NLP1

Sequence Labelling

Wilker Aziz
w.aziz@uva.nl
probabll.github.io

ILLC

Where are we at?

Week 1

- HC1a: text classification
- HC1b: language modelling

HC2a (today)

- **Sequence labelling**

In NGram LMs Words are Atomic Symbols

We gave words *categorical treatment*, namely, we treated words as if they were completely unrelated to one another. This led to:

- large tabular cpds
- statistical inneficiency (struggles with data sparsity)

Today we try to overcome this in 2 ways:

- a linguistically-motivated change in the data we model, accompanied by a change in the model and new ideas for *factorisation*
- a change in *parameterisation*

Table of contents

Word Categories

Organising Words into Classes

semantic criteria: what does the word refer to?

- nouns often refer to ‘people’, ‘places’ or ‘things’

formal criteria: what form does the word have?

- -ly makes an adverb out of an adjective
- -tion makes a noun out of a verb

distributional criteria: in what contexts can the word occur?

- adjectives precede nouns

Word classes capture aspects of word relatedness.

Examples

	Semantically	Formally	Distributionally
Nouns	refer to things, concepts	-ness, -tion, -ity, -ance	After determiners, possessives
Verbs	refer to actions, states	-ate, -ize	infinitives: to jump, to learn
Adjectives	properties of nouns	-al, -ble	appear before nouns
Adverbs	properties of actions	-ly	next to verbs, beginning of sentence

Why?

Word classes enable a form of delexicalised natural language processing in which we can learn about patterns that are common to all words that share a given property (e.g., in English, a pronoun is typically followed by a verb).

How many classes are there?

This depends on what dimensions of ‘relatedness’ we focus on, and what language we are talking about.

For example, for *Parts-of-Speech* (POS), which mostly capture a word’s syntactic function,

- the English Brown corpus has 87 categories
- the Penn Treebank has 45

Universal POS tags are simplified tags aimed at cross-lingual compatibility (it maps variants of a base class to that base class, e.g., VBD, VBN, VB, VBG, VBP $\rightarrow$ VERB)

Universal Parts-of-Speech (POS)

- ADJ (adjectives)
- ADP (prepositions and postpositions)
- ADV (adverbs)
- CONJ (conjunctions)
- DET (determiners and articles)
- NOUN (nouns)
- NUM (numerals)
- PRON (pronouns)
- PRT (particles)
- PUNCT (punctuation marks)
- VERB (verbs)
- X (anything else, such as abbreviations or foreign words)

Example of POS-Tagged Data (PennTreebank-style)

The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN** a/**DT** number/**NN** of/**IN** other/**JJ** topics/**NNS** ./**.**

There/**EX** was/**VBD** still/**JJ** lemonade/**NN** in/**IN** the/**DT** bottle/**NN** ./**.**

Hidden Markov Model

POS-Tagged Data

We will prescribe a **joint distribution** over the space of **texts annotated with their POS tags**.

That is, we will be learning to assign probability to sequence pairs of the kind $(w_{1:\ell}, c_{1:\ell})$, where $w_{1:\ell}$ is a *word sequence* and $c_{1:\ell}$ is the corresponding *POS tag sequence*.

Example: $(\langle\text{a, nice, dog}\rangle, \langle\text{DT, JJ, NN}\rangle)$.

Applications

- Text analysis: annotating text with POS tags (e.g., input to other tools)
- Language modelling: address some limitations of NGram LMs
- Also, the ideas we develop now will prove useful in many labelling tasks

Formalisation

W is a random word. An outcome w is a symbol in a vocabulary $\mathcal{W}$ of size V.

C is a random POS tag. An outcome c is a symbol in the tagset $\mathcal{C}$ of size K.

$X = \langle W_1, \ldots, W_L \rangle$ is a random word sequence. An outcome $w_{1:\ell}$ is a sequence of ℓ words from $\mathcal{W}$.

$Y = \langle C_1, \ldots, C_L \rangle$ is a random tag sequence. An outcome $c_{1:\ell}$ is a sequence of ℓ tags from $\mathcal{C}$.

Statistical Task

Design a mechanism to assign probability $P_{XY}(w_{1:\ell}, c_{1:\ell})$ to any outcome $(w_{1:l}, c_{1:\ell}) \in \mathcal{W}^* \times \mathcal{C}^*$.

- factorise $P_{XY}(w_{1:\ell}, c_{1:\ell})$
  e.g., chain rule, conditional independencies
- parameterise its elementary factors
  e.g., tabular Categorical cpds

Estimate the parameters of this mechanism from data (i.e., text annotated with POS tags).

- e.g., use MLE to estimate the free parameters of our parameterisation

NLP Tasks

Predict a POS tag sequence for a given text. For example, via mode-seeking search:

$$\arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} \; P_{Y|X}(c_{1:\ell}|w_{1:\ell})$$

Assign probability to text that is **not** annotated with POS tags, via marginalisation:

$$P_X(w_{1:\ell}) = \sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

The outcome assigned largest probability mass is known as the *mode* of the probability distribution.

Let's get started – Factorisation

Challenge. P_{XY} is a distribution over a countably infinite space of sequence pairs.

Key Idea. Re-express the probability of a sequence pair using the probabilities of the "steps" needed to generate it. Design steps such that they have a simple, countably finite sample space.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story

BoS

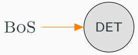
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



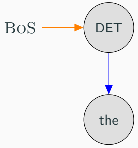
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



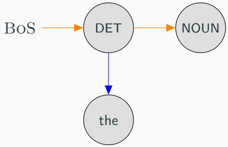
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



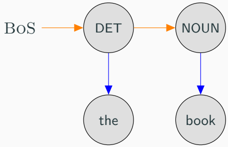
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



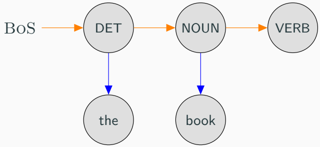
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



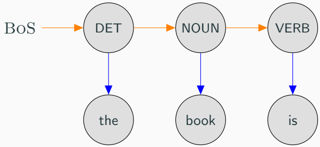
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



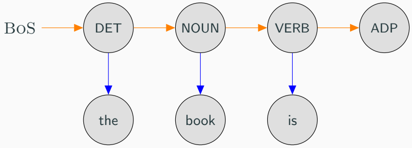
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



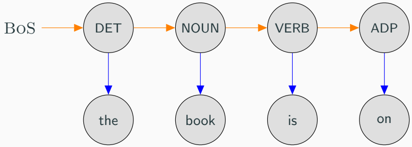
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



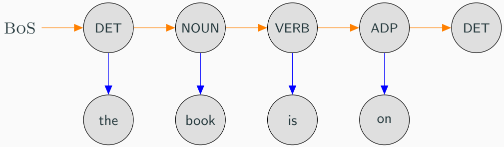
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



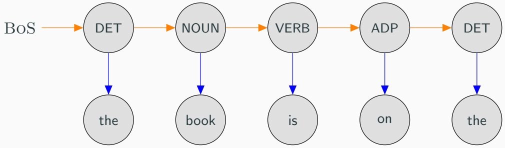
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



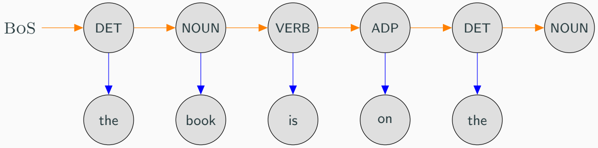
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



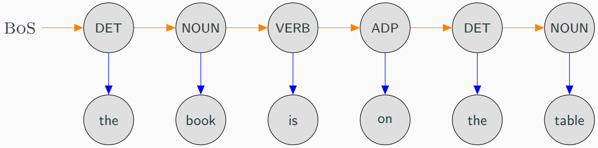
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



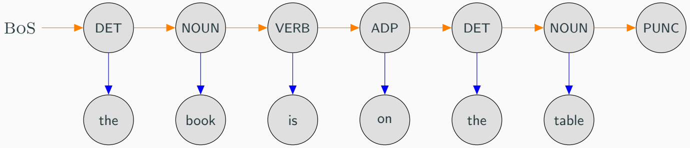
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



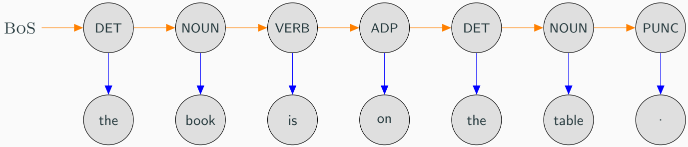
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



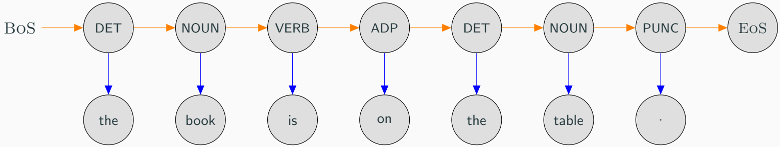
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



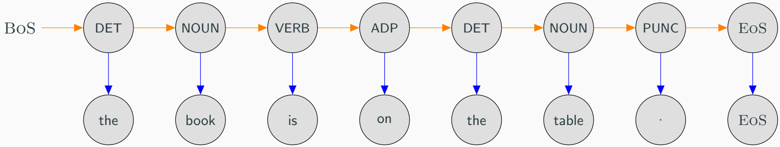
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



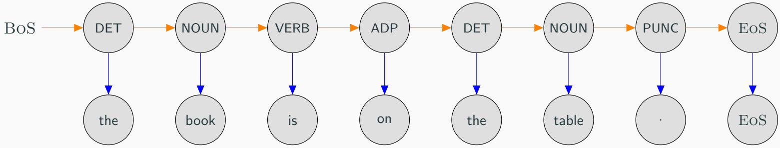
We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Modelling POS-tagged data: illustration

Joint observations

the/DET book/NOUN is/VERB on/ADP the/DET table/NOUN ./PUNC

Generative story



Joint probability

$$P_{C|C_{\text{prev}}}(\text{DET}|\text{BoS})P_{W|C}(\text{the}|\text{DET})$$
$$\times P_{C|C_{\text{prev}}}(\text{NOUN}|\text{DET})P_{W|C}(\text{book}|\text{NOUN})$$
$$\times \ldots$$
$$\times P_{C|C_{\text{prev}}}(\text{EoS}|\text{PUNC})P_{W|C}(\text{EoS}|\text{EoS})$$

We pad the tag sequence with a BoS symbol. We pad both sequences with a EoS symbol.

Chain Rule for the HMM

Conditional independences

- W_i is independent of all but C_i;
- C_i is independent of all but C_{i-1}.

Leading to

$$P_{XY}(w_{1:\ell}, c_{1:\ell}) \stackrel{\text{ind.}}{=} \prod_{i=1}^{\ell} \underbrace{P_{C|C_{\text{prev}}}(c_i|c_{i-1})}_{\text{transition}} \underbrace{P_{W|C}(w_i|c_i)}_{\text{emission}} \tag{1}$$

Hint. Pad the sequences with a special BOS token (or tag) and a special EOS token (or tag).

Generative Story

1. Start with $X = \langle W_0 = \text{BOS} \rangle$, $Y = \langle C_0 = \text{BOS} \rangle$ and set $i = 1$;
2. Condition on the previous class c_{i-1} and draw a class c_i with probability $P_{C|C_{\text{prev}}}(c_i|c_{i-1})$ extending Y with it;
3. Condition on the current class c_i and draw a word w_i with probability $P_{W|C}(w_i|c_i)$ extending X with it;
4. If w_i is a special end-of-sequence symbol (EOS), terminate, else increment i and repeat from (2).

This specifies a **factorisation** of P_{XY} in terms of elementary factors of the kind $P_{C|C_{\text{prev}}}$ and $P_{W|C}$.

Tabular Parameterisation

Transition distributions. Given a *previous* tag r, the transition distribution over (next) tags is Categorical:

$$C|C_{\text{prev}} = r \sim \mathrm{Categorical}(\lambda_{1:K}^{(r)})$$

Tabular Parameterisation

Transition distributions. Given a *previous* tag r, the transition distribution over (next) tags is Categorical:

$$C|C_{\text{prev}} = r \sim \mathrm{Categorical}(\lambda_{1:K}^{(r)}) \qquad \text{hence, } P_{C|C_{\text{prev}}}(c|r) = \lambda_c^{(r)}$$

Tabular Parameterisation

Transition distributions. Given a *previous* tag r, the transition distribution over (next) tags is Categorical:

$$C|C_{\text{prev}} = r \sim \text{Categorical}(\lambda_{1:K}^{(r)}) \qquad \text{hence, } P_{C|C_{\text{prev}}}(c|r) = \lambda_c^{(r)}$$

Emission distribution. Given a tag c, the emission distribution over words is also Categorical:

$$W|C = c \sim \text{Categorical}(\theta_{1:V}^{(c)})$$

Tabular Parameterisation

Transition distributions. Given a *previous* tag r, the transition distribution over (next) tags is Categorical:

$$C|C_{\text{prev}} = r \sim \text{Categorical}(\lambda_{1:K}^{(r)}) \qquad \text{hence, } P_{C|C_{\text{prev}}}(c|r) = \lambda_c^{(r)}$$

Emission distribution. Given a tag c, the emission distribution over words is also Categorical:

$$W|C = c \sim \text{Categorical}(\theta_{1:V}^{(c)}) \qquad \text{hence, } P_{W|C}(w|c) = \theta_w^{(c)}$$

Probability mass function (pmf).

$$P_{XY}(w_{1:\ell}, c_{1:\ell}) = \prod_{i=1}^{\ell} \underbrace{\lambda_{c_i}^{(c_{i-1})}}_{\text{transition}} \times \underbrace{\theta_{w_i}^{(c_i)}}_{\text{emission}}$$

Example

For *a/DT nice/JJ dog/NN*, we have probability mass:

$$\lambda_{\text{DT}}^{(\text{BOS})}\theta_{\text{a}}^{(\text{DT})}\lambda_{\text{JJ}}^{(\text{DT})}\theta_{\text{nice}}^{(\text{JJ})}\lambda_{\text{NN}}^{(\text{JJ})}\theta_{\text{dog}}^{(\text{NN})}\lambda_{\text{EOS}}^{(\text{NN})}\theta_{\text{EOS}}^{(\text{EOS})}$$

Parameter Estimation via MLE

Given a dataset of observed texts annotated with POS, the maximum likelihood estimate of:

- Transition. The conditional probability $P_{C|C_{\text{prev}}}(c|r)$ of generating a tag c right after having generated a tag r is

$$\lambda_c^{(r)} \overset{\text{MLE}}{=} \frac{\mathrm{count}_{C_{\text{prev}}C}(r, c)}{\sum_{k=1}^{K} \mathrm{count}_{C_{\text{prev}}C}(r, k)} = \frac{\mathrm{count}_{C_{\text{prev}}C}(r, c)}{\mathrm{count}_{C_{\text{prev}}}(r)}$$

- Emission. The conditional probability $P_{W|C}(w|c)$ of generating word w from tag c is

$$\theta_w^{(c)} \overset{\text{MLE}}{=} \frac{\mathrm{count}_{CW}(c, w)}{\sum_{o=1}^{V} \mathrm{count}_{CW}(c, o)} = \frac{\mathrm{count}_{CW}(c, w)}{\mathrm{count}_{C}(c)}$$

Data Sparsity

It's still possible that this model suffers from data sparsity (e.g., unseen word-tag pairs or unseen tag-tag pairs), but much less so than an NGram LM: contextual information is only available through the POS tag of the previous position (K possible outcomes, instead of V^{N-1} outcomes).

Strong Conditional Independence Assumptions

PLAN as a verb (I read that the government plans to ...) or noun (I read the government plans to ...)

- older history (read that vs. read the) affects the analysis

HER as possessive determiner (I read *her* book) or personal pronoun (I saw *her* there).

- the (semantics of the) verb (to read vs. to see) affects the analysis

LIKE as verb (Children like to play outside) or preposition (Children like their parents need support).

- analysing *like* requires looking ahead of it

Agreement features cannot always be delexicalised: a cat vs a cats.

Possible Improvements

Relax some independencies, e.g.

- have C_i depend on (C_{i-2}, C_{i-1});
- have W_i depend on C_{i-1}, or W_{i-1}, etc.

These ideas can lead to better models, but tabular representations become larger (and sparser) and they lead to other problems (as we will see next).

Evaluation

Tagging Performance

Predict a POS tag sequence for novel text. For example via mode-seeking search:

$$\hat{c}_{1:\ell} = \underset{c_{1:\ell} \in \mathcal{C}^\ell}{\arg\max} \; P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

Compare predicted $\hat{c}_{1:\ell}$ to human-annotated $c^\star_{1:\ell}$ step by step: assess the rate at which the ith prediction matches the ith target (accuracy). Or, since POS categories are likely imbalanced, compute per-POS F_1 and report macro (or weighted) average.

Search

Let's understand what it means to solve this expression

$$\hat{c}_{1:\ell} = \arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

the	cute	cat
c_1	c_2	c_3

Search

Let's understand what it means to solve this expression

$$\hat{c}_{1:\ell} = \arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

1. Enumerate *all candidate tag sequences*

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Search

Let's understand what it means to solve this expression

$$\hat{c}_{1:\ell} = \arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Search

Let's understand what it means to solve this expression

$$\hat{c}_{1:\ell} = \arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate
3. Sort by probability and pick the best

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Search

Let's understand what it means to solve this expression

$$\hat{c}_{1:\ell} = \arg\max_{c_{1:\ell} \in \mathcal{C}^\ell} P_{Y|X}(c_{1:\ell} | w_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate
3. Sort by probability and pick the best

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

We have K^ℓ candidates, enumeration is **intractable**!

LM Performance

Use the HMM to assign probability to observed text $w_{1:\ell}$

$$P_X(w_{1:\ell}) = \sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

Use a heldout dataset and the marginal pmf to assess the perplexity of the model.

Marginal

Let's understand what it means to solve this expression

$$\sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

the	cute	cat
c_1	c_2	c_3

Marginal

Let's understand what it means to solve this expression

$$\sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

1. Enumerate *all candidate tag sequences*

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Marginal

Let's understand what it means to solve this expression

$$\sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Marginal

Let's understand what it means to solve this expression

$$\sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate
3. Sum their probabilities

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

Marginal

Let's understand what it means to solve this expression

$$\sum_{c_{1:\ell} \in \mathcal{C}^\ell} P_{XY}(w_{1:\ell}, c_{1:\ell})$$

1. Enumerate *all candidate tag sequences*
2. Assess the probability of each candidate
3. Sum their probabilities

the	cute	cat
c_1	c_2	c_3
D	D	D
D	D	J
D	D	N
D	D	V
D	D	X
D	J	D
	...	

We have K^ℓ candidates, enumeration is **intractable**!

Dynamic Programming

Enumeration is intractable, but, as it turns out, it's unnecessary.

Because of the conditional independences in the HMM, changing the POS tag of position i can only affect

- one emission probability ($\underline{C_i} \to w_i$)
- and two transition probabilities ($C_{i-1} \to \underline{C_i}$ and $\underline{C_i} \to C_{i+1}$).

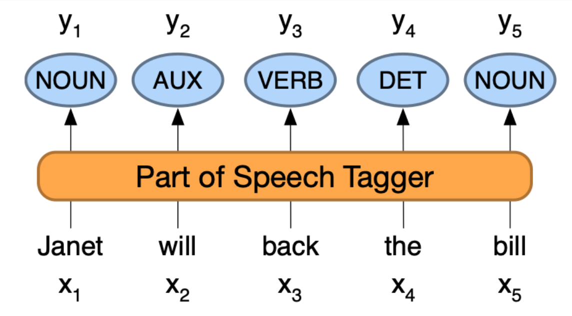
This allows us to solve search and marginalisation incrementally from left to right in time $\mathcal{O}(L \times K^2)$ using the Viterbi or Forward algorithms. Watch the video I prepared for you:

https://youtu.be/rVCd7NrGcSI

Sequence Labelling

POS Tagging

We are **given** the text and we do not care to assign probability to it.



Our goal is to develop a system that can POS tag the input sequence.

NER is a labelling task from a semantic perspective, where we recognise proper nouns that refer to a certain type of entity.

> Citing high fuel prices, [$_{\text{ORG}}$ **United Airlines**] said [$_{\text{TIME}}$ **Friday**] it has increased fares by [$_{\text{MONEY}}$ **\$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{\text{ORG}}$ **American Airlines**], a unit of [$_{\text{ORG}}$ **AMR Corp.**], immediately matched the move, spokesman [$_{\text{PER}}$ **Tim Wagner**] said. [$_{\text{ORG}}$ **United**], a unit of [$_{\text{ORG}}$ **UAL Corp.**], said the increase took effect [$_{\text{TIME}}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{\text{LOC}}$ **Chicago**] to [$_{\text{LOC}}$ **Dallas**] and [$_{\text{LOC}}$ **Denver**] to [$_{\text{LOC}}$ **San Francisco**].

The text (in black) is **given** and we do not care to assign probability to it. Our goal is to develop a system that can detect and categorise mentions to named entities (i.e., the blue spans)

Chunking as Labelling

We can see NER as sequence labelling by labelling tokens as inside or outside a span of text that refers to a named-entity.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Figure 17.7 NER as a sequence model, showing IO, BIO, and BIOES taggings.

These annotation schemes fit right into the sequence labelling framework we developed for POS tagging.

Key Technical Limitation

Because HMMs need to generate text, they power sequence labellers that make fairly limited use of linguistic context in $w_{1:\ell}$.

Having C_i interact with words other than W_i would make key quantities in the HMM **very** hard to compute (e.g., marginal and mode probabilities). It would also make the tabular CPDs rather sparse.

Limitations from a Linguistic Perspective

Unseen words and phrases (e.g., proper names and acronyms, inflected verbs, phrasal verbs) are actually quite frequent.

In many cases, their likely interpretation (e.g., syntactic or semantic function) are identifiable from fine-grained features: capitalisation (in English), prefixes and suffixes (e.g., ‘un-’ or ‘-ed’), knowing the words surrounding a certain position (e.g., a window of 5 words), etc.

Local Log-Linear Models

Rethinking Factorisation

Sequence labelling tasks map from a token sequence to the tag sequence that's assigned highest probability under the model

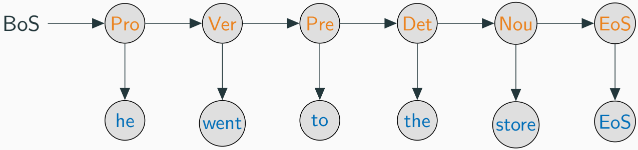
$$\hat{c}_{1:\ell} = \underset{c_{1:\ell} \in \mathcal{C}^\ell}{\arg\max}\ P_{Y|X}(c_{1:\ell}|w_{1:\ell})$$

In an HMM, we obtain this conditional by *inferring* it from a joint distribution (which we design, i.e., factorise and parameterise).

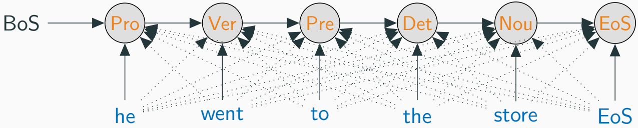
What if, instead, we attempted to factorise and parameterise this conditional *directly*?

Conditional modelling

The HMM is a *generative model* of labelled text.



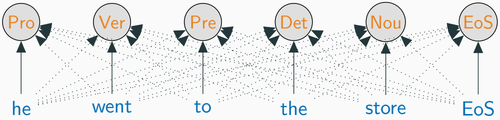
We may choose to regard the text as a predictor, and *model the conditional distribution of tag sequences*.



First Idea: 0-order model

Let's start even simpler and make a 0-order Markov assumption: $C_i \perp C_{j \neq i} | X = x, I = i$.

$$P_{Y|X}(c_{1:\ell}|w_{1:\ell}) \overset{\text{ind.}}{=} \prod_{i=1}^{\ell} P_{C|XI}(c_i|w_{1:\ell}, i) \quad (4)$$

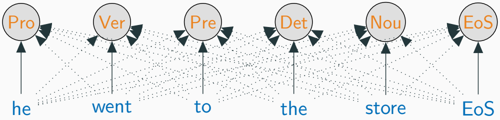


$C_i \perp C_{j \neq i} | X = x, I = i$ is pronounced: given x and that we want to tag the ith word, the distribution of the ith tag C_i is independent of the rest of the tag sequence.

First Idea: 0-order model

Let's start even simpler and make a 0-order Markov assumption: $C_i \perp C_{j \neq i} | X = x, I = i$.

$$P_{Y|X}(c_{1:\ell}|w_{1:\ell}) \overset{\text{ind.}}{=} \prod_{i=1}^{\ell} P_{C|XI}(c_i|w_{1:\ell}, i) \quad (4)$$



To make this happen, we will need to rethink *parameterisation*!

$C_i \perp C_{j \neq i} | X = x, I = i$ is pronounced: given x and that we want to tag the ith word, the distribution of the ith tag C_i is independent of the rest of the tag sequence.

Rethinking Parameterisation

In the 0-order conditional model, the cpd of any one tag depends on the entire text $w_{1:\ell}$, for each position $i \in [\ell]$.

$$C \mid \underbrace{X = w_{i:\ell}, I = i}_{\text{conditioning context}} \sim \text{Categorical}(\underbrace{\square_1, \ldots, \square_K}_{\text{conditional probs}})$$

Since the conditioning context is a high-dimensional, variable-length outcome, we cannot give this cpd tabular treatment (i.e., store conditional probs for every $(w_{1:\ell}, i)$).

Instead we can **learn to predict** **conditional probs** from a D-dimensional representation of the conditioning context.

Note how this parallels the design of a text classifier: in a given textual context we want to predict a distribution over K labels.

Logistic CPDs

Let $\phi(w_{1:\ell}, i) \in \mathbb{R}^D$ be a feature vector representing ('describing') the ith position of $w_{1:\ell}$.

Examples:

- $\phi(\langle \text{he, went, \underline{to}, the, store} \rangle, 3)$ is a vector $\mathbf{u}$ such that $u_{\text{id(word:to)}} = 1$, $u_{\text{id(before:went)}} = 1$, $u_{\text{id(after:the)}} = 1$, $u_{\text{id(position)}} = {}^3\!/\!_5$, and other coordinates of $\mathbf{u}$ are 0;

Logistic CPDs

Let $\phi(w_{1:\ell}, i) \in \mathbb{R}^D$ be a feature vector representing ('describing') the ith position of $w_{1:\ell}$.

Examples:

- $\phi(\langle \text{he, went, } \underline{\text{to}}\text{, the, store} \rangle, 3)$ is a vector $\mathbf{u}$ such that $u_{\text{id(word:to)}} = 1$, $u_{\text{id(before:went)}} = 1$, $u_{\text{id(after:the)}} = 1$, $u_{\text{id(position)}} = {}^3\!/_5$, and other coordinates of $\mathbf{u}$ are 0;
- $\phi(\langle \text{he, went, to, the, } \underline{\text{store}} \rangle, 5)$ is a vector $\mathbf{v}$ such that $v_{\text{id(word:store)}} = 1$, $v_{\text{id(before:the)}} = 1$, $v_{\text{id(after:EOS)}} = 1$, $v_{\text{id(position)}} = {}^{35}\!/_5$, and other coordinates of $\mathbf{v}$ are 0;

Logistic CPDs

Let $\phi(w_{1:\ell}, i) \in \mathbb{R}^D$ be a feature vector representing ('describing') the ith position of $w_{1:\ell}$.

Examples:

- $\phi(\langle \text{he, went, } \underline{\text{to}}\text{, the, store} \rangle, 3)$ is a vector $\mathbf{u}$ such that $u_{\text{id(word:to)}} = 1$, $u_{\text{id(before:went)}} = 1$, $u_{\text{id(after:the)}} = 1$, $u_{\text{id(position)}} = 3/5$, and other coordinates of $\mathbf{u}$ are 0;
- $\phi(\langle \text{he, went, to, the, } \underline{\text{store}} \rangle, 5)$ is a vector $\mathbf{v}$ such that $v_{\text{id(word:store)}} = 1$, $v_{\text{id(before:the)}} = 1$, $v_{\text{id(after:EOS)}} = 1$, $v_{\text{id(position)}} = 35/5$, and other coordinates of $\mathbf{v}$ are 0;

We map it to a K-dimensional probability vector via:

$$\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta}) = \text{softmax}(\mathbf{W}\phi(w_{1:\ell}, i) + \mathbf{b})$$

with $\boldsymbol{\theta} = \{\mathbf{W} \in \mathbb{R}^{K \times D}, \mathbf{b} \in \mathbb{R}^K\}$.

Logistic CPDs

Let $\phi(w_{1:\ell}, i) \in \mathbb{R}^D$ be a feature vector representing ('describing') the ith position of $w_{1:\ell}$.

Examples:

- $\phi(\langle$he, went, <u>to</u>, the, store$\rangle, 3)$ is a vector $\mathbf{u}$ such that $u_{\text{id(word:to)}} = 1$, $u_{\text{id(before:went)}} = 1$, $u_{\text{id(after:the)}} = 1$, $u_{\text{id(position)}} = {}^3\!/_5$, and other coordinates of $\mathbf{u}$ are 0;
- $\phi(\langle$he, went, to, the, <u>store</u>$\rangle, 5)$ is a vector $\mathbf{v}$ such that $v_{\text{id(word:store)}} = 1$, $v_{\text{id(before:the)}} = 1$, $v_{\text{id(after:EOS)}} = 1$, $v_{\text{id(position)}} = {}^{35}\!/_5$, and other coordinates of $\mathbf{v}$ are 0;

We map it to a K-dimensional probability vector via:

$$\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta}) = \text{softmax}(\mathbf{W}\phi(w_{1:\ell}, i) + \mathbf{b})$$

with $\boldsymbol{\theta} = \{\mathbf{W} \in \mathbb{R}^{K \times D}, \mathbf{b} \in \mathbb{R}^K\}$.

Exercise (for after class)

Here are templates for the coordinates of a feature function $\phi(w_{1:\ell}, i)$: word:w_i, before:w_{i-1}, after:w_{i+1}, all binary-valued. If we know a total of V words, what is the dimensionality D of this feature space?

What if we add the templates before2:w_{i-2} and after2:w_{i+2}?

And what if we add the templates wordpref:prefix(w_i), wordsuff:suffix(w_i) and wordstem:stem(w_i)? Assume we know R prefixes, S suffixes, and a number $V' \propto V$ of stems.

For example, prefix(unwanted) = un, stem(unwanted) = want, suffix(unwanted) = ed.

Pmf

For $i \in [\ell]$, here's our model

$$C \mid \underbrace{X = w_{i:\ell}, I = i}_{\text{conditioning context}} \sim \text{Categorical}(\underbrace{\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta})}_{\text{predicted probs}})$$

Here's the pmf

$$\begin{aligned} P_{Y|X}(c_{1:\ell}|w_{1:\ell}) &= \prod_{i=1}^{\ell} P_{C|XI}(c_i|w_{1:\ell}, i) \\ &= \prod_{i=1}^{\ell} [\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta})]_{c_i} \end{aligned} \tag{5}$$

The notation $[\mathbf{v}]_i$ is equivalent to v_i, it's easier to read when the vector argument is the output of a function with multiple arguments.

Pmf

For $i \in [\ell]$, here's our model

$$C \mid \underbrace{X = w_{i:\ell}, I = i}_{\text{conditioning context}} \sim \text{Categorical}(\underbrace{\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta})}_{\text{predicted probs}})$$

Here's the pmf

$$\begin{aligned} P_{Y|X}(c_{1:\ell}|w_{1:\ell}) &= \prod_{i=1}^{\ell} P_{C|XI}(c_i|w_{1:\ell}, i) \\ &= \prod_{i=1}^{\ell} [\mathbf{f}(w_{1:\ell}, i; \boldsymbol{\theta})]_{c_i} \end{aligned} \tag{5}$$

Note how this amounts to designing 1 'text classifier'-type thing and re-using it for each and every step of the sequence.

The notation $[\mathbf{v}]_i$ is equivalent to v_i, it's easier to read when the vector argument is the output of a function with multiple arguments.

Exercise (for after class)

About the 0-order model we designed.

Claim 1: it cannot be used as a language model (i.e., it cannot assign probability to a given piece of text).

Claim 2: finding the best tag sequence for a given text $w_{1:\ell}$ can be done by solving a sequence of ℓ independent classifications.

Accept or reject the claims, justifying your decision in each case.

Likelihood Function

Just like before, we do MLE. Unlike before, this time we will not have an exact, closed-form expression.

We have a training corpus $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N data points, the nth of which is a labelled sequence of length ℓ_n.

We assess the log-likelihood of $\boldsymbol{\theta}$ given $\mathcal{D}$

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} \log P_{Y|X}(y_n|x_n) \\ &= \sum_{n=1}^{N} \sum_{i=1}^{\ell_n} \log \underbrace{P_{C|XI}(y_{n,i}|x_n, i)}_{=[\mathbf{f}(x_n,i;\boldsymbol{\theta})]_{c_{n,i}}}\end{aligned} \tag{6}$$

The technical term from statistics is *likelihood of model parameter given observed data*, in ML and applied ML, esp in recent years, $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ is often referred to as 'likelihood of data'.

Gradient-Based Optimisation

We search for the parameter value that optimises the log-likelihood function:

$$\boldsymbol{\theta}^{\mathsf{MLE}} = \arg\max_{\boldsymbol{\theta}} \; \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) \tag{7}$$

There's no closed-form solution to this optimisation problem (for our log-linear model), but we can approximately solve it via an iterative gradient-based optimisation

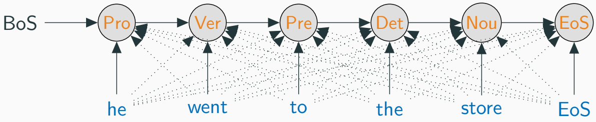
$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \gamma \boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) \tag{8}$$

We typically use autograd [**?**] and a stochastic optimiser for this [**?**].

More Dependencies: 1-order model

We could make a strong 1-order Markov assumption:

$$P_{Y|X}(c_{1:\ell}|w_{1:\ell}) \triangleq \prod_{i=1}^{\ell} P_{C|XIC_{\text{prev}}}(c_i|w_{1:\ell}, i, c_{i-1}) \qquad (9)$$



A feature function for this has access to the *previous class*: e.g., $\phi(\langle \text{he, went, \underline{to}, the, store} \rangle, 3, \text{Verb})$ is a vector $\mathbf{u}$ such that $u_{\text{id(word:to)}} = 1$, $u_{\text{id(before:went)}} = 1$, $u_{\text{id(after:the)}} = 1$, $u_{\text{id(position)}} = {}^3\!/_5$, $u_{\text{id(prevtag:Verb)}} = 1$ and other coordinates are 0

$C_i \perp C_{j \notin \{i-1,i\}} | X = x, I = i, C_{\text{prev}} = r$: given x, that we want to tag the ith word, and that the previous word received tag $C_{\text{prev}} = r$, the distribution of the ith tag C_i is independent of all but the previous tag in the tag sequence.

Exercise (for after class)

Claim 1. Compared to the example feature function for the 0-order model, the example feature function for the 1-order model has K new coordinates.

Claim 2. Finding the best tag sequence for the 1-order model can be done as follows: start with $\hat{c}_0 = \mathrm{BOS}$, iteratively for each i from 1 to ℓ, solve:

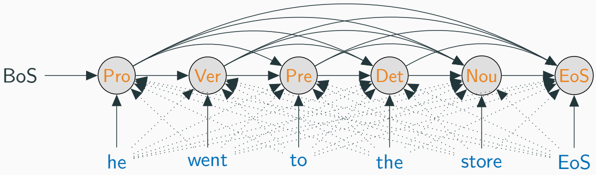
$$\hat{c}_i = \operatorname*{arg\,max}_{k \in [K]} \; P_{C|XI}(k | w_{1:\ell}, i, \hat{c}_{i-1})$$

Accept or reject the claims, justifying your decision in each case.

Conditional Chain Rule

We could use chain rule, conditioned on the word sequence

$$P_{Y|X}(c_{1:\ell}|w_{1:\ell}) \triangleq \prod_{i=1}^{\ell} P_{C|XH}(c_i|w_{1:\ell}, c_{<i}) \tag{10}$$



A powerful feature function for this is not trivial to design. That's because to describe the complex dependencies in the history using *indicators*, we increase the dimensionality of the feature space. We will get back to this when we know more about NNs.

Overview

Log-linear models can achieve *a lot*!

- We can use more context, word internal features, etc.
- They are more statistically efficient than tabular cpds: the size of the model does not depend on how many condition-outcome pairs are possible.
- They have been applied to POS tagging, NER, semantic role labelling, etc.

But they are tricky to design

- Good feature functions require enough intuitions about what's likely useful for a task.
- Interesting feature spaces are typically very large.

Summary

- HMMs combine generation and classification, they can be used as taggers or LMs.
- HMMs make strong factorisation assumptions and have are parameterised inefficiently (tabular cpds).
- Log-linear models are a good alternative parameterisation of Categorical cpds.
- Local log-linear models can power direct modelling of complex conditional distributions.
- These ideas power various sequence labelling tasks.

We can extend these in two ways: global modelling (not covered in this course, look for CRFs), and less Markov assumptions (with neural parameterisation).

What Next?

Self-study

- Exercises throughout slides (solutions at the end of slides)
- Watch the video on Viterbi/Forward
- Watch the videos on logistic CPDs: theory, example, and code.

Solutions

Feature Spaces

Here are templates for the coordinates of a feature function $\phi(w_{1:\ell}, i)$: word:w_i, before:w_{i-1}, after:w_{i+1}, all binary-valued. If we know a total of V words, what is the dimensionality D of this feature space? The feature space has to accommodate every possible instantiation of those templates, there are V ways to instantiate each of the 3 templates, hence $D = 3 \times V$.

What if we add the templates before2:w_{i-2} and after2:w_{i+2}? That would increment D with $2 \times V$ features, since each of these templates can be instantiated in V possible ways.

And what if we add the templates wordpref:prefix(w_i), wordsuff:suffix(w_i) and wordstem:stem(w_i)? Assume we know R prefixes, S suffixes, and a number $V' \propto V$ of stems. The are R ways to instantiate the wordpref template, S ways to instantiate the wordsuff template, and a number $V' \propto V$ ways to instantiate the template wordstem, hence we would increment D by $R + S + V'$.

0-order model

About the 0-order model we designed.

Claim 1: it cannot be used as a language model (i.e., it cannot assign probability to a given piece of text).

This is true because the model is not inferred from a joint distribution over the space of labelled text, instead, it is a probability distribution directly specified over the space of tag sequences.

Claim 2: finding the best tag sequence for a given text $w_{1:\ell}$ can be done by solving a sequence of ℓ independent classifications.

This is true because the definition of the Categorical parameter $\mathbf{f}(\mathbf{W}\phi(w_{1:\ell}, i) + \mathbf{b})$ is such that for any position i we would like to classify, it only has access to the text $w_{1:\ell}$ itself. In other words, for each and every step, there are no dependencies on other tags, and hence the tagging of each position is independent of the tagging of any other position.

About 1st-order model

Claim 1. Compared to the example feature function for the 0-order model, the example feature function for the 1-order model has K new coordinates.

This is true because the new feature function has a template prevtag:c_{i-1} for the previous tag, which can be instantiated in K possible ways.

Claim 2. Finding the best tag sequence for the 1-order model can be done as follows: start with $\hat{c}_0 = \text{BOS}$, iteratively for each i from 1 to ℓ, solve:

$$\hat{c}_i = \arg\max_{k \in [K]} \; P_{C|XI}(k | w_{1:\ell}, i, \hat{c}_{i-1})$$

This is false because now each tagging decision depends on the tagging decision done previously. It is possible that a decision that's locally optimal for step $i = 2$ is not globally optimal a few steps later. *Extra information for you:* To solve this correctly, we need dynamic programming. As the *unobserved* variables of this problem (the tags in the tag sequence) depend on one another in the same way as they would in the standard HMM (i.e., in a 1st-order linear chain), we can actually use a version of the Viterbi algorithm for this.

References