Introduction to Computational Linguistics and Natural Language Processing

Mark Johnson

Department of Computing
Macquarie University

2014 Machine Learning Summer School

Updated slides available from
http://web.science.mq.edu.au/~mjohnson/

Outline

Overview of computational linguistics and natural language processing

Key ideas in NLP and CL

Grammars and parsing

Non-parametric Bayesian extensions to grammars

Conclusion and future directions

Natural language processing and computational linguistics

- *Natural language processing* (NLP) develops methods for solving practical problems involving language
  - automatic speech recognition
  - machine translation
  - information extraction from documents
- *Computational linguistics* (CL) studies the computational processes underlying (human) language
  - how do we understand language?
  - how do we produce language?
  - how do we learn language?
- Similiar methods and models are used in NLP and CL
  - my recommendation: *be clear what your goal is!*

A brief history of CL and NLP

- Computational linguistics goes back to the dawn of computer science
  - syntactic parsing and machine translation started in the 1950s
- Until the 1990s, computational linguistics was closely connected to linguistics
  - linguists write grammars, computational linguists implement them
- The "statistical revolution" in the 1990s:
  - techniques developed in neighbouring fields work better
    - hidden Markov models produce better speech recognisers
    - bag-of-words methods like tf-idf produce better information retrieval systems

  $\Rightarrow$ NLP and CL adopted probabilistic models
- NLP and CL today:
  - oriented towards machine learning rather than linguistics
  - NLP applications-oriented, driven by large internet companies

MACQUARIE UNIVERSITY

Outline

MACQUARIE UNIVERSITY

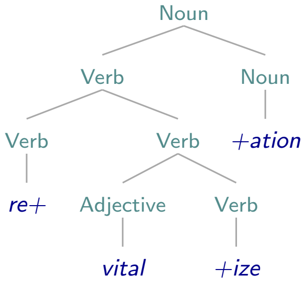
Phonetics and phonology

- *Phonetics* studies the *sounds* of a language
  - E.g., [*t*] and [*d*] differ in *voice onset time*
  - E.g., English *aspirates stop consonants* in certain positions (e.g., [$t^h op$] vs. [*stop*])
- *Phonology* studies the *distributional properties of these sounds*
  - E.g., the English noun plural is [*s*] following unvoiced segments and [*z*] following voiced segments
  - E.g., English speakers pronounce /*t*/ differently (e.g., in *water*)

MACQUARIE UNIVERSITY

Morphology

- *Morphology* studies the structure of words
  - E.g., *re+structur+ing*, *un+remark+able*
- *Derivational morphology* exhibits *hierarchical structure*
- Example: *re+vital+ize+ation*

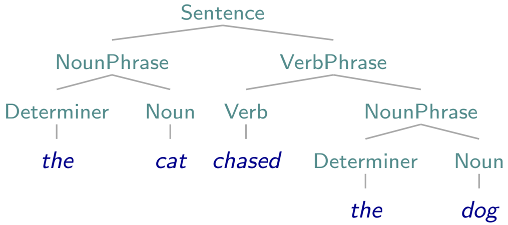


- The *suffix* usually determines the *syntactic category* of the derived word

MACQUARIE UNIVERSITY

Syntax

- *Syntax* studies the ways words combine to form phrases and sentences



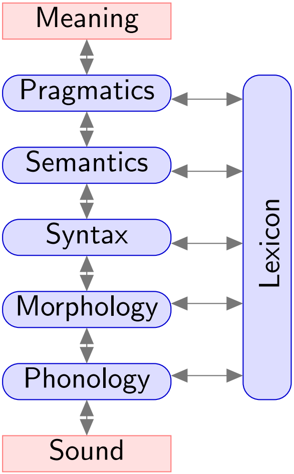
- Syntactic parsing helps identify *who did what to whom*, a key step in understanding a sentence

Semantics and pragmatics

- *Semantics* studies the *meaning of words, phrases and sentences*
  - E.g., *I ate the oysters in/for an hour.*
  - E.g., *Who do you want to talk to $\emptyset$/him?*
- *Pragmatics* studies how we use language to *do things in the world*
  - E.g., *Can you pass the salt?*
  - E.g., in a letter of recommendation:
    Sam is punctual and extremely well-groomed.

The lexicon

- A language has a *lexicon*, which lists for each morpheme
  - how it is pronounced (phonology),
  - its distributional properties (morphology and syntax),
  - what it means (semantics), and
  - its discourse properties (pragmatics)
- The lexicon interacts with all levels of linguistic representation



MACQUARIE UNIVERSITY

Linguistic levels on one slide

- *Phonology* studies the *distributional patterns of sounds*
  - E.g., *cats* vs *dogs*
- *Morphology* studies the *structure of words*
  - E.g., *re+vital+ise*
- *Syntax* studies how *words combine to form phrases and sentences*
  - E.g., *Flying planes can be dangerous*
- *Semantics* studies how *meaning is associated with language*
  - E.g., *I sprayed the paint onto the wall/I sprayed the wall with paint*
- *Pragmatics* studies how *language is used to do things*
  - E.g., *Can you pass the salt?*
- The *lexicon* stores phonological, morphological, syntactic, semantic and pragmatic information about morphemes and words

MACQUARIE UNIVERSITY

Outline

MACQUARIE UNIVERSITY

What's driving NLP and CL research?

- Tools for managing the "information explosion"
  - extracting information from and managing large text document collections
  - NLP is often free "icing on the cake" to sell more ads; e.g., speech recognition, machine translation, document clustering (news), etc.
- Mobile and portable computing
  - keyword search / document retrieval don't work well on very small devices
  - we want to be able to talk to our computers (speech recognition) and have them say something intelligent back (NL generation)
- The intelligence agencies
- The old Artificial Intelligence (AI) dream
  - language is the richest window into the mind

Automatic speech recognition

- Input: an acoustic waveform a
- Output: a text transcript $\widehat{t}(a)$ of a
- Challenges for Automatic Speech Recognition (ASR):
  - speaker and pronunciation variability
    the same text can be pronounced in many different ways
  - homophones and near homophones:
    e.g. *recognize speech* vs. *wreck a nice beach*

MACQUARIE UNIVERSITY

Machine translation

- Input: a sentence (usually text) f in the *source language*
- Output: a sentence e in the *target language*
- Challenges for Machine Translation:
  - the best translation of a word or phrase depends on the context
  - the order of words and phrases varies from language to language
  - there's often no single "correct translation"

The inspiration for statistical machine translation

Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography — methods which I believe succeed even when one does not know what language has been coded — one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography.

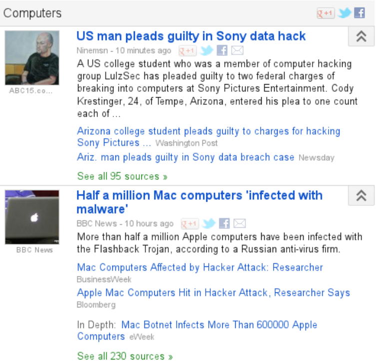
When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Warren Weaver – 1947

MACQUARIE UNIVERSITY

Topic modelling

- Topic models *cluster documents on same topic*
  - *unsupervised* (i.e., topics aren't given in training data)
- Important for document analysis and information extraction
  - Example: clustering news stories for information retrieval
  - Example: tracking evolution of a research topic over time



Example input to a topic model (NIPS corpus)

Annotating an unlabeled dataset is one of the bottlenecks in using supervised learning to build good predictive models. Getting a dataset labeled by experts can be expensive and time consuming. With the advent of crowdsourcing services ...

The task of recovering intrinsic images is to separate a given input image into its material-dependent properties, known as reflectance or albedo, and its light-dependent properties, such as shading, shadows, specular highlights, ...

In each trial of a standard visual short-term memory experiment, subjects are first presented with a display containing multiple items with simple features (e.g. colored squares) for a brief duration and then, after a delay interval, their memory for ...

Many studies have uncovered evidence that visual cortex contains specialized regions involved in processing faces but not other object classes. Recent electrophysiology studies of cells in several of these specialized regions revealed that at least some ...

Example (cont): ignore function words

Annotating an unlabeled dataset is one of the bottlenecks in using supervised learning to build good predictive models. Getting a dataset labeled by experts can be expensive and time consuming. With the advent of crowdsourcing services . . .

The task of recovering intrinsic images is to separate a given input image into its material-dependent properties, known as reflectance or albedo, and its light-dependent properties, such as shading, shadows, specular highlights, . . .

In each trial of a standard visual short-term memory experiment, subjects are first presented with a display containing multiple items with simple features (e.g. colored squares) for a brief duration and then, after a delay interval, their memory for . . .

Many studies have uncovered evidence that visual cortex contains specialized regions involved in processing faces but not other object classes. Recent electrophysiology studies of cells in several of these specialized regions revealed that at least some . . .

Example (cont): admixture topic model

Annotating an unlabeled dataset is one of the bottlenecks in using supervised learning to build good predictive models. Getting a dataset labeled by experts can be expensive and time consuming. With the advent of crowdsourcing services . . .

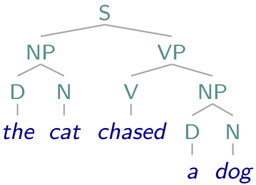
The task of recovering intrinsic images is to separate a given input image into its material-dependent properties, known as reflectance or albedo, and its light-dependent properties, such as shading, shadows, specular highlights, . . .

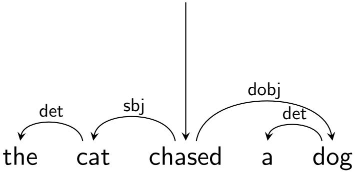
In each trial of a standard visual short-term memory experiment, subjects are first presented with a display containing multiple items with simple features (e.g. colored squares) for a brief duration and then, after a delay interval, their memory for . . .

Many studies have uncovered evidence that visual cortex contains specialized regions involved in processing faces but not other object classes. Recent electrophysiology studies of cells in several of these specialized regions revealed that at least some . . .

MACQUARIE UNIVERSITY

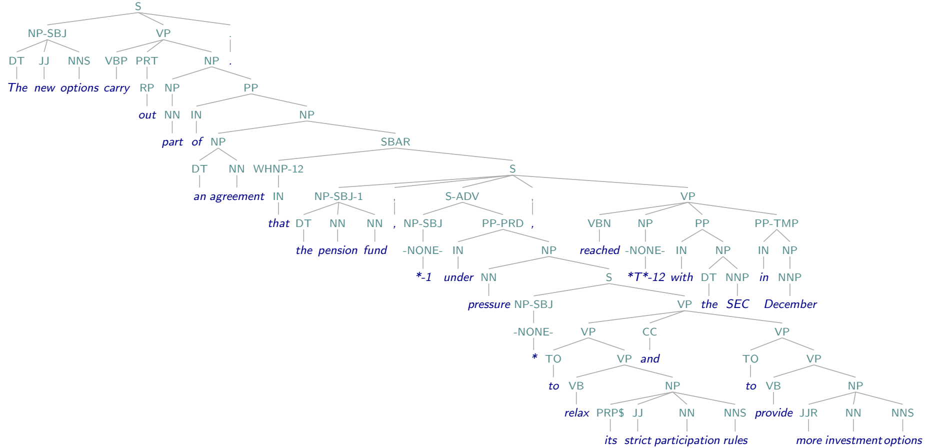
Phrase structure and dependency parses





- A *phrase structure parse* represents phrases as nodes in a tree
- A *dependency parse* represents dependencies between words
- Phrase structure and dependency parses are approximately inter-translatable:
  - Dependency parses can be translated to phrase structure parses
  - If every phrase in a phrase structure parse has a *head word*, then phrase structure parses can be translated to dependency parses

Syntactic structures of real sentences



- State-of-the-art parsers have accuracies of over 90%
- Dependency parsers can parse thousands of sentences a second

MACQUARIE UNIVERSITY

Advantages of probabilistic parsing

- In the GofAI approach to syntactic parsing:
  - a hand-written grammar defines the grammatical (i.e., well-formed) parses
  - given a sentence, the parser returns the set of grammatical parses for that sentence

  ⇒ unable to distinguish more likely from less likely parses

  ⇒ hard to ensure *robustness* (i.e., that every sentence gets a parse)
- In a probabilistic parser:
  - the grammar *generates all possible parse trees* for all possible strings (roughly)
  - use probabilities to identify plausible syntactic parses
- Probabilistic syntactic models usually encode:
  - the probabilities of syntactic constructions
  - the probabilities of lexical dependencies

    e.g., how likely is *pizza* as direct object of *eat*?

MACQUARIE UNIVERSITY

Named entity recognition and linking

- *Named entity recognition* finds all "mentions" referring to an entity in a document

  Example: $\underbrace{\textit{Tony Abbott}}_{\text{PERSON}}$ *bought* $\underbrace{\textit{300}}_{\text{NUMBER}}$ *shares in* $\underbrace{\textit{Acme Corp}}_{\text{CORPORATION}}$ *in* $\underbrace{\textit{2006}}_{\text{DATE}}$

- *Noun phrase coreference* tracks mentions to entities within or across documents

  Example: *Julia Gillard met the president of Indonesia yesterday. Ms. Gillard told him that she ...*

- *Entity linking* maps entities to database entries

  Example: $\underbrace{\textit{Tony Abbott}}_{\text{/M/XW2135}}$ *bought* $\underbrace{\textit{300}}_{\text{NUMBER}}$ *shares in* $\underbrace{\textit{Acme Corp}}_{\text{/M/YZW9W}}$ *in* $\underbrace{\textit{2006}}_{\text{DATE}}$

MACQUARIE UNIVERSITY

Relation extraction

- *Relation extraction* mines texts to find *relationships between named entities*, i.e., "who did what to whom (when)?"

  The new Governor General, Peter Cosgrove, visited Buckingham Palace yesterday.

Has-role

Person	**Role**
Peter Cosgrove	Governor General of Australia

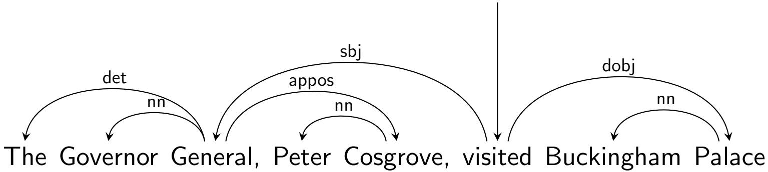
Offical-visit

Visitor	**Organisation**
Peter Cosgrove	Queen of England

- The syntactic parse provides useful features for relation extraction
- Text mining bio-medical literature is a major application

MACQUARIE UNIVERSITY

Syntactic parsing for relation extraction

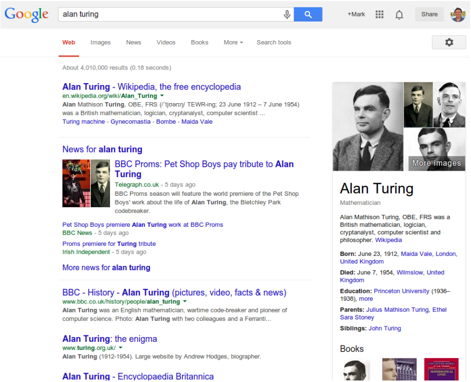


- The *syntactic path* in a *dependency parse* is a useful feature in relation extraction

$$X \xrightarrow{\text{appos}} Y \Rightarrow \textit{has-role}(Y, X)$$

$$X \xleftarrow{\text{sbj}} \textit{visited} \xrightarrow{\text{dobj}} Y \Rightarrow \textit{official-visit}(X, Y)$$

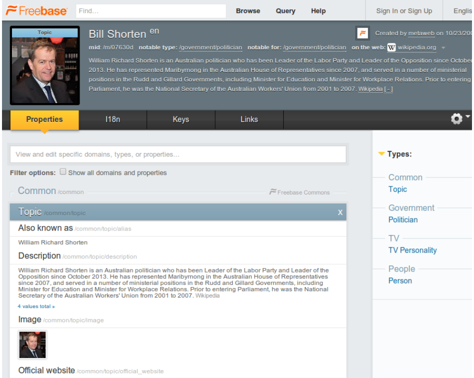
Google's Knowledge Graph



- Goal: move beyond keyword search document retrieval to *directly answer user queries*
  - ⇒ easier for mobile device users
- Google's Knowledge Graph:
  - built on top of FreeBase
  - entries are synthesised from Wikipedia, news stories, etc.
  - manually curated (?)

MACQUARIE UNIVERSITY

FreeBase: an open (?) knowledge base



- An entity-relationship database on top of a graph triple store
- Data mined from Wikipedia, ChefMoz, NNDB, FMD, MusicBrainz, etc.
- 44 million topics (entities), 2 billion facts, 25GB compressed dump
- Created by Metaweb, which was acquired by Google

MACQUARIE UNIVERSITY

Distant supervision for relation extraction

- Ideal labelled data for relation extraction: large text corpus annotated with entities and relations
  - expensive to produce, especially for a lot of relations!
- *Distant supervision assumption:* if two or more entities that appear in the same sentence also appear in the same database relation, then probably the sentence expresses the relation
  - assumes entity tuples are sparse
- With the distant supervision assumption, we obtain relation extraction training data by:
  - taking a large text corpus (e.g., 10 years of news articles)
  - running a named entity linker on the corpus
  - looking up the entity tuples that appear in the same sentence in the large knowledge base (e.g., FreeBase)

Opinion mining and sentiment analysis

- Used to analyse e.g., social media (Web 2.0)
- Typical goals: given a corpus of messages:
  - classify each message along a *subjective–objective scale*
  - identify the message *polarity* (e.g., on dislike–like scale)
- Training opinion mining and sentiment analysis models:
  - in some domains, *supervised learning* with simple *keyword-based features* works well
  - but in other domains it's necessary to model *syntactic structure* as well
    - E.g., *I doubt she had a very good experience ...*
- Opinion mining can be combined with:
  - *topic modelling* to cluster messages with similar opinions
  - multi-document *summarisation* to summarise results

MACQUARIE UNIVERSITY

Why do statistical models work so well?

- Statistical models can be *trained from large datasets*
  - large document collections are available or can be constructed
  - machine learning methods can automatically adjust a model so it *performs well on the data it will be used on*
- Probabilistic models can *integrate disparate and potentially conflicting evidence*
  - standard linguistic methods make hard categorical classifications
  - the weighted features used in probabilistic models can *weigh conflicting information from diverse sources*
- Statistical models can *rank alternative possible analyses*
  - in NLP, the *number of possible analyses is often astronomical*
  - a statistical model provides a *principled way of selecting the most probable analysis*

Outline

Overview of computational linguistics and natural language processing

Key ideas in NLP and CL

Grammars and parsing

Non-parametric Bayesian extensions to grammars

Conclusion and future directions

MACQUARIE UNIVERSITY

Why is NLP difficult?

- Abstractly, most NLP applications can be viewed as prediction problems
  $\Rightarrow$ should be able to solve them with Machine Learning
- The label set is often the set of all possible sentences
  - infinite (or at least astronomically large)
  - constrained in ways we don't fully understand
- Training data for supervised learning is often not available
  $\Rightarrow$ techniques for training from available data
- Algorithmic challenges
  - vocabulary can be large (e.g., 50K words)
  - data sets are often large (GB or TB)

Outline

Motivation for the noisy channel model

- Speech recognition and machine translation models as *prediction problems*:
  - speech recognition: given an acoustic string, predict the text
  - translation: given a foreign source language sentence, predict its target language translation
- The "natural" training data for these tasks is relatively rare/expensive:
  - speech recognition: *acoustic signals labelled with text transcripts*
  - translation: *(source language sentence,target language translation) pairs*
- The noisy channel model lets us leverage *monolingual text data in output language*
  - large amounts of such text are cheaply available

MACQUARIE UNIVERSITY

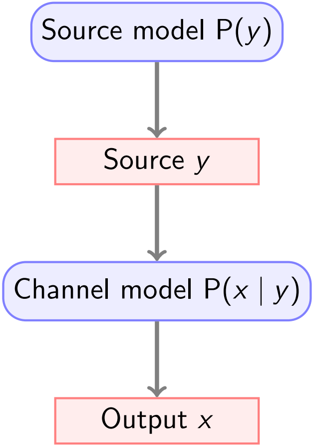
The noisy channel model

- The *noisy channel model* is a common structure for *generative models*
  - the *source* y is a *hidden variable* generated by $\mathrm{P}(y)$
  - the *output* x is a *visible variable* generated from y by $\mathrm{P}(y \mid x)$
- Given output x, find *most likely source* $\widehat{y}(x)$

$$\widehat{y}(x) \quad = \quad \operatorname{argmax} \mathrm{P}(y \mid x)$$

- Bayes rule: $\mathrm{P}(y \mid x) \;=\; \dfrac{\mathrm{P}(x \mid y)\,\mathrm{P}(y)}{\mathrm{P}(x)}$
- Since output x is fixed:

$$\widehat{y}(x) \quad = \quad \operatorname*{argmax}_{y} \mathrm{P}(x \mid y)\,\mathrm{P}(y)$$



MACQUARIE UNIVERSITY

The noisy channel model in speech recognition

- Input: acoustic signal a
- Output: most likely text $\widehat{t}(a)$, where:

$$\begin{aligned}\widehat{t}(a) &= \operatorname*{argmax}_t \mathrm{P}(t \mid a) \\ &= \operatorname*{argmax}_t \mathrm{P}(a \mid t)\,\mathrm{P}(t), \text{ where:}\end{aligned}$$

  - $\mathrm{P}(a \mid t)$ is an *acoustic model*, and
  - $\mathrm{P}(t)$ is an *language model*
- The acoustic model uses pronouncing dictionaries to decompose the sentence t into sequences of phonemes, and map each phoneme to a portion of the acoustic signal a
- The language model is responsible for distinguishing *more likely sentences* from *less likely sentences* in the output text, e.g., distinguishing *recognise speech* vs. *wreck a nice beach*

The noisy channel model in machine translation

- Input: target language sentence f
- Output: most likely source language sentence $\widehat{e}(f)$, where:

$$\begin{aligned}\widehat{e}(f) &= \underset{e}{\operatorname{argmax}}\, \mathrm{P}(e \mid f) \\ &= \underset{e}{\operatorname{argmax}}\, \mathrm{P}(f \mid e)\, \mathrm{P}(e), \text{ where:}\end{aligned}$$

  - $\mathrm{P}(f \mid e)$ is a *translation model*, and
  - $\mathrm{P}(e)$ is an *language model*
- The translation model calculates $\mathrm{P}(f \mid e)$ as a product of two submodels:
  - a *word or a phrase translation model*
  - a *distortion model*, which accounts for the word and phrase reorderings between source and target language
- The language model is responsible for distinguishing *more fluent sentences* from *less fluent sentences* in the target language, e.g., distinguishing *Sasha will the car lead* vs. *Sasha will drive the car*

MACQUARIE UNIVERSITY

Outline

Overview of computational linguistics and natural language processing

Key ideas in NLP and CL

- The noisy channel model
- Language models
- Sequence labelling models
- Expectation Maximisation (EM)

Grammars and parsing

Non-parametric Bayesian extensions to grammars

Conclusion and future directions

The role of language models

- A language model estimates the probability $\mathsf{P}(\boldsymbol{w})$ that a string of words $\boldsymbol{w}$ is a sentence
  - useful in tasks such as speech recognition and machine translation that involve predicting entire sentences
- Language models provide a way of *leveraging large amounts of text* (e.g., from the web)
- Primary challenge in language modelling: *infinite number of possible sentences*

$\Rightarrow$ *Factorise $P(\boldsymbol{w})$ into a product of submodels*
  - we'll look at *n-gram sequence models* here
  - but *syntax-based language models* are also used, especially in machine translation

MACQUARIE UNIVERSITY

n-gram language models

- Goal: estimate $\mathrm{P}(\boldsymbol{w})$, where $\boldsymbol{w} = (w_1, \ldots, w_m)$ is a sequence of words
- n-gram models decompose $\mathrm{P}(\boldsymbol{w})$ into product of conditional distributions

$$\mathrm{P}(\boldsymbol{w}) \quad = \quad \mathrm{P}(w_1)\mathrm{P}(w_2 \mid w_1)\mathrm{P}(w_3 \mid w_1, w_2) \ldots \mathrm{P}(w_m \mid w_1, \ldots, w_{m-1})$$

E.g., $\mathrm{P}(\textit{wreck a nice beach}) \; = \; \mathrm{P}(\textit{wreck})\,\mathrm{P}(\textit{a} \mid \textit{wreck})\,\mathrm{P}(\textit{nice} \mid \textit{wreck a})$
$\mathrm{P}(\textit{beach} \mid \textit{wreck a nice})$

- n-gram assumption: *no dependencies span more than n words*, i.e.,

$$\mathrm{P}(w_i \mid w_1, \ldots, w_{i-1}) \quad \approx \quad \mathrm{P}(w_i \mid w_{i-n}, \ldots, w_{i-1})$$

E.g., A *bigram model* is an n-gram model where $n = 2$:

$$\mathrm{P}(\textit{wreck a nice beach}) \quad \approx \quad \mathrm{P}(\textit{wreck})\,\mathrm{P}(\textit{a} \mid \textit{wreck})\,\mathrm{P}(\textit{nice} \mid \textit{a})\,\mathrm{P}(\textit{beach} \mid \textit{nice})$$

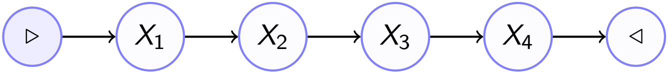
MACQUARIE UNIVERSITY

n-gram language models as Markov models and Bayes nets

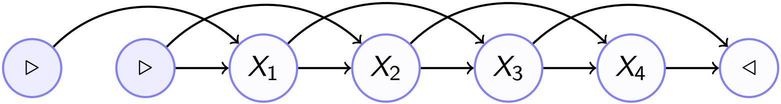
- An n-gram language model is a *Markov model* that *factorises the distribution over sentences into a product of conditional distributions*:

$$\mathrm{P}(\boldsymbol{w}) \;=\; \prod_{i=1}^{m} \mathrm{P}(w_i \mid w_{i-n}, \ldots, w_{i-1})$$

  - *pad $\boldsymbol{w}$ with end markers*, i.e., $\boldsymbol{w} = (\triangleright, x_1, x_2, \ldots, x_m, \triangleleft)$
- Bigram language model as Bayes net:



- Trigram language model as Bayes net:



The conditional word models in *n*-gram models

- An *n*-gram model factorises $\mathrm{P}(\boldsymbol{w})$ into a product of conditional models, each of the form:

$$\mathrm{P}(x_n \mid x_1, \ldots, x_{n-1})$$

- The performance of an *n*-gram model depends greatly on exactly how these conditional models are defined
    - huge amount of work on this
- *Deep learning* methods for estimating these conditional distributions currently produce state-of-the-art language models

MACQUARIE UNIVERSITY

Outline

What is sequence labelling?

- A *sequence labelling* problem is one where:
  - the input consists of a sequence $\boldsymbol{X} = (X_1, \ldots, X_n)$, and
  - the output consists of a sequence $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ of labels, where:
  - Y_i is the label for element X_i
- Example: Part-of-speech tagging

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{pmatrix} = \begin{pmatrix} \text{Verb}, & \text{Determiner}, & \text{Noun} \\ \text{spread}, & \text{the}, & \text{butter} \end{pmatrix}$$

- Example: Spelling correction

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{pmatrix} = \begin{pmatrix} \text{write}, & \text{a}, & \text{book} \\ \text{rite}, & \text{a}, & \text{buk} \end{pmatrix}$$

Named entity extraction with IOB labels

- *Named entity recognition and classification* (NER) involves finding the named entities in a text and identifying what type of entity they are (e.g., person, location, corporation, dates, etc.)
- NER can be formulated as a sequence labelling problem
- *Inside-Outside-Begin* (IOB) labelling scheme indicates the *beginning and span* of each named entity

B-ORG	I-ORG	O	O	O	B-LOC	I-LOC	I-LOC	O
Macquarie	University	is	located	in	New	South	Wales	.

- The IOB labelling scheme lets us identify *adjacent named entities*

B-LOC	I-LOC	I-LOC	B-LOC	I-LOC	O	B-LOC	O	...
New	South	Wales	Northern	Territory	and	Queensland	are	...

- This technology can extract information from:
  - news stories
  - financial reports
  - classified ads

Other applications of sequence labelling

- Speech transcription as a sequence labelling task
  - The input $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a sequence of *acoustic frames* X_i, where X_i is a set of features extracted from a 50msec window of the speech signal
  - The output $\boldsymbol{Y}$ is a sequence of words (the transcript of the speech signal)
- Financial applications of sequence labelling
  - identifying trends in price movements
- Biological applications of sequence labelling
  - gene-finding in DNA or RNA sequences

A first (bad) approach to sequence labelling

- Idea: train a supervised classifier to *predict entire label sequence at once*

B-ORG	I-ORG	O	O	O	B-LOC	I-LOC	I-LOC	O
Macquarie	University	is	located	in	New	South	Wales	.

- Problem: *the number of possible label sequences grows exponentially with the length of the sequence*
  - with *binary labels*, there are 2^n different label sequences of a sequence of length n ($2^{32} = 4$ billion)

$\Rightarrow$ most labels won't be observed even in very large training data sets

- This approach fails because it has massive *sparse data problems*

MACQUARIE UNIVERSITY

A better approach to sequence labelling

- Idea: train a supervised classifier to *predict the label of one word at a time*

B-LOC	I-LOC	O	O	O	O	O	B-LOC	O
Western	Australia	is	the	largest	state	in	Australia	.

- Avoids sparse data problems in label space
- As well as current word, classifiers can use *previous and following words as features*
- But this approach can produce *inconsistent label sequences*

O	B-LOC	I-LOC	I-ORG	O	O	O	O
The	New	York	Times	is	a	newspaper	.

⇒ Track *dependencies between adjacent labels*

- "chicken-and-egg" problem that Hidden Markov Models and Conditional Random Fields solve!

MACQUARIE UNIVERSITY

Introduction to Hidden Markov models

- Hidden Markov models (HMMs) are a simple sequence labelling model
- HMMs are *noisy channel models* generating

$$\mathrm{P}(\boldsymbol{X}, \boldsymbol{Y}) \;=\; \mathrm{P}(\boldsymbol{X} \mid \boldsymbol{Y})\mathrm{P}(\boldsymbol{Y})$$

  - the *source model* $\mathrm{P}(\boldsymbol{Y})$ is a Markov model (e.g., a bigram language model)

$$\mathrm{P}(\boldsymbol{Y}) \;=\; \prod_{i=1}^{n+1} \mathrm{P}(Y_i \mid Y_{i-1})$$

  - the *channel model* $\mathrm{P}(\boldsymbol{X} \mid \boldsymbol{Y})$ generates each X_i independently, i.e.,

$$\mathrm{P}(\boldsymbol{X} \mid \boldsymbol{Y}) \;=\; \prod_{i=1}^{n} \mathrm{P}(X_i \mid Y_i)$$

- At testing time we only know $\boldsymbol{X}$, so $\boldsymbol{Y}$ is unobserved or *hidden*

MACQUARIE UNIVERSITY

Terminology in Hidden Markov Models

- Hidden Markov models (HMMs) generate pairs of sequences $(\boldsymbol{x}, \boldsymbol{y})$
- The sequence $\boldsymbol{x}$ is called:
  - the *input sequence*, or
  - the *observations*, or
  - the *visible data*

  because $\boldsymbol{x}$ is given when an HMM is used for sequence labelling
- The sequence $\boldsymbol{y}$ is called:
  - the *label sequence*, or
  - the *tag sequence*, or
  - the *hidden data*

  because $\boldsymbol{y}$ is unknown when an HMM is used for sequence labelling
- A $y \in \mathcal{Y}$ is sometimes called a *hidden state* because an HMM can be viewed as a *stochastic automaton*
  - each different $y \in \mathcal{Y}$ is a state in the automaton
  - the x are *emissions* from the automaton

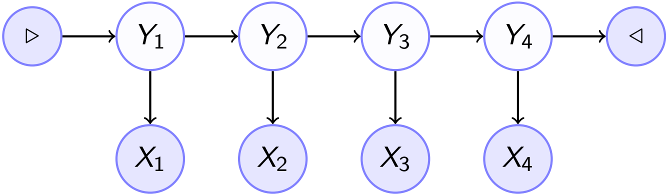
MACQUARIE UNIVERSITY

Hidden Markov models

- A *Hidden Markov Model* (HMM) defines a joint distribution $\mathrm{P}(\boldsymbol{X}, \boldsymbol{Y})$ over:
  - *item sequences* $\boldsymbol{X} = (X_1, \ldots, X_n)$ and
  - *label sequences* $\boldsymbol{Y} = (Y_0 = \triangleright, Y_1, \ldots, Y_n, Y_{n+1} = \triangleleft)$:

$$\mathrm{P}(\boldsymbol{X}, \boldsymbol{Y}) \quad = \quad \left(\prod_{i=1}^{n} \mathrm{P}(Y_i \mid Y_{i-1}) \, \mathrm{P}(X_i \mid Y_i) \right) \mathrm{P}(Y_{n+1} \mid Y_n)$$

- HMMs can be expressed as Bayes nets, and standard message-passing inference algorithms work well with HMMs

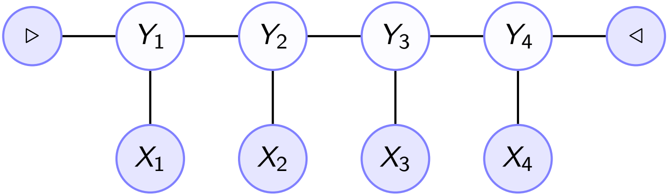


Conditional random fields

- *Conditional Random Fields* (CRFs) are the Markov Random Field generalisation of HMMs.

$$\mathrm{P}(\boldsymbol{X}, \boldsymbol{Y}) \quad = \quad \frac{1}{Z} \left(\prod_{i=1}^{n} \theta_{Y_{i-1}, Y_i} \; \psi_{Y_i, X_i} \right) \theta_{Y_n, Y_{n+1}}$$

- CRFs are usually used to define *conditional distributions* $\mathrm{P}(\boldsymbol{Y} \mid \boldsymbol{X})$ over *label sequences* $\boldsymbol{Y}$ given *observed sequences* $\boldsymbol{X}$
- CRFs can be expressed using the undirected MRF graphical models

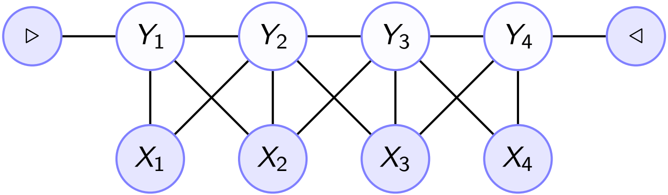


Advantages of CRFs over HMMs

- Recall that in MRFs, *conditioning on a node deletes the node and all edges connected to it*
  - after conditioning on X all that remains is a linear chain

$\Rightarrow$ Complexity of computing $\mathrm{P}(\boldsymbol{Y} \mid \boldsymbol{X}{=}\boldsymbol{x})$ *does not depend on complexity of connections between* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$

$\Rightarrow$ We can use *arbitrary features* to connect $\boldsymbol{X}$ and $\boldsymbol{Y}$
  - must optimise *conditional likelihood* for training to be tractable

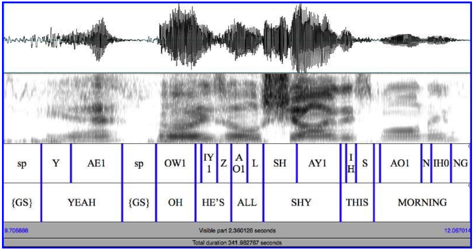


MACQUARIE UNIVERSITY

Outline

Ideal training data for acoustic models

- The acoustic model $P(a \mid t)$ in a speech recogniser predicts the acoustic waveform a given a text transcript t
- Ideal training data for an acoustic model would:
  - segment the acoustic waveform into phones
  - map the phones to words in the text

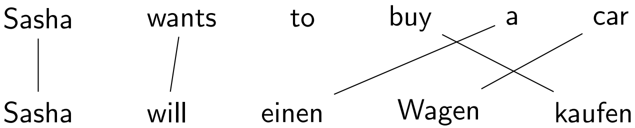


- Manually segmenting and labelling speech is *very expensive*!
- Expectation Maximisation lets us induce this information from cheap *sentence level transcripts*

MACQUARIE UNIVERSITY

Ideal training data for the translation model

- The translation model $\mathrm{P}(f \mid e)$ in a MT system:
  - predicts the translation of each word or phrase, and
  - predicts the reordering or words and phrases
- Ideal training data would *align words and phrases in the source and target language sentences*



- Manually aligning words and phrases is *very expensive*!
- Expectation Maximisation lets us induce this information from cheap *sentence aligned translations*

What Expectation Maximisation does

- Expectation Maximisation (and related techniques such as Gibbs sampling and Variational Bayes) are *"recipies" for generalising maximum likelihood supervised learning methods to unsupervised learning problems*
  - they are techniques for *hidden variable imputation*
- Intuitive idea behind the EM algorithm:
  - if we had a good acoustic model/translation model,
    we could use it to compute the phoneme labelling/word alignment
- Intuitive description of the EM algorithm:

  guess an initial model somehow:
  repeat until converged:
    use current model to label the data
    learn a new model from the labelled data

- Amazingly, this *provably converges* under very general conditions, and
- it converges to a *local maximum* of the likelihood

MACQUARIE UNIVERSITY

Forced alignment for training speech recognisers

- Speech recogniser training typically uses *forced alignment* to produce a *phone labelling* of the training data
- Inputs to forced alignment:
  - a speech corpus with *sentence-level transcripts*
  - a *pronouncing dictionary*, mapping words to their possible phone sequences
  - an *acoustic model* mapping phones to waveforms trained on a small amount of data
- *Forced alignment procedure* (a version of EM)

```
repeat until converged:
   for each sentence s in the training data:
      use pronouncing dictionary to find all possible phone sequences for s
      use current acoustic model to compute probability
            of each possible alignment of each phone sequence
      keep most likely phone alignments for s
   retrain acoustic model based on most likely phone alignments
```

MACQUARIE UNIVERSITY

Mathematical description of EM

- Input: data $\tilde{x}$ and a model $\mathrm{P}(x, z, \theta)$ where *finding the "visible data" MLE $\widehat{\theta}$ would be easy* if we knew $\tilde{x}$ *and $\tilde{z}$*:

$$\widehat{\theta} \;=\; \operatorname*{argmax}_{\theta} \log \mathrm{P}_{\theta}(\tilde{x}, \tilde{z})$$

- The *"hidden data" MLE $\widehat{\widehat{\theta}}$* (which EM approximates) is:

$$\widehat{\widehat{\theta}} \;=\; \operatorname*{argmax}_{\theta} \log \mathrm{P}_{\theta}(\tilde{x}) \;=\; \operatorname*{argmax}_{\theta} \log \sum_{z} \mathrm{P}_{\theta}(\tilde{x}, z)$$

- The EM algorithm:
  initialise $\theta^{(0)}$ somehow (e.g., randomly)
  for $t = 1, 2, \ldots$ until convergence:
  E-step: set $Q^{(t)}(z) = \mathrm{P}_{\theta^{(t-1)}}(\tilde{x}, z)$
  M-step: set $\theta^{(t)} = \operatorname{argmax}_{\theta} \sum_{z} Q^{(t)}(z) \log \mathrm{P}_{\theta}(\tilde{x}, z)$
- $\theta^{(t)}$ converges to a *local maximum* of the hidden data likelihood
  - the $Q(z)$ distributions *impute values for the hidden variable z*
  - in practice we summarise $Q(z)$ with *expected values of the sufficient statistics for θ*

MACQUARIE UNIVERSITY

EM versus directly optimising log likelihood

- It's possible to directly optimise the "hidden data" log likelihood with a gradient-based approach (e.g., SGD, L-BFGS):

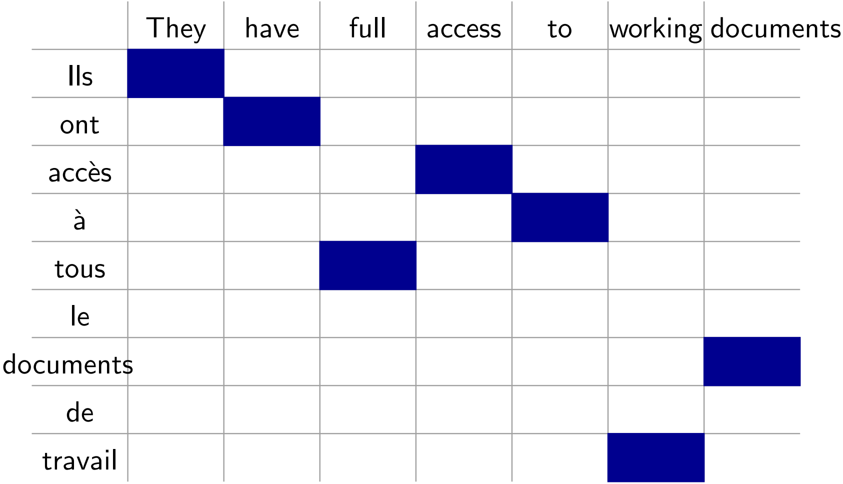
$$\widehat{\widehat{\theta}} \;=\; \underset{\theta}{\operatorname{argmax}} \log \mathrm{P}_{\theta}(\tilde{x}) \;=\; \underset{\theta}{\operatorname{argmax}} \log \sum_{z} \mathrm{P}_{\theta}(\tilde{x}, z)$$

- The log likelihood is typically not convex $\Rightarrow$ local maxima
- If the model is in the exponential family (most NLP models are), the derivatives of the log likelihood are the same expectations as required for EM

  $\Rightarrow$ both EM and direct optimisation are equally hard to program
- EM has no adjustable parameters, while SGD and L-BFGS have adjustable parameters (e.g., step size)
- I don't know of any systematic study, but in my experience:
  - EM starts faster $\Rightarrow$ if you're only going to do a few iterations, use EM
  - after many iterations, L-BFGS converges faster (quadratically)

MACQUARIE UNIVERSITY

A word alignment matrix for sentence translation pair



- Can we use this to learn the probability $\mathrm{P}(f \mid e) = \theta_{f,e}$ of English word e translating to French word f?

Learning P(f | e) from a word-aligned corpus

- A *word alignment* $\boldsymbol{a}$ pairs each French word f_k with its English translation word e_{a_k}
  - an English word may be aligned with several French words
  - $\Diamond$ generates French words with no English translation
- Let $\mathrm{P}(f \mid e) = \theta_{f,e}$. The MLE $\widehat{\boldsymbol{\theta}}$ is:

$$
\begin{aligned}
\widehat{\theta}_{f,e} &= \frac{n_{f,e}(\boldsymbol{a})}{n_{\cdot,e}(\boldsymbol{a})}, \text{ where:} \\
n_{f,e}(\boldsymbol{a}) &= \text{number of times } f \text{ aligns to } e \\
n_{\cdot,e}(\boldsymbol{a}) &= \sum_f n_{f,e}(\boldsymbol{a}) \\
&= \text{number of times } e \text{ aligns to anything}
\end{aligned}
$$

Position	$\boldsymbol{e}$	$\boldsymbol{a}$	$\boldsymbol{f}$
0	$\Diamond$		
1	*They*		*Ils*
2	*have*		*ont*
3	*full*		*accès*
4	*access*		*à*
5	*to*		*tous*
6	*working*		*le*
7	*documents*		*documents*
			de
			travail

$\boldsymbol{a} = (1, 2, 4, 5, 3, 0, 7, 0, 6)$

Sentence-aligned parallel corpus (Canadian Hansards)

- *English:* $\boldsymbol{e}$

  provincial officials are consulted through conference calls and negotiated debriefings .

  they have full access to working documents .

  consultations have also taken place with groups representing specific sectors of the Canadian economy , including culture , energy , mining , telecommunications and agrifood .

- *French:* $\boldsymbol{f}$

  les fonctionnaires provinciaux sont consultés par appels conférence et comptes rendus .

  ils ont accès à tous les documents de travail .

  les consultations ont aussi eu lieu avec de les groupes représentant de les secteurs précis de le économie canadienne , y compris la culture , le énergie , les mines , les télécommunications et le agro - alimentaire .

- *Word alignments* $\boldsymbol{a}$ *are not included!*

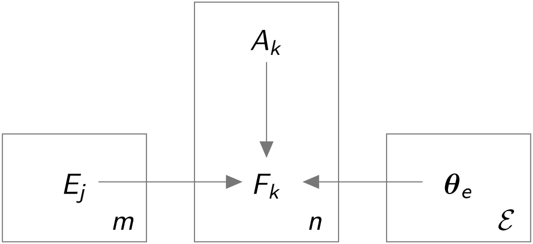
Learning word translations $\boldsymbol{\theta}$ and alignments $\boldsymbol{a}$ with EM

- It is easy to learn translation probabilities $\boldsymbol{\theta}$ from *word-aligned data* $(\boldsymbol{e}, \boldsymbol{f}, \boldsymbol{a})$
- But the available data is only *sentence aligned* $(\boldsymbol{e}, \boldsymbol{f})$
  - $\boldsymbol{a}$ is a *hidden variable*
- This is a perfect problem for Expectation Maximisation!
  - simultaneously learn translation probabilities $\boldsymbol{\theta}$ and word alignments $\boldsymbol{a}$
- It turns out that a very stupid probabilistic model of $\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e})$ (IBM Model 1) plus EM produces good word alignments $\boldsymbol{a}$!
  - IBM developed more sophisticated models, up to IBM Model 5

MACQUARIE UNIVERSITY

The IBM Model 1 generative story

- IBM Model 1 defines $\mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{A}, \boldsymbol{F} \mid \boldsymbol{E})$
  - $\theta_{f,e}$ is *probability of generating French word f when aligned to English word e*
- Each French word $F_k, k = 1, \ldots, n$ is generated independently conditional on English words $\boldsymbol{e} = (e_1, \ldots, e_m)$
- To generate French word F_k given English words $\boldsymbol{e}$:
  - generate an *alignment* $A_k \in 1, \ldots, m$ for F_k uniformly at random
  - generate *French word* F_k from $\theta_{e_{a_k}}$



MACQUARIE UNIVERSITY

Using IBM1 to predict word alignments $\boldsymbol{a}$

- IBM1 *generates all word alignments with same probability*
- But conditional on the English *and French* words, IBM1 generates *non-uniform word alignments*
- Probability of kth French word aligning to j English word:

$$
\begin{aligned}
\mathrm{P}(A_k{=}j \mid \boldsymbol{E}{=}\boldsymbol{e}, F_k{=}f) &= \frac{\mathrm{P}(A_k{=}j, F_k{=}f \mid \boldsymbol{E}{=}\boldsymbol{e})}{\mathrm{P}(F_k{=}f \mid \boldsymbol{E}{=}\boldsymbol{e})} \\
&= \frac{\mathrm{P}(A_k{=}j)\,\mathrm{P}(F_k{=}f \mid E_j{=}e_j)}{\sum_{j'=1}^{m} \mathrm{P}(A_k{=}j')\,\mathrm{P}(F_k{=}f \mid E_{j'}{=}e_{j'})} \\
&= \frac{\theta_{f,e_j}}{\sum_{j'=1}^{m} \theta_{f,e_{j'}}}
\end{aligned}
$$

Example alignment calculation

- English and French strings:

$$\boldsymbol{e} = (\textit{the}, \textit{morning}) \qquad \boldsymbol{f} = (\textit{le}, \textit{matin})$$

- English word to French word translation probabilities:

$\boldsymbol{\theta} =$

	the	*a*	*morning*	*evening*
le	0.7	0.1	0.2	0.2
un	0.1	0.7	0.2	0.2
matin	0.1	0.1	0.3	0.3
soir	0.1	0.1	0.3	0.3

- Alignment probability calculation:

$$\begin{aligned}
\mathrm{P}(A_1{=}1 \mid \boldsymbol{E}{=}(\textit{the}, \textit{morning}), F_1{=}\textit{le}) &= \frac{\theta_{\textit{le},\textit{the}}}{\theta_{\textit{le},\textit{the}} + \theta_{\textit{le},\textit{morning}}} \\
&= 0.7/(0.7+0.2) \\
\mathrm{P}(A_2{=}2 \mid \boldsymbol{E}{=}(\textit{the}, \textit{morning}), F_2{=}\textit{matin}) &= \frac{\theta_{\textit{matin},\textit{morning}}}{\theta_{\textit{matin},\textit{the}} + \theta_{\textit{matin},\textit{morning}}} \\
&= 0.3/(0.1+0.3)
\end{aligned}$$

MACQUARIE UNIVERSITY

"Viterbi" EM for estimating translation probabilities

- We could learn transition probabilities $\boldsymbol{\theta}$ easily *if we had word-aligned data*, but we only have *sentence aligned data*
- Suppose we knew the true $\boldsymbol{\theta}$ (and French really was English + IBM1). We could:
  - use $\boldsymbol{\theta}$ to compute the *most likely alignment* $\widehat{a}_k$ for each French word f_k
  - pretend $\widehat{a}_k$ is the true alignment a_k
  - count the (English word, French word) co-occurences $\widehat{\boldsymbol{n}}$ according to $\widehat{\boldsymbol{a}}$
  - estimate $\widehat{\boldsymbol{\theta}}$ from $\widehat{\boldsymbol{n}}$
- Now suppose $\widehat{\boldsymbol{\theta}}^{(0)}$ is a rough estimate to $\boldsymbol{\theta}$ (and French is English + IBM1)
  - *run this procedure to get a new estimate* $\widehat{\boldsymbol{\theta}}^{(1)}$; maybe it'll be better than $\widehat{\boldsymbol{\theta}}^{(0)}$
- This is called *Viterbi EM* because it uses the *most-likely alignment* $\widehat{\boldsymbol{a}}$

MACQUARIE UNIVERSITY

Viterbi EM example iteration

- English and French strings:

$$\begin{aligned} \boldsymbol{e} &= ((\textit{the}, \textit{morning}), (\textit{the}, \textit{evening})) \\ \boldsymbol{f} &= ((\textit{le}, \textit{matin}), (\textit{le}, \textit{soir})) \end{aligned}$$

- English word to French word translation probabilities:

$$\widehat{\boldsymbol{t}}^{(0)} = \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 0.7 & 0.4 & 0.4 \\ \textit{matin} & 0.2 & 0.3 & 0.3 \\ \textit{soir} & 0.1 & 0.3 & 0.3 \end{array}$$

- Maximum probability alignments:
  the $\rightarrow$ *le* (twice), *morning* $\rightarrow$ *matin*, *evening* $\rightarrow$ *soir*
- Counts derived from these alignments:

$$\widehat{\boldsymbol{n}} = \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 2 & 0 & 0 \\ \textit{matin} & 0 & 1 & 0 \\ \textit{soir} & 0 & 0 & 1 \end{array}$$

MACQUARIE UNIVERSITY

Viterbi EM example iteration (cont.)

- Counts derived from the alignments:

$$\widehat{\boldsymbol{n}} = \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 2 & 0 & 0 \\ \textit{matin} & 0 & 1 & 0 \\ \textit{soir} & 0 & 0 & 1 \end{array}$$

- Normalise counts to update $\mathrm{P}(f|e)$ probability estimates:

$$\widehat{\boldsymbol{t}}^{(1)} = \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 1.0 & 0 & 0 \\ \textit{matin} & 0 & 1.0 & 0 \\ \textit{soir} & 0 & 0 & 1.0 \end{array}$$

$\Rightarrow$ Resolved translation ambiguity for *morning* and *evening*

MACQUARIE UNIVERSITY

Problems with Viterbi EM

- Viterbi EM is *too optimistic* about the alignments
  - Our translation probability estimates $\widehat{\boldsymbol{\theta}}^{(i)}$ express our uncertainty about true alignments
  - But Viterbi EM assumes the most likely alignment is correct, and all others are wrong
- Because Viterbi EM makes a "hard" choice about alignments, it can "get stuck" at a suboptimal alignment
  - k-means clustering is a kind of Viterbi EM procedure
  - There are "real EM" generalisations of the k-means algorithm
- "Real" EM *doesn't commit to a single alignment* like Viterbi EM does
- But in some applications Viterbi EM uses much less memory than "real" EM, so Viterbi EM is all we can do!

MACQUARIE UNIVERSITY

From Viterbi EM to EM

- The probability of *aligning kth French word to jth English word:*

$$\mathrm{P}(A_k{=}j \mid \boldsymbol{E}{=}\boldsymbol{e}, F_k{=}f) \quad = \quad \frac{\theta_{f,e_j}}{\sum_{j'=1}^{m} \theta_{f,e_{j'}}}$$

- Viterbi EM assumes most probable alignment $\widehat{a}_k$ is true alignment
- EM distributes *fractional counts* according to $\mathrm{P}(A_k{=}j \mid \boldsymbol{E}, F_k)$
- Thought experiment: imagine $\boldsymbol{e} = (\textit{morning evening})$, $\boldsymbol{f} = (\textit{matin soir})$ *occurs 1,000 times in our corpus*
- Suppose our current model $\hat{\boldsymbol{\theta}}$ says $\mathrm{P}(\textit{matin} \to \textit{evening}) = 0.6$ and $\mathrm{P}(\textit{matin} \to \textit{morning}) = 0.4$
- Viterbi EM gives all 1,000 counts to $(\textit{matin}, \textit{evening})$
- EM gives 600 counts to $(\textit{matin}, \textit{evening})$ and 400 counts to $(\textit{matin}, \textit{morning})$

MACQUARIE UNIVERSITY

The EM algorithm for estimating translation probabilities

- The EM algorithm for estimating English word to French word translation probabilities $\boldsymbol{\theta}$:

  Initialise $\widehat{\boldsymbol{\theta}}^{(0)}$ somehow (e.g., randomly)

  For iterations $i = 1, 2, \ldots,$:

  E-step: compute the *expected counts* $\widehat{\boldsymbol{n}}^{(i)}$ using $\widehat{\boldsymbol{\theta}}^{(i-1)}$

  M-step: set $\widehat{\boldsymbol{\theta}}^{(i+1)}$ to the MLE for $\boldsymbol{\theta}$ given $\widehat{\boldsymbol{n}}^{(i)}$

- Recall: the MLE (Maximum Likelihood Estimate) for $\boldsymbol{\theta}$ is the relative frequency

- The EM algorithm is *guaranteed to converge to a local maximum*

The E-step: calculating the expected counts

Clear $\widehat{\boldsymbol{n}}$

For each sentence $(\boldsymbol{f}, \boldsymbol{e})$ in training data:

for each French word position $k = 1, \ldots, |\boldsymbol{f}|$:

for each English word position $j = 1, \ldots, |\boldsymbol{e}|$:

$\widehat{n}_{f_k, e_j} += \mathrm{P}(A_k{=}j \mid \boldsymbol{E}{=}\boldsymbol{e}, F_k = f_k)$

Return $\widehat{\boldsymbol{n}}$

- Recall that:

$$\mathrm{P}(A_k{=}j \mid \boldsymbol{E}{=}\boldsymbol{e}, F_k{=}f) \quad = \quad \frac{\theta_{f, e_j}}{\sum_{j'=1}^{m} \theta_{f, e_{j'}}}$$

EM example iteration

- English word to French word translation probabilities:

$$\hat{\boldsymbol{t}}^{(0)} = \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 0.7 & 0.4 & 0.4 \\ \textit{matin} & 0.2 & 0.3 & 0.3 \\ \textit{soir} & 0.1 & 0.3 & 0.3 \end{array}$$

- Probability of French to English alignments $\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F})$
  - Sentence 1: $\boldsymbol{e} = (\textit{the}, \textit{morning})$, $\boldsymbol{f} = (\textit{le}, \textit{matin})$

$$\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F}) = \begin{array}{c|cc} & \textit{le} & \textit{matin} \\ \hline \textit{the} & 0.64 & 0.4 \\ \textit{morning} & 0.36 & 0.6 \end{array}$$

  - Sentence 2: $\boldsymbol{e} = (\textit{the}, \textit{evening})$, $\boldsymbol{f} = (\textit{le}, \textit{soir})$

$$\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F}) = \begin{array}{c|cc} & \textit{le} & \textit{soir} \\ \hline \textit{the} & 0.64 & 0.25 \\ \textit{evening} & 0.36 & 0.75 \end{array}$$

EM example iteration (cont.)

- Probability of French to English alignments $\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F})$
  - Sentence 1: $\boldsymbol{e} = (\textit{the}, \textit{morning})$, $\boldsymbol{f} = (\textit{le}, \textit{matin})$

$$\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F}) \quad = \quad \begin{array}{c|cc} & \textit{le} & \textit{matin} \\ \hline \textit{the} & 0.64 & 0.4 \\ \textit{morning} & 0.36 & 0.6 \end{array}$$

  - Sentence 2: $\boldsymbol{e} = (\textit{the}, \textit{evening})$, $\boldsymbol{f} = (\textit{le}, \textit{soir})$

$$\mathrm{P}(\boldsymbol{A} \mid \boldsymbol{E}, \boldsymbol{F}) \quad = \quad \begin{array}{c|cc} & \textit{le} & \textit{soir} \\ \hline \textit{the} & 0.64 & 0.25 \\ \textit{evening} & 0.36 & 0.75 \end{array}$$

- Expected counts derived from these alignments:

$$\widehat{\boldsymbol{n}} \quad = \quad \begin{array}{c|ccc} & \textit{the} & \textit{morning} & \textit{evening} \\ \hline \textit{le} & 1.28 & 0.36 & 0.36 \\ \textit{matin} & 0.4 & 0.6 & 0 \\ \textit{soir} & 0.25 & 0 & 0.75 \end{array}$$

EM example iteration (cont.)

- Expected counts derived from these alignments:

$\widehat{\boldsymbol{n}} =$

	the	*morning*	*evening*
le	1.28	0.36	0.36
matin	0.4	0.6	0
soir	0.25	0	0.75

- Normalise counts to estimate English word to French word probability estimates:

$\widehat{\boldsymbol{t}}^{(1)} =$

	the	*morning*	*evening*
le	0.66	0.38	0.32
matin	0.21	0.62	0
soir	0.13	0	0.68

$\Rightarrow$ Resolved translation ambiguity for *morning* and *evening*

MACQUARIE UNIVERSITY

Determining convergence of the EM algorithm

- It's possible to prove that an EM iteration *never decreases the likelihood of the data*
  - the *likelihood* is the *probability of the training data* under the current model
  - usually the likelihood increases rapidly with the first few iterations, and then starts decreasing much slower
  - *often people just run 10 EM iterations*
- Tracing the likelihood is a good way of debugging an EM implementation
  - the theorem says "likelihood *never* decreases"
  - but the likelihood can get *extremely small*

  $\Rightarrow$ to avoid underflow, calculate $-$ *log likelihood* (which should *decrease* on every iteration)
- It's easy to calculate the likelihood while calculating the expected counts (see next slide)

Calculating the likelihood

- Recall: the *probability of French word $F_k = f$* is:

$$\mathrm{P}(F_k{=}f \mid \boldsymbol{E}{=}\boldsymbol{e}) \quad = \quad \frac{1}{|\boldsymbol{e}|} \sum_{j=1}^{|\boldsymbol{e}|} \theta_{f,e_j}$$

- You need $\sum_{j=1}^{|\boldsymbol{e}|} \theta_{f_k,e_j}$ to calculate the alignment probabilities anyway

$$\mathrm{P}(A_k{=}j \mid \boldsymbol{E}{=}\boldsymbol{e}, F_k{=}f_k) \quad = \quad \frac{\theta_{f,e_j}}{\sum_{j'=1}^{m} \theta_{f,e_{j'}}}$$

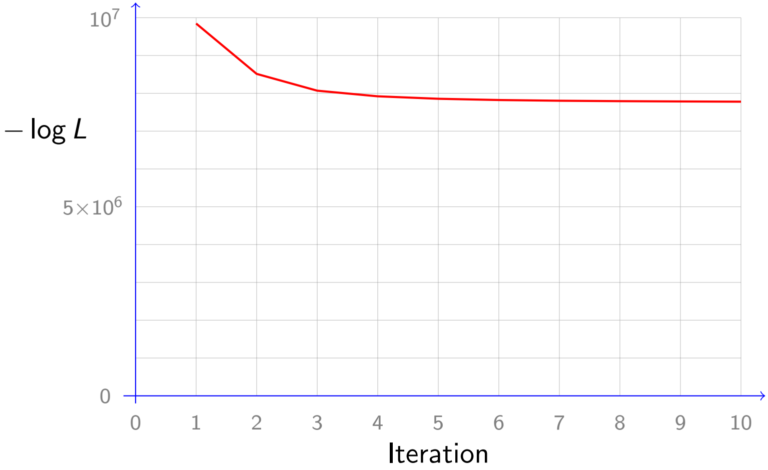
- The negative log likelihood is:

$$\begin{aligned} -\log L \quad &= \quad \sum_{(\boldsymbol{e},\boldsymbol{f})\in D} \sum_{k=1}^{|\boldsymbol{f}|} -\log \mathrm{P}(F_k{=}f_k \mid \boldsymbol{E}{=}\boldsymbol{e}) \\ &= \quad \sum_{(\boldsymbol{e},\boldsymbol{f})\in D} \sum_{k=1}^{|\boldsymbol{f}|} -\log \frac{1}{|\boldsymbol{e}|} \sum_{j=1}^{|\boldsymbol{e}|} \theta_{f_k,e_j} \end{aligned}$$

where the first sum is over all the sentence pairs in the training data

MACQUARIE UNIVERSITY

IBM1 – log likelihood on Hansards corpus



Alignments found by IBM1 from Hansards corpus

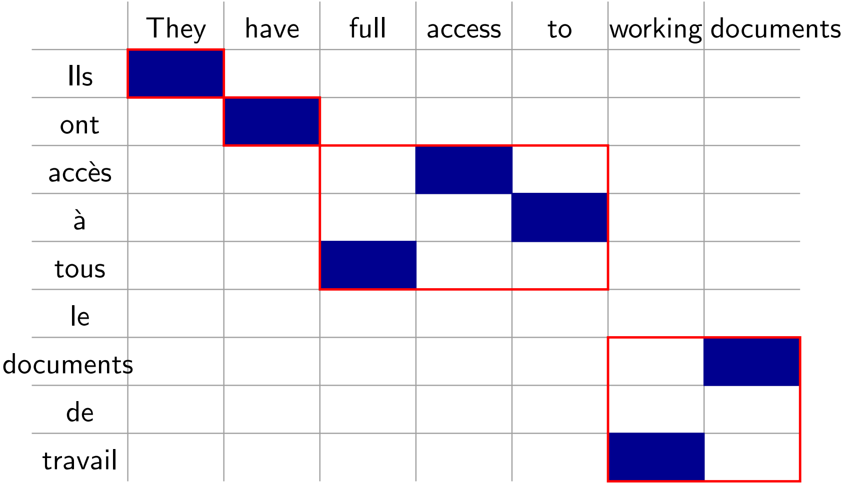
the	le	0.36	add	ajouter	0.32
to	à	0.29	claims	revendications	0.35
of	de	0.58	achieve	atteindre	0.21
and	et	0.79	else	autre	0.37
in	dans	0.24	quality	qualité	0.77
that	que	0.49	encourage	encourager	0.28
a	un	0.42	adopted	adoptées	0.16
is	est	0.53	success	succès	0.60
i	je	0.79	representatives	représentants	0.70
it	il	0.32	gave	a	0.30
legislation	loi	0.45	vinyl	vinyle	0.29
federal	fédéral	0.69	continuous	maintient	0.06
c	c	0.69	tractor	est	0.36
first	première	0.37	briefs	mémoires	0.19
plan	régime	0.58	unethical	ni	0.21
any	ne	0.16	rcms	mrc	0.25
only	seulement	0.29	specifies	montré	0.05
must	doit	0.26	proportionately	proportionnellement	0.32
could	pourrait	0.29	videos	vidéos	0.23
how	comment	0.43	girlfriend	amie	0.15

MACQUARIE UNIVERSITY

From IBM1 to phrase-based translation models

- IBM model 1 over-simplifies in many respects:
  - it translates each word independently
  - $\Rightarrow$ translate multi-word "phrases" rather than words
  - it doesn't model word reordering, i.e., $\mathrm{P}(A_k \mid \boldsymbol{E})$ is uniform
  - $\Rightarrow$ alignments should depend on:
    - location k of French word in sentence
    - alignments of neighbouring French words
  - it doesn't model "fertility", i.e., check that each English word is translated approximately once
- Modern statistical MT systems correct these problems
- Interestingly, IBM1 still plays a central role in modern SMT *because it is not bad at word alignment*
  - alignments are more reliable if you run IBM1 in both directions (i.e., $\boldsymbol{e} \rightarrow \boldsymbol{f}$ and $\boldsymbol{f} \rightarrow \boldsymbol{e}$) and merge the results
  - alignments are useful for *identifying "phrases"* for phrase-based translation
- A phrase-based translation system is similiar to a word-based system, except that the tokens are larger

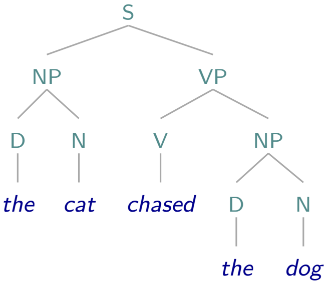
Identifying "phrases" given word alignments



Outline

Syntactic phrase structure and parsing

- Words compose to form *phrases*, which recursively compose to form larger phrases and sentences
  - this recursive structure can be represented by a tree
  - to *"parse" a sentence* means to identify its structure
- Each phrase has a *syntactic category*



- Phrase structure helps identify *semantic roles*, i.e., *who did what to whom*
  - Entities are typically noun phrases
  - Propositions are often represented by sentences
- Syntactic parsing is currently used for:
  - named entity recognition and classification
  - machine translation
  - automatic summarisation

Outline

MACQUARIE UNIVERSITY

Context-Free Grammars

- *Context-Free Grammars* (CFGs) are a simple formal model of compositional syntax
- A probabilistic version of CFG is easy to formulate
- CFG parsing algorithms are comparatively simple
- We know that natural language is not context-free
  - $\Rightarrow$ more complex models, such as Chomsky's *transformational grammar*
- But by *splitting nonterminal labels* PCFGs can approximate natural language fairly well
- There are *efficient dynamic-programming algorithms* for Probabilistic Context Free Grammar inference *that can't be expressed as graphical model inference algorithms* (as far as I know)

Informal presentation of CFGs

Grammar rules

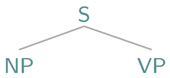
S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V NP
Det $\rightarrow$ the
Det $\rightarrow$ a
N $\rightarrow$ cat
N $\rightarrow$ dog
V $\rightarrow$ chased
V $\rightarrow$ liked

Parse tree

S

Informal presentation of CFGs

Parse tree



Grammar rules

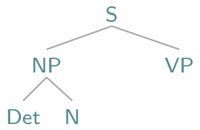
S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V NP
Det $\rightarrow$ the
Det $\rightarrow$ a
N $\rightarrow$ cat
N $\rightarrow$ dog
V $\rightarrow$ chased
V $\rightarrow$ liked

Informal presentation of CFGs

Grammar rules

S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V NP
Det $\rightarrow$ the
Det $\rightarrow$ a
N $\rightarrow$ cat
N $\rightarrow$ dog
V $\rightarrow$ chased
V $\rightarrow$ liked

Parse tree



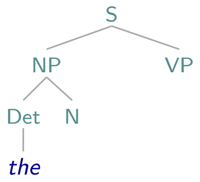
MACQUARIE UNIVERSITY

Informal presentation of CFGs

Grammar rules

S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V NP
Det $\rightarrow$ the
Det $\rightarrow$ a
N $\rightarrow$ cat
N $\rightarrow$ dog
V $\rightarrow$ chased
V $\rightarrow$ liked

Parse tree

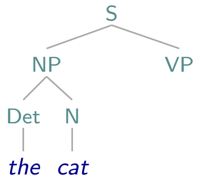


Informal presentation of CFGs

Grammar rules

S → NP VP
NP → Det N
VP → V NP
Det → the
Det → a
N → cat
N → dog
V → chased
V → liked

Parse tree

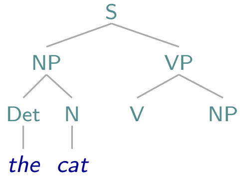


Informal presentation of CFGs

Grammar rules

S → NP VP
NP → Det N
VP → V NP
Det → the
Det → a
N → cat
N → dog
V → chased
V → liked

Parse tree

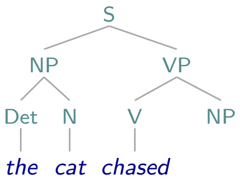


Informal presentation of CFGs

Grammar rules

S → NP VP
NP → Det N
VP → V NP
Det → the
Det → a
N → cat
N → dog
V → chased
V → liked

Parse tree

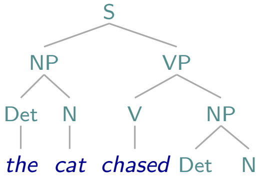


Informal presentation of CFGs

Grammar rules

S → NP VP
NP → Det N
VP → V NP
Det → the
Det → a
N → cat
N → dog
V → chased
V → liked

Parse tree

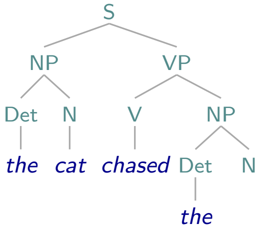


Informal presentation of CFGs

Grammar rules

$S \rightarrow NP\ VP$
$NP \rightarrow Det\ N$
$VP \rightarrow V\ NP$
$Det \rightarrow the$
$Det \rightarrow a$
$N \rightarrow cat$
$N \rightarrow dog$
$V \rightarrow chased$
$V \rightarrow liked$

Parse tree



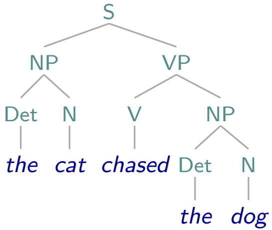
MACQUARIE UNIVERSITY

Informal presentation of CFGs

Grammar rules

S → NP VP
NP → Det N
VP → V NP
Det → the
Det → a
N → cat
N → dog
V → chased
V → liked

Parse tree



How to check if a CFG generates a tree

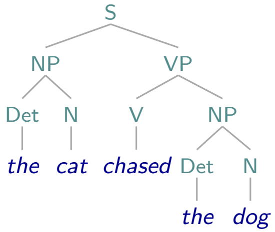
- A CFG $G = (N, V, R, S)$ *generates* a *labelled, finite, ordered tree* t iff:
  - t's *root node* is labelled S,
  - for every node n in t labelled with a terminal $v \in V$, n has no children
  - for every node n in t labelled with a nonterminal $A \in N$, there is a rule $A \to \alpha \in R$ such that α is the sequence of labels of n's children

$$[N = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{Det}, \mathrm{N}, \mathrm{V}\}$$

"N"s are different!

$$V = \{\mathrm{the}, \mathrm{a}, \mathrm{cat}, \mathrm{dog}, \mathrm{chased}, \mathrm{liked}\}$$

$$R = \left\{ \begin{array}{ll} \mathrm{S} \to \mathrm{NP\ VP} & \mathrm{NP} \to \mathrm{Det\ N} \\ \mathrm{VP} \to \mathrm{V} & \mathrm{VP} \to \mathrm{V\ NP} \\ \mathrm{Det} \to \mathrm{a} & \mathrm{Det} \to \mathrm{the} \\ \mathrm{N} \to \mathrm{cat} & \mathrm{N} \to \mathrm{dog} \\ \mathrm{V} \to \mathrm{chased} & \mathrm{V} \to \mathrm{liked} \end{array} \right\}$$

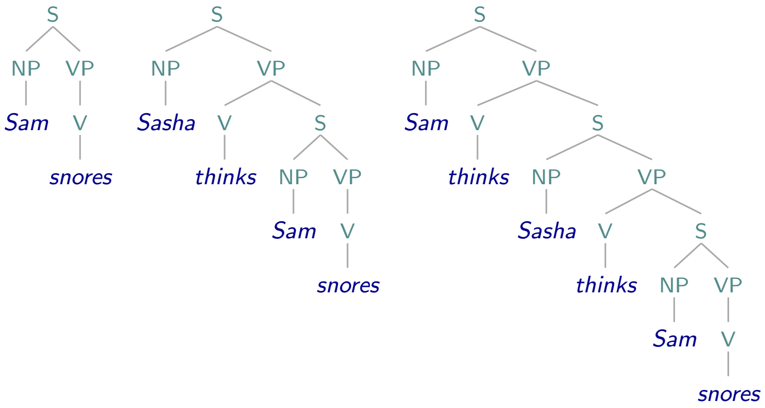


- A CFG G *generates a string of terminals* w iff w is the *terminal yield* of a tree that G generates
  - E.g., this CFG generates *the cat chased the dog*

MACQUARIE UNIVERSITY

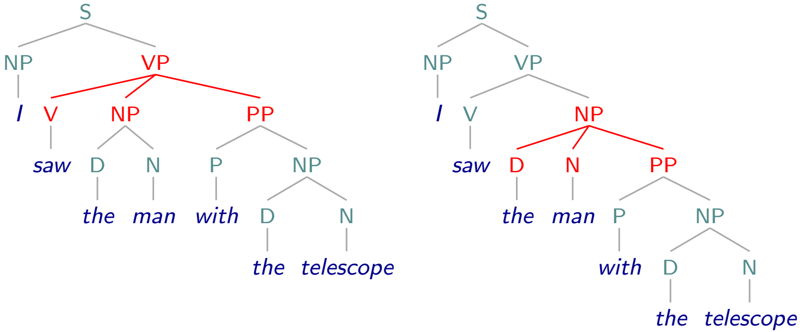
CFGs can generate infinitely many trees

$$
\begin{array}{llll}
S \rightarrow NP\ VP & VP \rightarrow V & VP \rightarrow V\ S & \\
NP \rightarrow Sam & NP \rightarrow Sasha & V \rightarrow thinks & V \rightarrow snores
\end{array}
$$



Syntactic ambiguity

- *Ambiguity* is pervasive in human languages



- Grammars can generate multiple trees with the same *terminal yield*

⇒ A *combinatorial explosion* in the number of parses

  - number of parses usually is an *exponential function of sentence length*
  - some of our grammars generate more that 10^{100} parses for some sentences

MACQUARIE UNIVERSITY

What is "context free" about a CFG?

- Grammars were originally viewed as *string rewriting systems*
- A rule $\alpha \to \beta$ permits a string α to rewrite to string β

S $\to$ NP VP		S
NP $\to$ dogs	$\Rightarrow$	NP VP
VP $\to$ V	$\Rightarrow$	*dogs* VP
V $\to$ bark	$\Rightarrow$	*dogs* V
	$\Rightarrow$	*dogs bark*

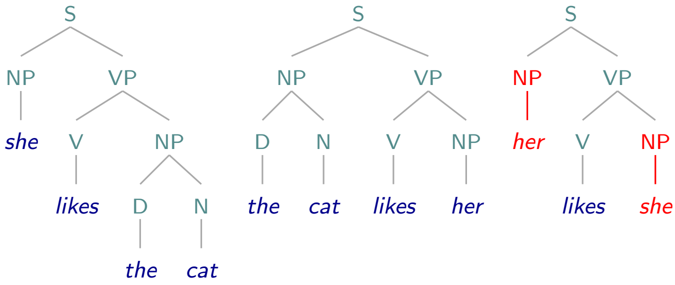
- The *Chomsky hierarchy* of grammars is based on the shapes of α and β
  - *Unrestricted:* no restriction on α or β, undecidable recognition
  - *Context sensitive:* $|\alpha| \leq |\beta|$, PSPACE-complete recognition
  - *Context free:* $|\alpha| = 1$, polynomial-time recognition
  - *Regular:* $|\alpha| = 1$, only one nonterminal at right edge in β, linear time recognition (finite state machines)
- Context sensitive and unrestricted grammars don't have much application in NLP
- The *mildly context-sensitive hierarchy* lies between context-free and context-sensitive

MACQUARIE UNIVERSITY

Context-free grammars can over-generate

- In a CFG, the *possible expansions* of a node depend *only on its label*
  - how one node expands *does not influence how other nodes expand*
  - the label is the "state" of a CFG
- Example: the following grammar over-generates

S → NP VP	NP → D N	VP → V NP
NP → she	NP → her	D → the
V → likes	N → cat	



MACQUARIE UNIVERSITY

Outline

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

S

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

S

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

S

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

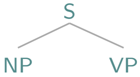
1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

S

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores



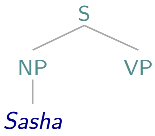
$1.0 \times$

MACQUARIE UNIVERSITY

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

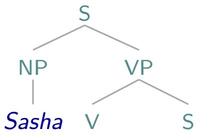


$1.0 \times 0.5 \times$

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

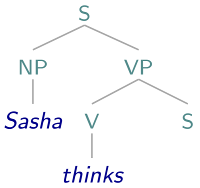


$1.0 \times 0.5 \times 0.2 \times$

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores



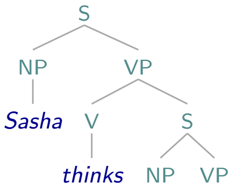
$1.0 \times 0.5 \times 0.2 \times 0.7 \times$

MACQUARIE UNIVERSITY

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores



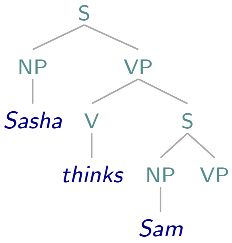
$1.0 \times 0.5 \times 0.2 \times 0.7 \times 1.0 \times$

MACQUARIE UNIVERSITY

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S → NP VP	0.8 VP → V	0.2 VP → V S	
0.5 NP → Sam	0.5 NP → Sasha	0.7 V → thinks	0.3 V → snores



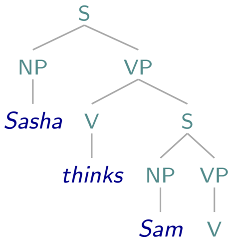
$1.0 \times 0.5 \times 0.2 \times 0.7 \times 1.0 \times 0.5 \times$

MACQUARIE UNIVERSITY

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

1.0 S $\rightarrow$ NP VP	0.8 VP $\rightarrow$ V	0.2 VP $\rightarrow$ V S	
0.5 NP $\rightarrow$ Sam	0.5 NP $\rightarrow$ Sasha	0.7 V $\rightarrow$ thinks	0.3 V $\rightarrow$ snores

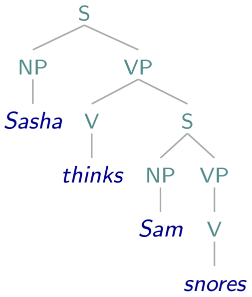


$1.0 \times 0.5 \times 0.2 \times 0.7 \times 1.0 \times 0.5 \times 0.8 \times$

Introduction to Probabilistic Context-Free Grammars

- Intuitive description of *Probabilistic Context-Free Grammars* (PCFGs):
  - *rules have probabilities*
  - the *probability of a tree* is the *product of the probability of the rules that generated it*
- Example:

$1.0\ \mathrm{S} \rightarrow \mathrm{NP\ VP}$	$0.8\ \mathrm{VP} \rightarrow \mathrm{V}$	$0.2\ \mathrm{VP} \rightarrow \mathrm{V\ S}$	
$0.5\ \mathrm{NP} \rightarrow \mathrm{Sam}$	$0.5\ \mathrm{NP} \rightarrow \mathrm{Sasha}$	$0.7\ \mathrm{V} \rightarrow \mathrm{thinks}$	$0.3\ \mathrm{V} \rightarrow \mathrm{snores}$



$$1.0 \times 0.5 \times 0.2 \times 0.7 \times 1.0 \times 0.5 \times 0.8 \times 0.3 = 0.0084$$

Probabilistic Context-Free Grammars (PCFGs)

- A PCFG is a 5-tuple (N, V, R, S, p) where:
  - (N, V, R, S) is a CFG
  - p maps each rule in R to a value in $[0, 1]$ where for each nonterminal $A \in N$:

$$\sum_{A \to \alpha \in R_A} p_{A \to \alpha} \quad = \quad 1.0$$

  where R_A is the subset of rules in R expanding A

- Example:

1.0	S → NP VP		
0.8	VP → V	0.2	VP → V S
0.5	NP → Sam	0.5	NP → Sasha
0.7	V → thinks	0.3	V → snores

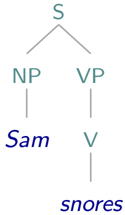
MACQUARIE UNIVERSITY

PCFGs define probability distributions over trees

- A CFG G defines a (possibly infinite) set of trees $\mathcal{T}_G$
- A PCFG G defines a probability $\mathrm{P}_G(t)$ for each $t \in \mathcal{T}_G$
  - $\mathrm{P}(t)$ is the product of the $p_{A \to \alpha}$ of the rules $A \to \alpha$ used to generate t
  - If $n_{A \to \alpha}(t)$ is the number of times rule $A \to \alpha$ is used in generating t, then

$$\mathrm{P}(t) \quad = \quad \prod_{A \to \alpha \in R} p_{A \to \alpha}^{n_{A \to \alpha}(t)}$$

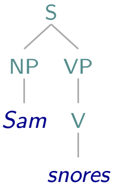
- Example: If t is the following tree:



then $n_{\mathrm{NP} \to \mathrm{Sam}}(t) = 1$ and $n_{\mathrm{V} \to \mathrm{thinks}}(t) = 0$

PCFGs define probability distributions over strings of terminals

- The *yield* of a tree is the sequence of its leaf labels
  - Example:

yield (

) = *Sam snores*

- If x is a string of terminals, let $\mathcal{T}_G(x)$ be the subset of trees in $\mathcal{T}_G$ with yield x
- Then the probability of a terminal string x is *the sum of the probabilities of trees with yield x*, i.e.:

$$\mathrm{P}_G(x) \;=\; \sum_{t \in \mathcal{T}_G(x)} \mathrm{P}_G(t)$$

$\Rightarrow$ *PCFGs can be used as language models*

MACQUARIE UNIVERSITY

PCFGs as recursive mixture distributions

- Given the PCFG rule:

$$\mathsf{A} \to \mathsf{B}_1 \;\ldots\; \mathsf{B}_n$$

  the distribution over strings for A is the *concatenation of the product of the distributions* for $\mathsf{B}_1, \ldots, \mathsf{B}_n$
- Given the two PCFGs rules:

$$\mathsf{A} \to \mathsf{B} \qquad \mathsf{A} \to \mathsf{C}$$

  the distribution over strings for A are a *mixture of the distributions* over strings for B and C with weights $p_{\mathsf{A}\to\mathsf{B}}$ and $p_{\mathsf{A}\to\mathsf{C}}$
- A PCFG with the rules:

$$\mathsf{A} \to \mathsf{A}\,\mathsf{B} \qquad \mathsf{A} \to \mathsf{C}$$

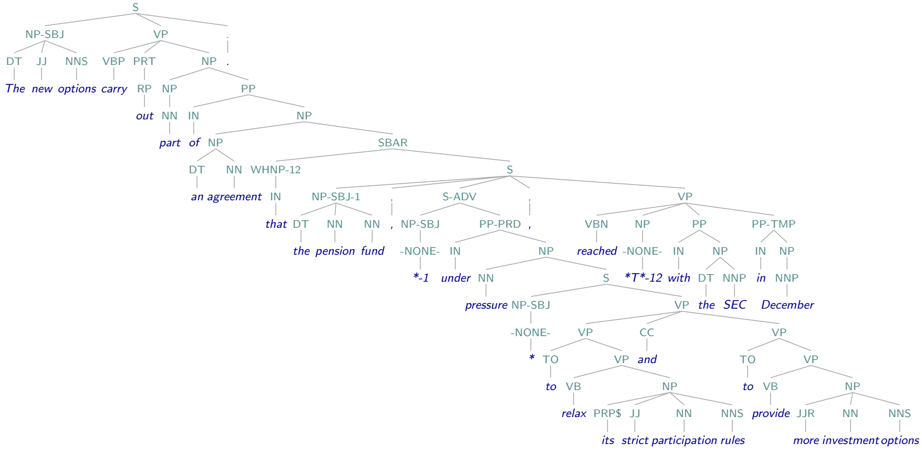
  defines a *recursive mixture distribution* where the strings of A begin with a C followed by zero or more Bs, with probabilities decaying as an exponential function of the number of Bs.

Outline

MACQUARIE UNIVERSITY

Treebank corpora contain phrase-structure analyses

- A treebank is a corpus where every sentence has been (manually) parsed
  - the Penn WSJ treebank has parses for 49,000 sentences



Learning PCFGs from treebanks

- Learning a PCFG from a treebank D
  - Count how often each rule $A \to \alpha$ and each nonterminal A appears in D
  - Relative frequency a.k.a. Maximum Likelihood estimator:

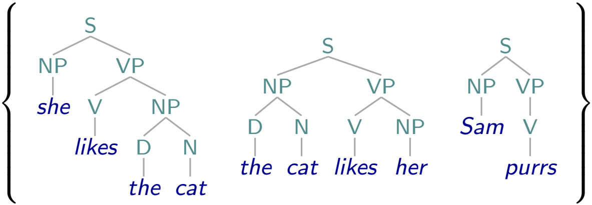
$$\begin{aligned} \hat{p}_{A \to \alpha} &= \frac{n_{A \to \alpha}}{n_A}, \quad \text{where:} \\ n_{A \to \alpha} &= \text{number of times } A \to \alpha \text{ is used in } D \\ n_A &= \text{number of times } A \text{ appears in } D \end{aligned}$$

  - Add-1 smoothed estimator:

$$\begin{aligned} \widehat{\hat{p}}_{A \to \alpha} &= \frac{n_{A \to \alpha} + 1}{n_A + |R_A|}, \quad \text{where:} \\ R_A &= \text{subset of rules } R \text{ that expand nonterminal } A \end{aligned}$$

Learning PCFGs from treebanks example (1)

$D =$



- Nonterminal counts:

$$\begin{array}{rclrclrcl} n_{\mathrm{S}} & = & 3 & n_{\mathrm{NP}} & = & 5 & n_{\mathrm{VP}} & = & 3 \\ n_{\mathrm{D}} & = & 2 & n_{\mathrm{N}} & = & 2 & n_{\mathrm{V}} & = & 3 \end{array}$$

- Rule counts:

$$\begin{array}{rclrclrcl} n_{\mathrm{S}\rightarrow\mathrm{NP\ VP}} & = & 3 & n_{\mathrm{VP}\rightarrow\mathrm{V\ NP}} & = & 2 & n_{\mathrm{VP}\rightarrow\mathrm{V}} & = & 1 \\ n_{\mathrm{NP}\rightarrow\mathrm{D\ N}} & = & 2 & n_{\mathrm{NP}\rightarrow\mathrm{she}} & = & 1 & n_{\mathrm{NP}\rightarrow\mathrm{her}} & = & 1 \\ n_{\mathrm{NP}\rightarrow\mathrm{Sam}} & = & 1 & n_{\mathrm{D}\rightarrow\mathrm{the}} & = & 2 & n_{\mathrm{N}\rightarrow\mathrm{cat}} & = & 2 \\ n_{\mathrm{V}\rightarrow\mathrm{likes}} & = & 2 & n_{\mathrm{V}\rightarrow\mathrm{purrs}} & = & 1 & & & \end{array}$$

MACQUARIE UNIVERSITY

Learning PCFGs from treebanks example (2)

- Nonterminal counts:

$$\begin{array}{rclrclrcl} n_{\mathrm{S}} &=& 3 & n_{\mathrm{NP}} &=& 5 & n_{\mathrm{VP}} &=& 3 \\ n_{\mathrm{D}} &=& 2 & n_{\mathrm{N}} &=& 2 & n_{\mathrm{V}} &=& 3 \end{array}$$

- Rule counts:

$$\begin{array}{rclrclrcl} n_{\mathrm{S}\rightarrow\mathrm{NP\ VP}} &=& 3 & n_{\mathrm{VP}\rightarrow\mathrm{V\ NP}} &=& 2 & n_{\mathrm{VP}\rightarrow\mathrm{V}} &=& 1 \\ n_{\mathrm{NP}\rightarrow\mathrm{D\ N}} &=& 2 & n_{\mathrm{NP}\rightarrow\mathrm{she}} &=& 1 & n_{\mathrm{NP}\rightarrow\mathrm{her}} &=& 1 \\ n_{\mathrm{NP}\rightarrow\mathrm{Sam}} &=& 1 & n_{\mathrm{D}\rightarrow\mathrm{the}} &=& 2 & n_{\mathrm{N}\rightarrow\mathrm{cat}} &=& 2 \\ n_{\mathrm{V}\rightarrow\mathrm{likes}} &=& 2 & n_{\mathrm{V}\rightarrow\mathrm{purrs}} &=& 1 & & & \end{array}$$

- Estimated rule probabilities:

$$\begin{array}{rclrclrcl} \hat{p}_{\mathrm{S}\rightarrow\mathrm{NP\ VP}} &=& 3/3 & \hat{p}_{\mathrm{VP}\rightarrow\mathrm{V\ NP}} &=& 2/3 & \hat{p}_{\mathrm{VP}\rightarrow\mathrm{V}} &=& 1/3 \\ \hat{p}_{\mathrm{NP}\rightarrow\mathrm{D\ N}} &=& 2/5 & \hat{p}_{\mathrm{NP}\rightarrow\mathrm{she}} &=& 1/5 & \hat{p}_{\mathrm{NP}\rightarrow\mathrm{her}} &=& 1/5 \\ \hat{p}_{\mathrm{NP}\rightarrow\mathrm{Sam}} &=& 1/5 & \hat{p}_{\mathrm{D}\rightarrow\mathrm{the}} &=& 2/2 & \hat{p}_{\mathrm{N}\rightarrow\mathrm{cat}} &=& 2/2 \\ \hat{p}_{\mathrm{V}\rightarrow\mathrm{likes}} &=& 2/3 & \hat{p}_{\mathrm{V}\rightarrow\mathrm{purrs}} &=& 1/3 & & & \end{array}$$

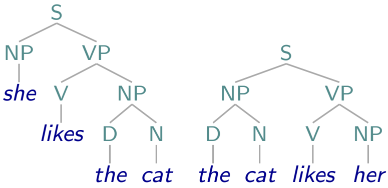
Accuracy of treebank PCFGs

- Parser accuracy is usually measured by *f-score* on a held-out test corpus
- A treebank PCFG (as described above) does fairly poorly (≈ 0.7 f-score)
- Accuracy can be improved by refining the categories
  - wide variety of programmed and fully automatic category-splitting procedures
  - modern PCFG parsers achieve f-score ≈ 0.9
- Category splitting *dramatically increases* the number of categories, and hence rules and parameters in PCFG
  - recall bias-variance tradeoff: category splitting reduces bias, but increases variance

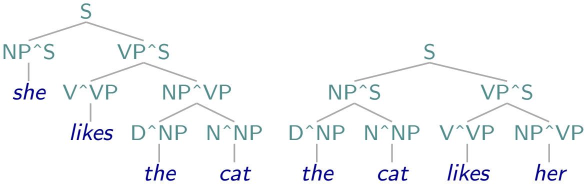
  $\Rightarrow$ smoothing is essential, and details of smoothing procedure make a big impact on parser f-score

Parent annotation of a treebank

- *Parent annotation* is a simple category-splitting procedure where the parent's label is added to every non-terminal label
- Original trees:



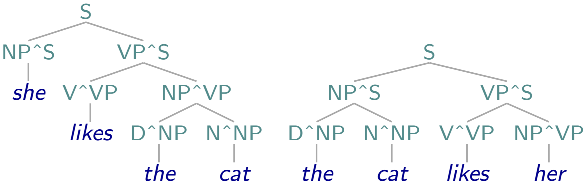
- After parent annotation:



MACQUARIE UNIVERSITY

Why does parent annotation improve parser accuracy?

- After parent annotation:



- Parent annotation adds important linguistic context
  - rules NP $\rightarrow$ she and NP $\rightarrow$ her get replaced with NP^S $\rightarrow$ she and NP^VP $\rightarrow$ her

  $\Rightarrow$ no longer over-generates *her likes she*
- But number of rules grows: the Penn WSJ treebank induces
  - 74,169 rules before parent annotation
  - 93,386 rules after parent annotation
- So sparse data becomes more of a problem after parent annotation

Outline

Goal of PCFG parsing

- Given a PCFG G and a string of terminals x, we want to find the *most probable parse tree* $\hat{t}(x)$ in the set of parses $\mathcal{T}_G(x)$ that G generates for x

$$\widehat{t}(x) \quad = \quad \underset{t \in \mathcal{T}_G(x)}{\operatorname{argmax}} \mathrm{P}_G(t)$$

- Naive algorithm to find $\hat{t}(x)$:
  - enumerate all trees with a terminal yield of length $|x|$
  - if $\text{yield}(t) = x$ and $\mathrm{P}_G(t)$ is greater than probability of best tree seen so far, keep t and $\mathrm{P}_G(t)$
  - return tree with highest probability

MACQUARIE UNIVERSITY

Why is PCFG parsing hard?

- Broad-brush ideas behind probabilistic parsing:
  - to avoid problems of coverage and robustness, *grammar generates all possible parses* (or at least most of the possibly useful ones)
  - probability distribution distinguishes "good" parses from "bad" ones

$\Rightarrow$ Even moderately long sentences have an astronomical number of parses

  - there are sentences in WSJ PTB with over 10^{100} parses

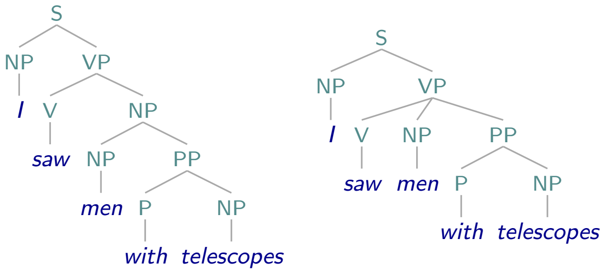
$\Rightarrow$ no hope that parsing via exhaustive enumeration will be practical

High-level overview of PCFG parsing

- All the efficient PCFG parsers I know of involve two steps:
  - *binarise grammar*, i.e., transform it so it has no rules $A \rightarrow \alpha$ where $|\alpha| > 2$
    - this can be done as a *pre-processing step*, or
    - *on-the-fly* as part of the parsing algorithm
  - use *dynamic programming* to search for *optimal parses of substrings of x*
- Together these permit us to parse e.g., 100-word sentences in millions or billions of operations (rather than the 10^{100} that the naive algorithm requires)

PCFG example (used to show parsing algorithm)

$D = \{$



$\}$

$$R = \begin{pmatrix} 1 & \mathrm{S} \to \mathrm{NP\ VP} & 0.5 & \mathrm{VP} \to \mathrm{V\ NP} & 0.5 & \mathrm{VP} \to \mathrm{V\ NP\ PP} \\ 0.29 & \mathrm{NP} \to \mathrm{I} & 0.29 & \mathrm{NP} \to \mathrm{men} & 0.29 & \mathrm{NP} \to \mathrm{telescopes} \\ 0.14 & \mathrm{NP} \to \mathrm{NP\ PP} & 1 & \mathrm{V} \to \mathrm{saw} & 1 & \mathrm{P} \to \mathrm{with} \\ 1 & \mathrm{PP} \to \mathrm{P\ NP} & & & & \end{pmatrix}$$

MACQUARIE UNIVERSITY

Rule binarisation

- Our dynamic programming algorithm requires all rules to have *at most two children*
- Binarisation: replace ternary and longer rules with *a sequence of binary rules*
  - replace rule $p\ A \rightarrow B_1\ B_2\ \ldots\ B_m$ with rules

$$
\begin{array}{ll}
p & A \rightarrow B_1\_B_2\_\ldots\_B_{m-1}\ B_m \\
1.0 & B_1\_B_2\_\ldots\_B_{m-1} \rightarrow B_1\_B_2\_\ldots\_B_{m-2}\ B_{m-1} \\
1.0 & B_1\_B_2 \rightarrow B_1\ B_2
\end{array}
$$

- Example: rule 0.5 VP → V NP PP is replaced with:

$$
\begin{array}{ll}
0.5 & \mathrm{VP} \rightarrow \mathrm{V\_NP\ PP} \\
1.0 & \mathrm{V\_NP} \rightarrow \mathrm{V\ NP}
\end{array}
$$

- This can expand the number of rules in the grammar
  - The WSJ PTB PCFG has:
    - 74,619 rules before binarisation
    - 89,304 rules after binarisation
    - 109,943 rules with both binarisation and parent annotation

PCFG example after binarisation

$$D = \left\{ \begin{array}{cc} [_{\mathrm{S}}\ [_{\mathrm{NP}}\ \textit{I}\,]\ [_{\mathrm{VP}}\ [_{\mathrm{V}}\ \textit{saw}\,]\ [_{\mathrm{NP}}\ [_{\mathrm{NP}}\ \textit{men}\,]\ [_{\mathrm{PP}}\ [_{\mathrm{P}}\ \textit{with}\,]\ [_{\mathrm{NP}}\ \textit{telescopes}\,]\,]\,]\,]\,] & [_{\mathrm{S}}\ [_{\mathrm{NP}}\ \textit{I}\,]\ [_{\mathrm{VP}}\ [_{\mathrm{V\_NP}}\ [_{\mathrm{V}}\ \textit{saw}\,]\ [_{\mathrm{NP}}\ \textit{men}\,]\,]\ [_{\mathrm{PP}}\ [_{\mathrm{P}}\ \textit{with}\,]\ [_{\mathrm{NP}}\ \textit{telescopes}\,]\,]\,]\,] \end{array} \right\}$$

$$R = \left(\begin{array}{llllll} 1 & \mathrm{S} \to \mathrm{NP\ VP} & 0.5 & \mathrm{VP} \to \mathrm{V\ NP} & 0.29 & \mathrm{NP} \to \mathrm{I} \\ 0.29 & \mathrm{NP} \to \mathrm{men} & 0.29 & \mathrm{NP} \to \mathrm{telescopes} & 0.14 & \mathrm{NP} \to \mathrm{NP\ PP} \\ 1 & \mathrm{V} \to \mathrm{saw} & 1 & \mathrm{PP} \to \mathrm{P\ NP} & 1 & \mathrm{P} \to \mathrm{with} \\ 0.5 & \mathrm{VP} \to \mathrm{V\_NP\ \ PP} & 1 & \mathrm{V\_NP} \to \mathrm{V\ NP} & & \end{array} \right)$$

String positions

- String positions are a convenient way of *identifying substrings of a fixed string x*
- Informally, string positions are *integers naming the "spaces" between words*
- If $|x| = n$, the a *string position* for x is an integer between 0 to n inclusive
- If $x = (x_0, \ldots, x_{n-1})$, the *pair of string positions* (i, j), $i \leq j$ identifies substring $x_i, \ldots, x_{j-1}$. *WARNING: 0-based indexing!*
- Example: In the string

|	I	|	saw	|	men	|	with	|	telescopes	|
0		1		2		3		4		5

  the pair of string positions $(1, 4)$ identifies *saw men with*
- Question: what substring does $(0, 5)$ identify?
- Question: what substring does $(2, 2)$ identify?

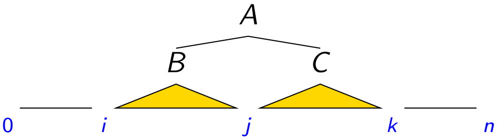
Chomsky-normal form

- We'll assume that our PCFG G is in *Chomsky-normal form* (CNF), i.e., every rule is of the form:
  - $A \to B\ C$, where $A, B, C \in N$ (i.e., A, B, C are nonterminals), or
  - $A \to v$, where $v \in V$ (i.e., A is a nonterminal and v is a terminal)
- Binarisation is a key step in bringing arbitrary PCFGs into CNF
- Our example grammar is in CNF

1	S → NP VP	0.5	VP → V NP	0.29	NP → I
0.29	NP → men	0.29	NP → telescopes	0.14	NP → NP PP
1	V → saw	1	PP → P NP	1	P → with
0.5	VP → V_NP PP	1	V_NP → V NP		

Introduction to dynamic programming parsing

- Key idea: find most probable parse trees with top node A for each substring (i, k) of string x
  - find most probable parse trees for shorter substrings first
  - use these to find most probable parse trees for longer substrings
- If $k = i + 1$, then most probable parse tree is $A \to x_i$
- If $k > i + 1$ then most probable parse tree for A can only be formed by:
  - finding a mid-point j, where $i < j < k$, and
  - combining a most probable parse tree for B spanning (i, j) with
  - a most probable parse tree for C spanning (j, k)
  - using some rule $A \to B\ C \in R$



Dynamic programming parsing algorithm

- Given a PCFG $G = (N, V, R, S, p)$ in CNF and a string x where $|x| = n$, fill a table $Q[A, i, k]$ for each $A \in N$ and $0 \leq i < k \leq n$
  - $Q[A, i, k]$ will be set to the maximum probability of any parse with top node A spanning (i, k)
- Algorithm:

  for each $i = 0, \ldots, n-1$:
  $\quad Q[A, i, i+1] = p_{A \to x_i}$
  for $\ell = 2, \ldots, n$:
  $\quad$ for $i = 0, \ldots, n-\ell$:
  $\quad\quad k = i + \ell$
  $\quad\quad$ for each $A \in N$:
  $$Q[A, i, k] = \max_j \max_{A \to B\ C} p_{A \to B\ C}\ Q[B, i, j]\ Q[C, j, k]$$
  return $Q[S, 0, n]$ (max probability of S)

- In recursion, the midpoint j ranges over $i+1, \ldots, k-1$, and the rule $A \to B\ C$ ranges over all rules with parent A
- Keep *back-pointers* from each $Q[A, i, k]$ to optimal children

MACQUARIE UNIVERSITY

Dynamic programming parsing example

- Grammar in CNF:

1	S → NP VP	0.5	VP → V NP	0.29	NP → I
0.29	NP → men	0.29	NP → telescopes	0.14	NP → NP PP
1	V → saw	1	PP → P NP	1	P → with
0.5	VP → V_NP PP	1	V_NP → V NP		

- String x to parse:

$$ {}_{0}|\ \text{I}\ |_{1}\ \text{saw}\ |_{2}\ \text{men}\ |_{3}\ \text{with}\ |_{4}\ \text{telescopes}\ |_{5} $$

- Base case ($\ell = 1$)

$$\begin{array}{rclrclrcl} Q[\text{NP},0,1] & = & 0.29 & Q[\text{V},1,2] & = & 1 & Q[\text{NP},2,3] & = & 0.29 \\ Q[\text{P},3,4] & = & 1 & Q[\text{NP},4,5] & = & 0.29 & & & \end{array}$$

- Recursive case $\ell = 2$:

$$\begin{array}{rcll} Q[\text{VP},1,3] & = & 0.15 & \text{from } Q[\text{V},1,2] \text{ and } Q[\text{NP},2,3] \\ Q[\text{V\_NP},1,3] & = & 0.29 & \text{from } Q[\text{V},1,2] \text{ and } Q[\text{NP},2,3] \\ Q[\text{PP},3,5] & = & 0.29 & \text{from } Q[\text{P},3,4] \text{ and } Q[\text{NP},4,5] \end{array}$$

Dynamic programming parsing example (cont)

- Recursive case $\ell = 3$:

$$\begin{aligned} Q[\mathrm{S},0,3] &= 0.044 \quad \text{from } Q[\mathrm{NP},0,1] \text{ and } Q[\mathrm{VP},1,3] \\ Q[\mathrm{NP},2,5] &= 0.011 \quad \text{from } Q[\mathrm{NP},2,3] \text{ and } Q[\mathrm{PP},3,5] \end{aligned}$$

- Recursive case $\ell = 4$:

$$Q[\mathrm{VP},1,5] = 0.042 \quad \text{from } Q[\mathrm{V\_NP},1,3] \text{ and } Q[\mathrm{PP},3,5]$$

(alternative parse from $Q[\mathrm{V},1,2]$ and $Q[\mathrm{NP},2,5]$ only has probability 0.0055)

- Recursive case $\ell = 6$:

$$Q[\mathrm{S},0,5] = 0.012 \quad \text{from } Q[\mathrm{NP},0,1] \text{ and } Q[\mathrm{VP},1,5]$$

- By chasing backpointers, we find the following parse:

```
(S (NP I) (VP (V_NP (V saw) (NP men)) (PP (P with) (NP telescopes))))
```

- After removing the "binarisation categories":

```
(S (NP I) (VP (V saw) (NP men) (PP (P with) (NP telescopes))))
```

Running time of dynamic programming parsing

- The dynamic programming parsing algorithm enumerates *all possible string positions* $0 \leq i < j < k \leq n$, where $n = |x|$ is the length of the string to be parsed
- There are $O(n^3)$ of these, so this will take $O(n^3)$ time
- For each possible (i, j, k) triple, it considers all $m = |R|$ rules in the grammar. This takes $O(m)$ time.

$\Rightarrow$ The dynamic programming parsing algorithm runs in $O(m\, n^3)$ time

- This is much better than the exponential time of the naive algorithm, but with large grammars it can still be very slow
- Question: what are the *space requirements* of the algorithm?

MACQUARIE UNIVERSITY

State of the art parsing algorithms

- State-of-the-art syntactic parsers come in two varieties: *phrase structure* and *dependency* parsers
- Phrase structure parsers are often effectively PCFGs with hundreds of thousands of states
  - "coarse to fine" search algorithms using dynamic programming
  - discriminatively trained, with the PCFG probability as a "feature"
- Dependency parsers are often incremental shift-reduce parsers without dynamic programming
  - each move is predicted locally by a classifier
  - beam search to avoid "garden pathing"
- State-of-the-art systems achieve over 90% f-score (accuracy)
- Major internet companies are reported to parse the web several times a day

Outline

Overview of computational linguistics and natural language processing

Key ideas in NLP and CL

Grammars and parsing

Non-parametric Bayesian extensions to grammars

Conclusion and future directions

PCFGs as products of multinomials

- The choice of rule to expand a state in an HMM or a nonterminal in a PCFG is a draw from a multinomial distribution

$\Rightarrow$ HMMs and PCFGs can be viewed as products of multinomial distributions

$\Rightarrow$ Dirichlet distributions are *conjugate priors* for HMMs and PCFGs

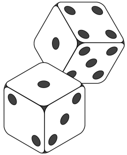
  - Bayesian inference for HMMs and PCFGs generally assumes Dirichlet priors

$\Rightarrow$ Non-parametric generalisations of multinomials should also let us generalise HMMs and PCFGs

MACQUARIE UNIVERSITY

Outline

Bayesian inference for Dirichlet-multinomials



- *Predictive probability* (probability of next event) with *uniform Dirichlet prior* with mass α over m outcomes and observed data $\boldsymbol{Z}_{1:n} = (Z_1, \ldots, Z_n)$

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}, \alpha) \quad \propto \quad n_k(\boldsymbol{Z}_{1:n}) + \alpha/m$$

  where $n_k(\boldsymbol{Z}_{1:n})$ is number of times k appears in $\boldsymbol{Z}_{1:n}$
- Example: Coin ($m = 2$), $\alpha = 1$, $\boldsymbol{Z}_{1:2} = (\text{heads}, \text{heads})$
  - $\mathrm{P}(Z_3 = \text{heads} \mid \boldsymbol{Z}_{1:2}, \alpha) \propto 2.5$
  - $\mathrm{P}(Z_3 = \text{tails} \mid \boldsymbol{Z}_{1:2}, \alpha) \propto 0.5$

Dirichlet-multinomials with many outcomes

- Predictive probability:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}, \alpha) \quad \propto \quad n_k(\boldsymbol{Z}_{1:n}) + \alpha/m$$

- Suppose the number of outcomes $m \gg n$. Then:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}, \alpha) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } n_k(\boldsymbol{Z}_{1:n}) > 0 \\ \alpha/m & \text{if } n_k(\boldsymbol{Z}_{1:n}) = 0 \end{cases}$$

- But *most outcomes will be unobserved*, so:

$$\mathrm{P}(Z_{n+1} \notin \boldsymbol{Z}_{1:n} \mid \boldsymbol{Z}_{1:n}, \alpha) \quad \propto \quad \alpha$$

MACQUARIE UNIVERSITY

From Dirichlet-multinomials to Chinese Restaurant Processes

...

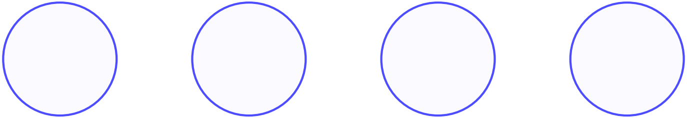
- Suppose *number of outcomes is unbounded* but *we* pick the event labels
- If we number event types in order of occurrence
  $\Rightarrow$ *Chinese Restaurant Process*

$$Z_1 = 1$$

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}, \alpha) \propto \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

MACQUARIE UNIVERSITY

Chinese Restaurant Process (0)



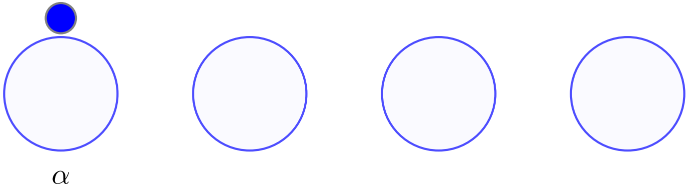
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} =$
- $\mathrm{P}(\boldsymbol{z}) = 1$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

MACQUARIE UNIVERSITY

Chinese Restaurant Process (1)



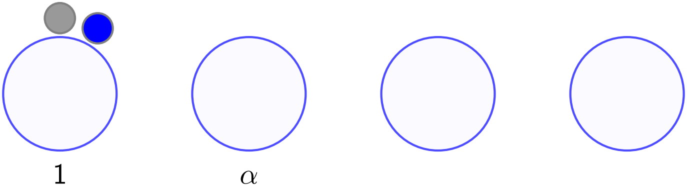
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1$
- $\mathrm{P}(\boldsymbol{z}) = \alpha/\alpha$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

MACQUARIE UNIVERSITY

Chinese Restaurant Process (2)



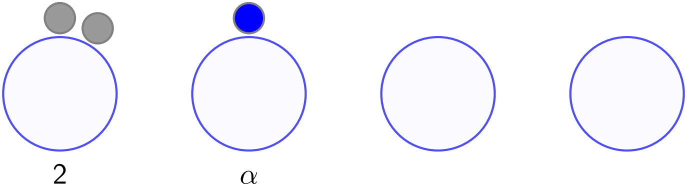
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1$
- $\mathrm{P}(\boldsymbol{z}) = \alpha/\alpha \times 1/(1+\alpha)$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

MACQUARIE UNIVERSITY

Chinese Restaurant Process (3)



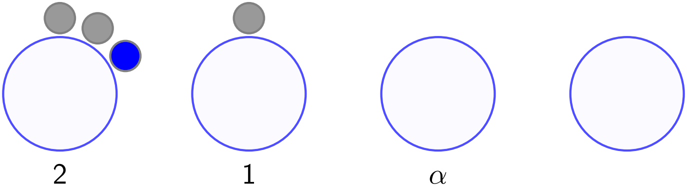
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1, 2$
- $\mathsf{P}(\boldsymbol{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha)$

- Next customer chooses a table according to:

$$\mathsf{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m+1 \end{cases}$$

MACQUARIE UNIVERSITY

Chinese Restaurant Process (4)



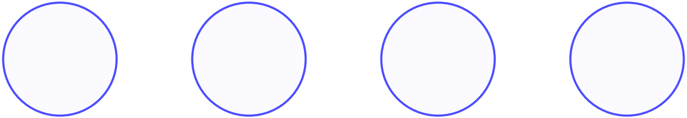
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1, 2, 1$
- $\mathrm{P}(\boldsymbol{z}) = \alpha/\alpha \times 1/(1+\alpha) \times \alpha/(2+\alpha) \times 2/(3+\alpha)$

- Next customer chooses a table according to:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{Z}_{1:n}) \quad \propto \quad \begin{cases} n_k(\boldsymbol{Z}_{1:n}) & \text{if } k \leq m = \max(\boldsymbol{Z}_{1:n}) \\ \alpha & \text{if } k = m + 1 \end{cases}$$

MACQUARIE UNIVERSITY

Labeled Chinese Restaurant Process (0)

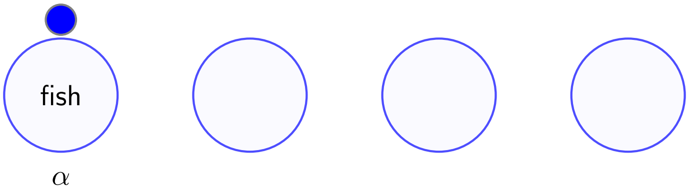


- Table $\rightarrow$ label mapping $\boldsymbol{Y} =$
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} =$
- Output sequence $\boldsymbol{X} =$
- $\mathrm{P}(\boldsymbol{X}) = 1$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* Y_k for each table k
- All customers sitting at table k (i.e., $Z_i = k$) share label Y_k
- Customer i sitting at table Z_i has label $X_i = Y_{Z_i}$

MACQUARIE UNIVERSITY

Labeled Chinese Restaurant Process (1)

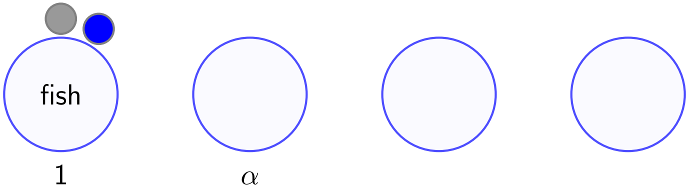


- Table $\rightarrow$ label mapping $\boldsymbol{Y}$ = fish
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1$
- Output sequence $\boldsymbol{X}$ = fish
- $\mathrm{P}(\boldsymbol{X}) = \alpha/\alpha \times \mathrm{P}_0(\text{fish})$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* Y_k for each table k
- All customers sitting at table k (i.e., $Z_i = k$) share label Y_k
- Customer i sitting at table Z_i has label $X_i = Y_{Z_i}$

MACQUARIE UNIVERSITY

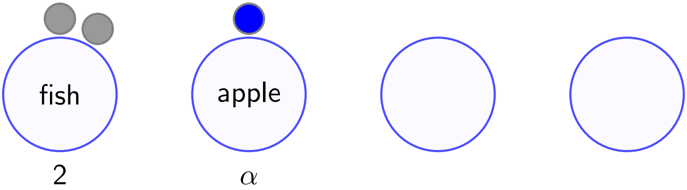
Labeled Chinese Restaurant Process (2)



- Table $\rightarrow$ label mapping $\boldsymbol{Y}$ = fish
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1$
- Output sequence $\boldsymbol{X}$ = fish,fish
- $\mathrm{P}(\boldsymbol{X}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha)$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* Y_k for each table k
- All customers sitting at table k (i.e., $Z_i = k$) share label Y_k
- Customer i sitting at table Z_i has label $X_i = Y_{Z_i}$

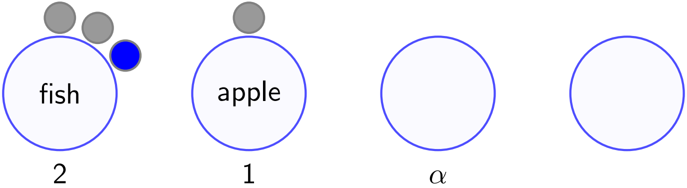
Labeled Chinese Restaurant Process (3)



- Table $\rightarrow$ label mapping $\boldsymbol{Y}$ = fish,apple
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1, 2$
- Output sequence $\boldsymbol{X}$ = fish,fish,apple
- $\mathrm{P}(\boldsymbol{X}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha) \times \alpha/(2+\alpha)\mathrm{P}_0(\text{apple})$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* Y_k for each table k
- All customers sitting at table k (i.e., $Z_i = k$) share label Y_k
- Customer i sitting at table Z_i has label $X_i = Y_{Z_i}$

Labeled Chinese Restaurant Process (4)



- Table $\rightarrow$ label mapping $\boldsymbol{Y}$ = fish,apple
- Customer $\rightarrow$ table mapping $\boldsymbol{Z} = 1, 1, 2$
- Output sequence $\boldsymbol{X}$ = fish,fish,apple,fish
- $\mathrm{P}(\boldsymbol{X}) = \mathrm{P}_0(\text{fish}) \times 1/(1+\alpha) \times \alpha/(2+\alpha)\mathrm{P}_0(\text{apple}) \times 2/(3+\alpha)$

- *Base distribution* $\mathrm{P}_0(Y)$ generates a *label* Y_k for each table k
- All customers sitting at table k (i.e., $Z_i = k$) share label Y_k
- Customer i sitting at table Z_i has label $X_i = Y_{Z_i}$

Summary: Chinese Restaurant Processes

- *Chinese Restaurant Processes* (CRPs) generalize Dirichlet-Multinomials to an *unbounded number of outcomes*
  - *concentration parameter* α controls how likely a new outcome is
  - CRPs exhibit a *rich get richer* power-law behaviour
- *Labeled CRPs* use a *base distribution* to label each table
  - base distribution can have *infinite support*
  - concentrates mass on a countable subset
  - power-law behaviour $\Rightarrow$ Zipfian distributions

Nonparametric extensions of PCFGs

- Chinese restaurant processes are a nonparametric extension of Dirichlet-multinomials because the number of states (occupied tables) depends on the data
- Two obvious nonparametric extensions of PCFGs:
  - let the number of nonterminals grow unboundedly
    - refine the nonterminals of an original grammar
      e.g., $S_{35} \rightarrow NP_{27}\ VP_{17}$

    $\Rightarrow$ infinite PCFG
  - let the number of rules grow unboundedly
    - "new" rules are compositions of several rules from original grammar
    - equivalent to caching tree fragments

    $\Rightarrow$ adaptor grammars
- No reason both can't be done together ...

MACQUARIE UNIVERSITY

Outline

Adaptor grammars: informal description

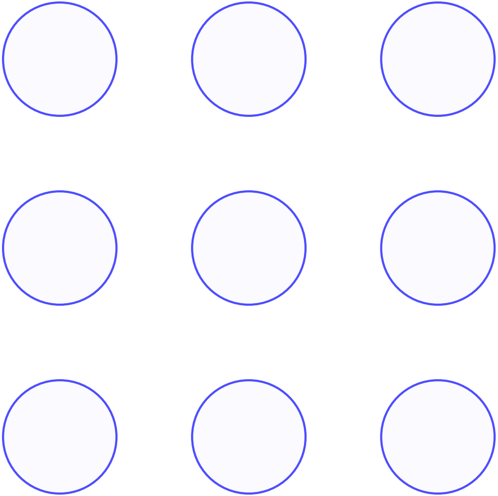
- The trees generated by an adaptor grammar are defined by CFG rules as in a CFG
- A subset of the nonterminals are *adapted*
- *Unadapted nonterminals* expand by picking a rule and recursively expanding its children, as in a PCFG
- *Adapted nonterminals* can expand in two ways:
  - by picking a rule and recursively expanding its children, or
  - by generating a previously generated tree (with probability proportional to the number of times previously generated)
- Implemented by having a CRP for each adapted nonterminal
- The CFG rules of the adapted nonterminals determine the *base distributions* of these CRPs

Adaptor grammar for stem-suffix morphology (0)

$\underline{\text{Word}} \rightarrow$ Stem Suffix

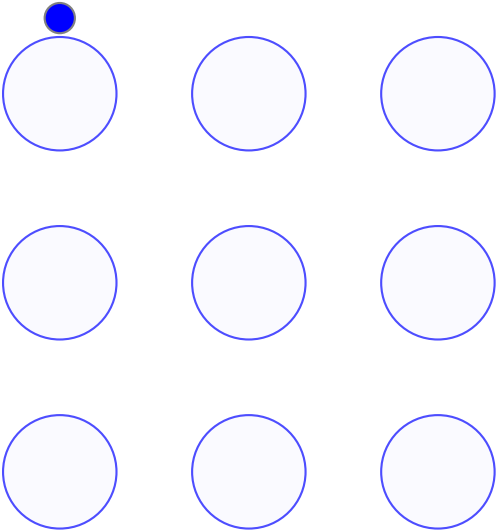
$\underline{\text{Stem}} \rightarrow$ Phoneme^{+}

$\underline{\text{Suffix}} \rightarrow$ Phoneme*



Generated words:

Adaptor grammar for stem-suffix morphology (1a)



Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*

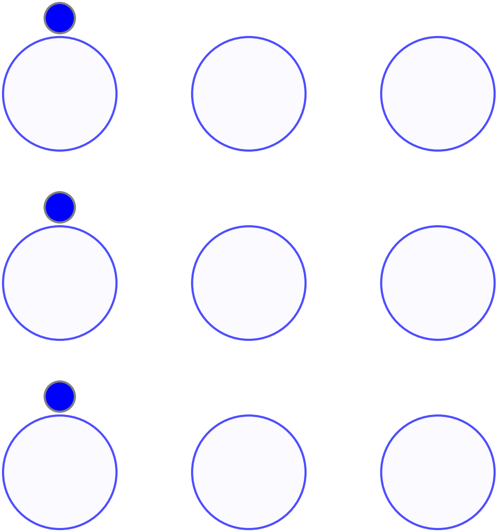
Generated words:

Adaptor grammar for stem-suffix morphology (1b)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme$^+$

Suffix $\rightarrow$ Phoneme*



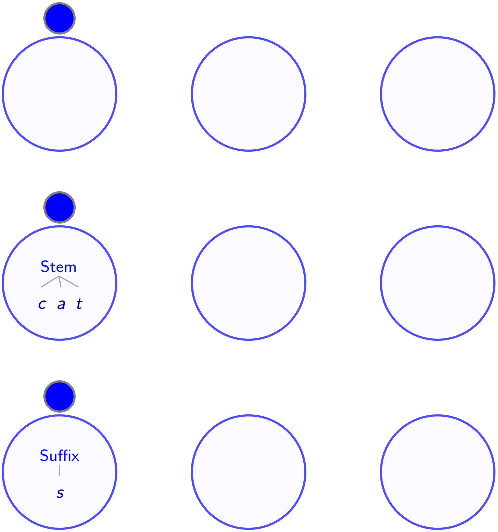
Generated words:

Adaptor grammar for stem-suffix morphology (1c)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



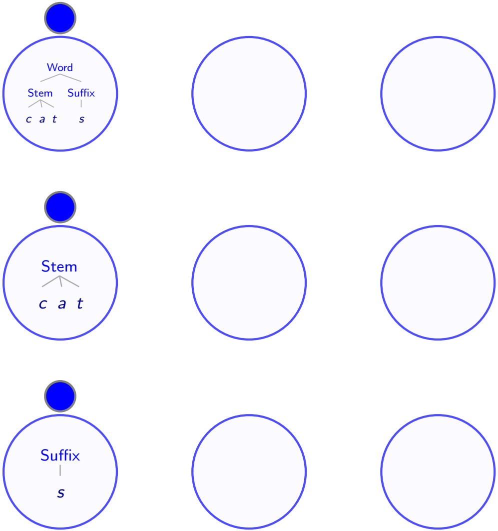
Generated words:

Adaptor grammar for stem-suffix morphology (1d)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



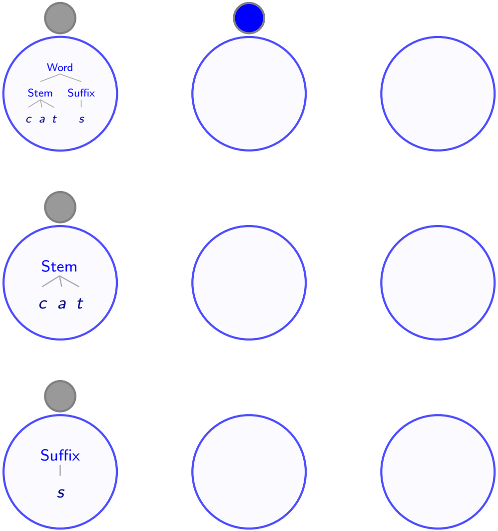
Generated words: cats

Adaptor grammar for stem-suffix morphology (2a)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



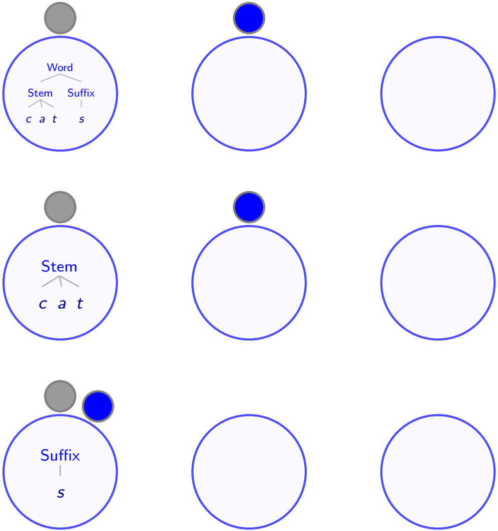
Generated words: cats

Adaptor grammar for stem-suffix morphology (2b)

<u>Word</u> $\rightarrow$ Stem Suffix

<u>Stem</u> $\rightarrow$ Phoneme$^+$

<u>Suffix</u> $\rightarrow$ Phoneme*



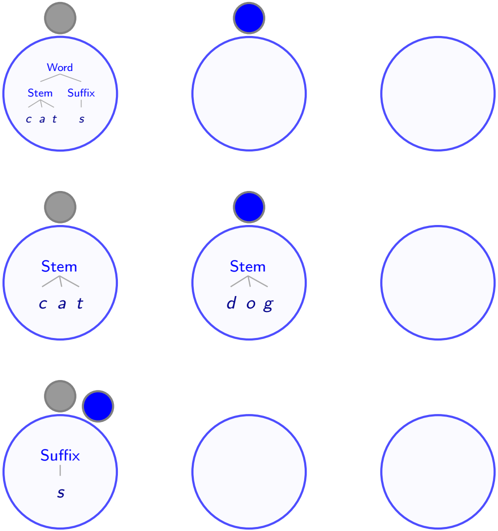
Generated words: cats

Adaptor grammar for stem-suffix morphology (2c)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme$^+$

Suffix $\rightarrow$ Phoneme*



Generated words: cats

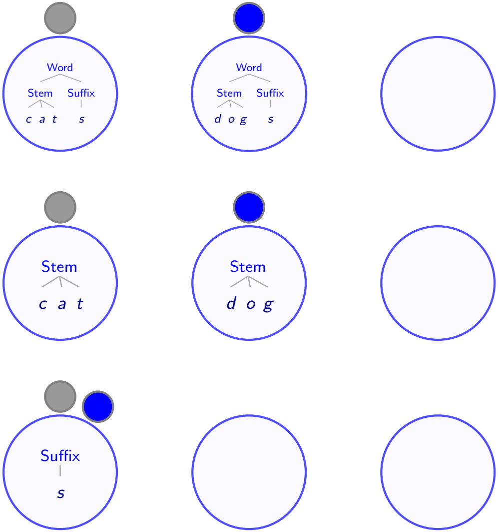
MACQUARIE UNIVERSITY

Adaptor grammar for stem-suffix morphology (2d)

Word → Stem Suffix

Stem → Phoneme$^+$

Suffix → Phoneme*



Generated words: cats, dogs

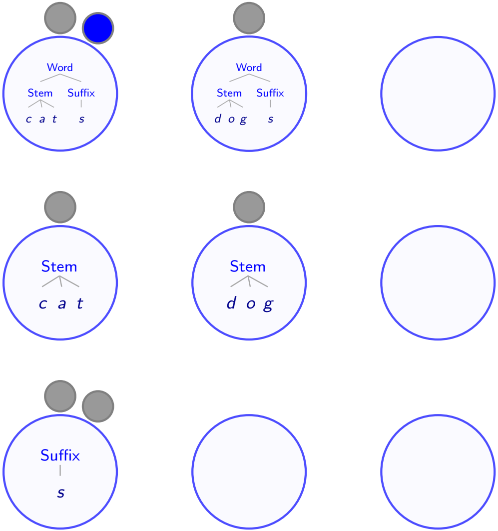
MACQUARIE UNIVERSITY

Adaptor grammar for stem-suffix morphology (3)

Word $\rightarrow$ Stem Suffix

Stem $\rightarrow$ Phoneme^{+}

Suffix $\rightarrow$ Phoneme*



Generated words: cats, dogs, cats

MACQUARIE UNIVERSITY

Adaptor grammars as generative processes

- The sequence of trees generated by an adaptor grammar are *not* independent
  - it *learns* from the trees it generates
  - if an adapted subtree has been used frequently in the past, it's more likely to be used again
- but the sequence of trees is *exchangable* (important for sampling)
- An *unadapted nonterminal* A expands using $A \to \beta$ with probability $\theta_{A\to\beta}$
- Each adapted nonterminal A is associated with a CRP (or PYP) that caches previously generated subtrees rooted in A
- An *adapted nonterminal* A expands:
  - to a subtree τ rooted in A with probability proportional to the number of times τ was previously generated
  - using $A \to \beta$ with probability proportional to $\alpha_A \theta_{A\to\beta}$

Properties of adaptor grammars

- Possible trees are generated by CFG rules
  but the probability of each adapted tree is learned separately
- Probability of adapted subtree τ is proportional to:
  - the number of times τ was seen before
    $\Rightarrow$ "rich get richer" dynamics (Zipf distributions)
  - plus α_A times prob. of generating it via PCFG expansion

$\Rightarrow$ Useful compound structures can be *more probable than their parts*

- PCFG rule probabilities estimated *from table labels*
  $\Rightarrow$ effectively *learns from types*, not tokens
  $\Rightarrow$ makes learner less sensitive to frequency variation in input

Applications of adaptor grammars

- Main application until now has been in *modelling human language acquisition*
  - unsupervised word segmentation
  - exploring the role of
    - information about the non-linguistic context
    - syllabic structure
    - prosodic structure
- By exploiting the *connection between PCFGs and LDA topic models*, we can:
  - develop *topical collocation models*
    e.g., *New York Times*, *White House*
  - learn the *structure of proper names*
    e.g., *Mr Pierre E. van Winken*

Outline

MACQUARIE UNIVERSITY

Summary

- Computational linguistics and natural language processing:
  - were originally inspired by linguistics,
  - but now they are almost applications of machine learning and statistics
- But they are unusual ML applications because they *involve predicting very highly structured objects*
  - phrase structure trees in syntactic parsing
  - entire sentences in speech recognition and machine translation
- We solve these problems using standard methods from machine learning:
  - define a probabilistic model over the relevant variables
  - factor the model into small components that we can learn
  - examples: HMMs, CRFs and PCFGs
- Often the relevant variables are not available in the training data
  - Expectation-Maximisation, MCMC, etc. can *impute the values of the hidden variables*

The future of NLP

- NLP applications are exploding, driven mainly by:
  - the information explosion (much of which is text)
  - the mobile computing revolution (talk with our computers)
- The major internet companies are investing in NLP at a scale not seen before
  - the *knowledge graph* and similiar information repositories provide much richer information than available before
- Topic modelling and opinion mining likely to be widely used to track rapidly changing document collections (e.g., social media)

Good areas for future research

- Improving existing NLP and CL models (parsing, relation extraction, machine translation, etc.)
  - explore new models
  - apply new ideas from machine learning, e.g., deep learning
- Combine and extend models to produce new NLP applications
  - parse speech data (and detect/correct speech disfluencies)
- Find new knowledge sources (e.g., kinds of training data) or new ways of exploiting existing ones
  - use the *Knowledge Graph* and similiar resources to improve parsing and information extraction
- Develop models in which natural language is just one of the kinds of information used
  - integrate language and vision
  - integrate language with external database (e.g., financial data, health records)

Deeper questions facing the field

- Our scientific understanding of semantics (meanings), world knowledge and real-world inference is still very poor
  - can existing methods scale up, or will we need a radical breakthrough?
- NLP (like most of ML) reduces *learning to optimisation*
  - we have good methods for estimating weights for features
  - but identifying possibly-useful features is a "black art"
- *Hierarchical non-parametric Bayesian methods* offer a mathematical framework for learning the relevant features as well as their weights
  - the *base distribution* generates (a possibly infinite number of) potentially useful elements
  - from which a finite subset are actually instantiated, based on the data
  - the algorithms simultaneously search for useful elements and learn their weights

MACQUARIE UNIVERSITY

Interested in Machine Learning and Computational Linguistics?

We're recruiting bright *PhD students*.

Contact *Mark.Johnson@MQ.edu.au* for more information.

MACQUARIE UNIVERSITY