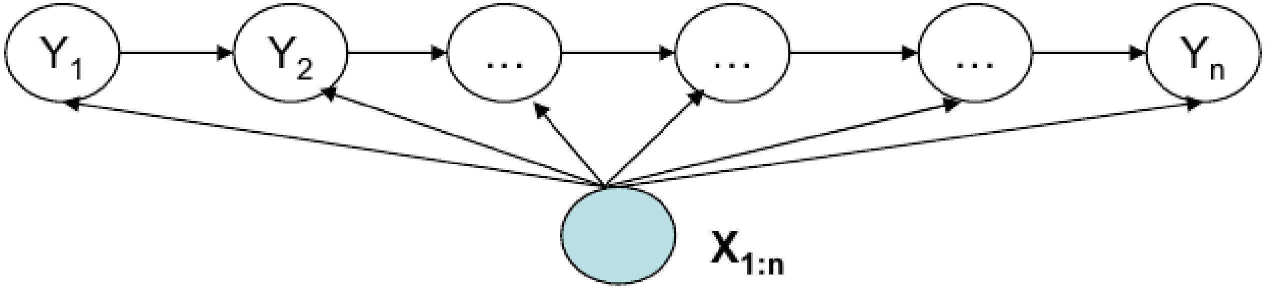
HMM IMPLEMENTATIONS

- NLTK HMM:
  http://www.nltk.org/_modules/nltk/tag/hmm.html

- hmmlearn:
  https://github.com/hmmlearn/hmmlearn

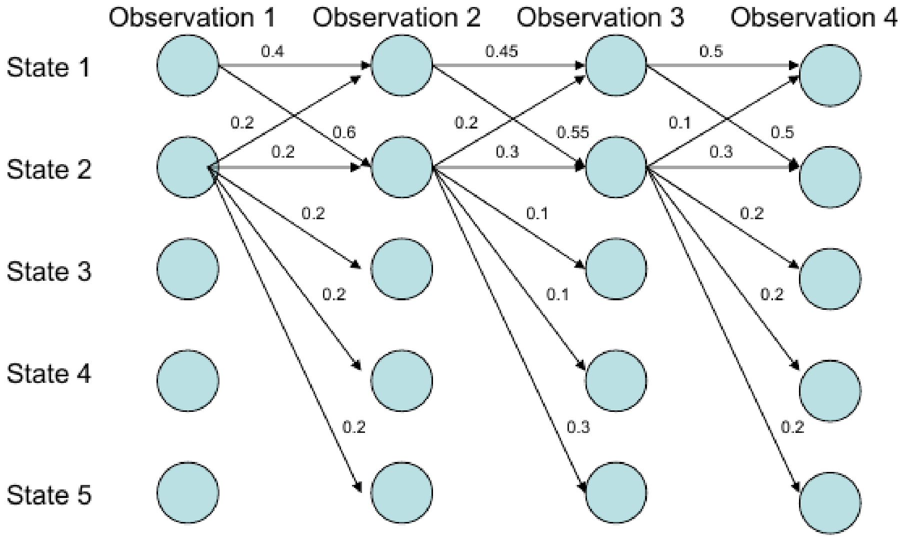
- Scikit HMM:
  http://scikit-learn.sourceforge.net/stable/modules/hmm.html

MEMM: ALTERNATIVE TO HMM

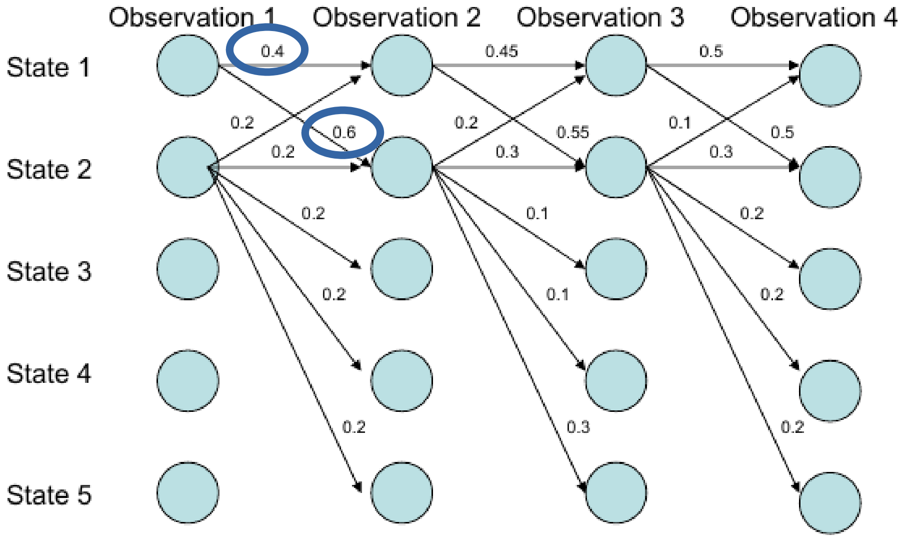


- Maximum Entropy Markov Models (MEMM): Models **dependencies** between **each state and the full observation sequence**.
- Learning objective function consistent with predictive function: P(Y|X).

MEMM: THE LABEL BIAS PROBLEM

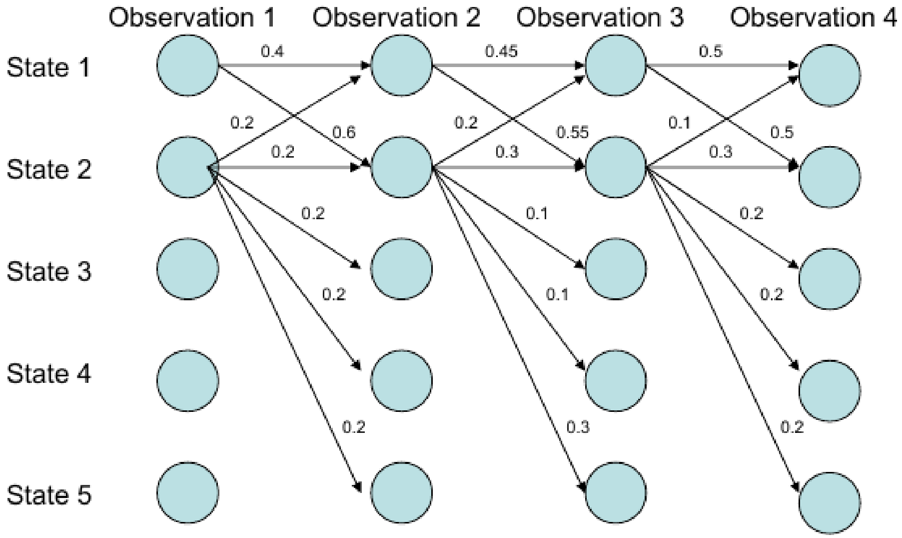


MEMM: THE LABEL BIAS PROBLEM



- In principle, locally, state 1 prefers state 2 in 2nd place.

MEMM: THE LABEL BIAS PROBLEM



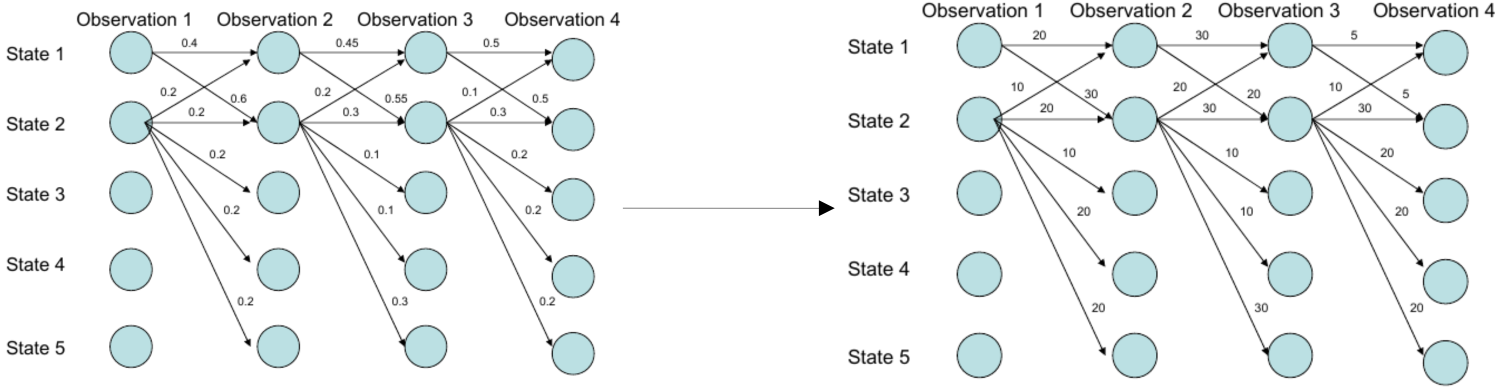
Despite 1→2 being more likely **locally**,

the **entire path probability will favour 1→1**

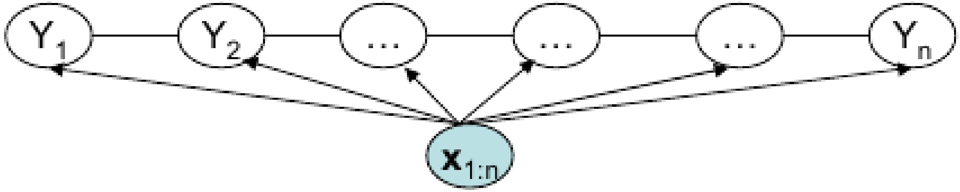
- But if we look at the entire path (which MEMM does):
  - P(1→1→1→1) = 0.4*0.45*0.5 = 0.09
  - P(1→2→2→2) = 0.6*0.3*0.3 = 0.054

SOLUTION TO LABEL BIAS

- Solution: don't normalise probabilities locally, use local potentials.



CONDITIONAL RANDOM FIELDS



$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

- CRF:
  - Global **normaliser Z(x) overcomes label bias** issue of MEMM.
  - Models the **dependency between each state and the entire observation sequence** (like MEMM).

CONDITIONAL RANDOM FIELDS

WARWICK

- Widely used for a range of sequence classification tasks:

  Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
  https://repository.upenn.edu/cgi/viewcontent.cgi?article=1162&context=cis_papers

IMPLEMENTATIONS OF CRF

WARWICK

- Python-crfsuite:
  http://python-crfsuite.readthedocs.io/en/latest/

- PyStruct:
  https://pystruct.github.io/

BEYOND CLASSIFIER LINEAR SEQUENCES

- We’ve talked about classifier linear sequences.

- But sometimes we can have more complex sequences:
  - Graph or tree-structured sequences.

WHERE DO WE FIND TREE SEQUENCES IN NLP?

- Post classification in forums, e.g. binary classification: does a post answer the question of the 1st post?

- Post 1
    - Post 1.1
        - Post 1.1.1
        - Post 1.1.2
    - Post 1.2
        - Post 1.2.1
            - Post 1.2.1.1
        - Post 1.2.2

HMM VS CRF

WARWICK

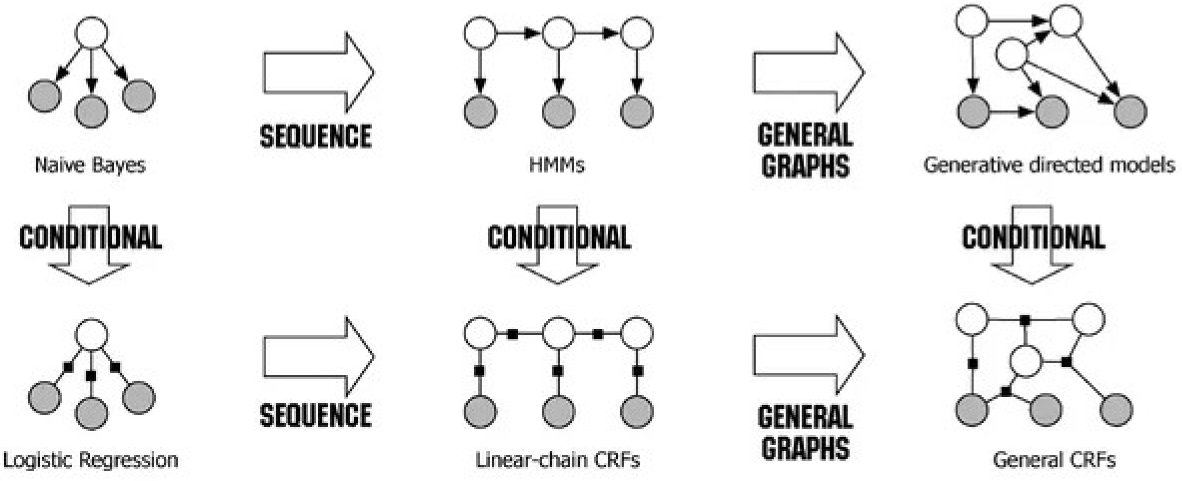


Fig. 2.4 Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

USING SEQUENCE CLASSIFIERS FOR PART-OF-SPEECH (POS) TAGGING

PART-OF-SPEECH TAGGING

WARWICK

- As we were saying, sequence can play an important role in determining POS tags in a sentence:

The	man	saw	a	cat
DET	NN	VB	DET	NN

“man” can’t be a verb if it’s preceded by a determiner.

PART-OF-SPEECH TAGGING

- As we were saying, sequence can play an important role in determining POS tags in a sentence:

The	man	saw	a	cat
DET	NN	VB	DET	NN

- With **HMM**, only looking at previous label, we **can end up predicting sequence of 5 nouns** (NN, NN, NN, NN, NN).

- Looking at the **entire sequence** (MEMM, CRF), we avoid this, **we never have a sentence only made of 5 nouns**.

PART-OF-SPEECH TAGGING

- The list of POS tag types is actually quite large!

Open class (lexical) words

- Nouns
  - Proper: *IBM*, *Italy*
  - Common: *cat / cats*, *snow*
- Verbs
  - Main: *see*, *registered*
  - Modals: *can*, *had*
- Adjectives: *old older oldest*
- Adverbs: *slowly*
- Numbers: *122,312*, *one*
- … more

Closed class (functional)

- Determiners: *the some*
- Conjunctions: *and or*
- Pronouns: *he its*
- Prepositions: *to with*
- Particles: *off up*
- Interjections: *Ow Eh*
- … more

THE PENN TREEBANK TAG SET

WARWICK

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	: ; ... – -

OPEN VS CLOSED CLASSES

- Open vs. Closed classes:
  - **Closed**:
    - **determiners**: a, an, the,...
    - **pronouns**: she, he, I,...
    - **prepositions**: on, under, over, near, by,…
  - **Open**:
    - **Nouns, Verbs, Adjectives, Adverbs.**

CHALLENGES IN POS TAGGING: AMBIGUITY

- Words often have more than one possible POS, e.g. **back**:
  - The **back** door = JJ (adjective)
  - On my **back** = NN (noun)
  - Win the voters **back** = RB (adverb)
  - Promised to **back** the bill = VB (verb)

- See list of POS tags: http://rednoise.org/rita/reference/PennTags.html

PART-OF-SPEECH TAGGING

- POS tagging example:
  - Input: Plays well with others

NNS	UH	IN	NNS
VBZ	JJ		
	NN		
	RB		

1. List candidate labels for each word.
2. Based on probabilities learnt from training data, the classifier predicts the most likely POS sequence.

NB: the fact that there are 2 unambiguous words (with & others) is useful to first label them, and then predict the other 2.

  - Output: Plays/VBZ well/RB with/IN others/NNS

PART-OF-SPEECH TAGGING

- **Uses of POS tagging** in NLP:
  - Text-to-speech.
  - Phrase search through regular expressions, e.g. (Det) Adj* N+
  - As input to or to speed up a full linguistic parser (later lectures)
  - If you know the tag, you can back off to it, e.g. in lemmatisation, saw → see, or saw → saw?
- In other fields:
  - Computational biology.
  - Prediction of series of data, e.g. weather forecasting.

POS TAGGING PERFORMANCE

WARWICK

- Current POS tagging system are very accurate:
  - **Baseline** system that predicts the most common POS for a word already gets **90% accuracy** → thanks to many words not being ambiguous.
  - **Standard classifiers** (no sequence) can achieve **93% accuracy**.
  - **Sequence classifiers** can achieve **97% accuracy**.

POS TAGGING: AMBIGUITY

Types:		WSJ		Brown	
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%**)

REFERENCES

- List of software for POS tagging: https://aclweb.org/aclwiki/Part-of-speech_tagging

ASSOCIATED READING

- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 3rd edition. **Chapters 8.**

- Bird Steven, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O’Reilly Media, Inc., 2009. **Chapter 6 Section 1.6.**