

Text Preprocessing

COMP90042
Natural Language Processing
Lecture 2



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L2

Definitions

- **Corpus:** a collection of documents.
- **Document:** one or more sentences.
- Sentence
 - ▶ "The student is enrolled at the University of Melbourne."
- Words
 - ▶ Sequence of characters with a meaning and/or function
- **Word token:** each instance of "the" in the sentence above.
 - E.g. 9 word tokens in the example sentence.
- **Word type:** the distinct word "the".
 - ▶ Lexicon ("dictionary"): a group of word types.
 - ▶ E.g. 8 word types in the example sentence.

2

COMP90042

L2

How Many Unique Words?

	#Tokens (N)	#Type (V)
Switchboard phone conversation	2.4 million	20 thousand
Shakespeare	800 thousand	31 thousand
Google N-gram	1 trillion	13 million

Church and Gale (1990): $V \gg O(N^{1/2})$

3

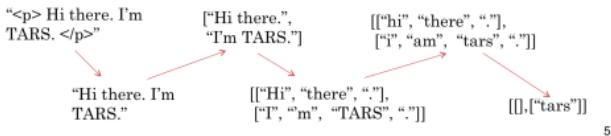
Why Preprocess?

- Most NLP applications have documents as inputs:
 - “This movie is so great!!! U should definitely watch it in the theater! Best sci-fi eva!” → 😊 *Sentiment analysis*
 - “Eu estive em Melbourne no ano passado.” → “I was in Melbourne last year.” *natural translation*
- Key point:** language is **compositional**. As humans, we can **break** these documents into individual **components**. To understand language, a computer should do the same.
- Preprocessing** is the first step.

4

Preprocessing Steps

- Remove unwanted **formatting** (e.g. HTML)
- Sentence segmentation:** break documents into **sentences**
- Word tokenisation:** break sentences into **words**
- Word normalisation:** transform words into **canonical forms**
- Stopword removal:** delete unwanted words



5

Sentence Segmentation

6

Sentence Segmentation

- Naïve approach:** break on sentence punctuation (., ?!)
- But periods are used for **abbreviations!** (U.S. dollar, ..., Yahoo! as a word)
- Second try: **use regex to require capital** ([.?!] [A-Z])
- But abbreviations often **followed by names** (Mr. Brown)
- Better yet: **have lexicons**:
 - But difficult to enumerate all names and abbreviations
- State-of-the-art **uses machine learning, not rules**

rule-based

7

Binary Classifier

- Looks at every “.” and decides whether it is the end of a sentence.

► Decision trees, logistic regression

- Features

► Look at the words before and after “.”

► Word shapes:

- Uppercase, lowercase, ALL_CAPS, number
- Character length

► Part-of-speech tags: (word classes)

- Determiners tend to start a sentence

rel to finish/start sentences

8

Word Tokenisation

9

Word Tokenisation: English

- Naïve approach: separate out alphabetic strings (w+)
- Abbreviations (U.S.A.)
- Hyphens (merry-go-round vs. well-respected vs. yes-but)
- Numbers (1,000,00.01)
- Dates (3/1/2016)
- Clitics (n't in can't)
- Internet language (<http://www.google.com>, #metoo, :-))
- Multiword units (New Zealand)

10

Word Tokenisation: Chinese

- Some Asian languages are written without spaces between words
- In Chinese, words often correspond to more than one character

墨大 的 学生 与众不同

Unimelb 's students (are) special

11

Word Tokenisation: Chinese

- Standard approach assumes an existing vocabulary
 - MaxMatch algorithm
 - Greedily match longest word in the vocabulary

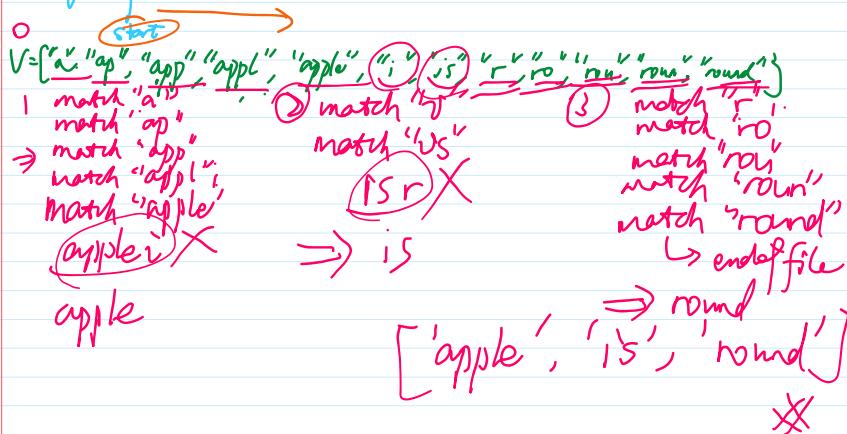
$V = \{\text{墨大, 的, 学生, 与, 众, 不, 同, 墨大, 学生, 不同, 与众不同}\}$

墨大的学生与众不同

match 墨大, match 的, match 学生, match 与众不同,
move to 的 move to 学 move to 与 done

maxmatch alg.

string → apple is round



Word Tokenisation: Chinese

- But how do we know what the vocabulary is
 - And doesn't always work

去 买 新西兰 花
go buy New Zealand flowers

去 买 新 西兰花
go buy new broccoli

Word Tokenisation: German

- Lebensversicherungsgesellschaftsangestellter
- = life insurance company employee
- Requires compound splitter

break long words into individual components



Subword Tokenisation

part-of words

- Colourless green ideas sleep furiously → [colour] [less] [green] [idea] [s] [sleep] [furious] [ly]
 - One popular algorithm: byte-pair encoding (BPE)
 - Core idea: iteratively merge frequent pairs of characters
 - Advantage:
 - Data-informed tokenisation
 - Works for different languages
 - Deals better with unknown words
- don't need to know anything about language
Only need a large corpus
doesn't care about scripts of language
at most break into individual characters
No unknown words problem*

Byte-Pair Encoding

- Dictionary
 - [5] l o w _
 - [2] l o w e s t _
 - [6] n e w e r _
 - [3] w i d e r _
 - [2] n e w _
- Vocabulary
 - _, d, e, i, l, n, o, r, s, t, w

Identify which character pairs occur most and merge them

Byte-Pair Encoding

- Dictionary
 - [5] l o w _
 - [2] l o w e s t _
 - [6] n e w e r _
 - [3] w i d e r _
 - [2] n e w _
- Vocabulary
 - _, d, e, i, l, n, o, r, s, t, w, r_

8 9

Byte-Pair Encoding

- Dictionary

- ▶ [5] l o w _
- ▶ [2] l o w e s t _
- ▶ [6] n e w **er**
- ▶ [3] w i d **er**_
- ▶ [2] n e w _

B > 9

- Vocabulary

- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_

18

Byte-Pair Encoding

- Dictionary

- ▶ [5] l o w _
- ▶ [2] l o w e s t _
- ▶ [6] n **ew** **er**_
- ▶ [3] w i d **er**_
- ▶ [2] n **ew** _

B > 8

- Vocabulary

- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew

19

Byte-Pair Encoding

- Vocabulary

- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew
- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, **new**
- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, **lo**
- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, **low**
- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, **newer**
- ▶ _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, newer, **low**

20

Byte-Pair Encoding

- In practice BPE will run with thousands of merges, creating a large vocabulary
- Most frequent words will be represented as full words
- Rarer words will be broken into subwords
- In the worst case, unknown words in test data will be broken into individual letter

21

Word Normalisation

transform a word into canonical form

Word Normalisation

- Lower casing (Australia → australia)
- Removing morphology $\text{[pl.} \rightarrow \text{sgl.]}$
- Correcting spelling
- Expanding abbreviations (U.S.A → USA)
- Goal:
 - Reduce vocabulary
 - Maps words into the same type

Same meaning

morphology

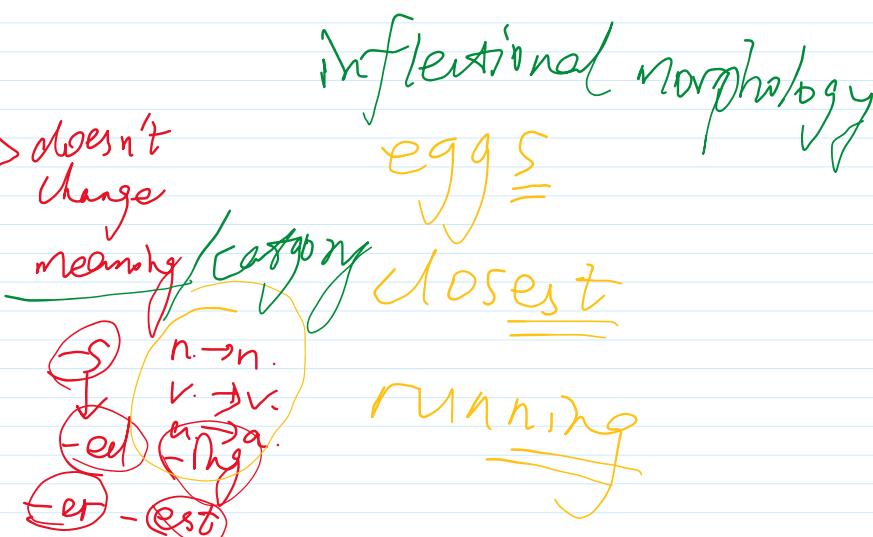
→ { inflectional ~
derivational ~

Inflectional Morphology

- Inflectional morphology creates grammatical variants
- English inflects nouns, verbs, and adjectives
 - Nouns: number of the noun (-s)
 - Verbs: number of the subject (-s), the aspect (-ing) of the action and the tense (-ed) of the action
 - Adjectives: comparatives (-er) and superlatives (-est)
- Many languages have much richer inflectional morphology than English
 - E.g. French inflects nouns for gender (*un chat, une chatte*)

Lemmatisation

- Lemmatisation means removing any inflection to reach the uninflected form, the lemma
 - speaking → speak



a meaningful word /, /

- Lemmatisation means removing any inflection to reach the uninflected form, the **lemma**
 - speaking → speak
- In English, there are irregularities that prevent a trivial solution:
 - poked → poke (not pok)
 - stopping → stop (not stopp)
 - watches → watch (not watche)
 - was → be (not wa)
- A lexicon of **lemmas** needed for accurate lemmatisation

25

a meaningful word / token

COMP90042

Derivational Morphology

L2

- Derivational morphology creates distinct words**
- English derivational **suffixes** often change the lexical category, e.g.
 - ly (personal → personally)
 - ise (final → finalise)
 - er (write → writer)
- English derivational **prefixes** often change the meaning without changing the lexical category
 - write → rewrite
 - healthy → unhealthy

26

change category
v. n. adj. adv.

teacher → teach

(n) → (v)

beautifully → beautiful

(adv) → (adj)

COMP90042

Stemming

L2

- Stemming strips off all suffixes, leaving a **stem****
 - E.g. automate, automatic, automation → automat
 - Often not an actual lexical item
- Even less lexical sparsity than lemmatisation
- Popular in information retrieval
- Stem not always interpretable

27

COMP90042

stem algorithm

L2

The Porter Stemmer

- Most popular stemmer for English
- Applies rewrite rules in stages
 - First strip inflectional suffixes,
 - E.g. -ies → -i
 - Then derivational suffixes
 - E.g. -isation → -ise → -i

28

The Porter Stemmer

- c (lowercase) = consonant; e.g. 'b', 'c', 'd'
- v (lowercase) = vowel; e.g. 'a', 'e', 'i', 'o', 'u'
- C = a sequence of consonants
 - ▶ s, ss, tr, bl
- V = a sequence of vowels
 - ▶ o, oo, ee, io

29

The Porter Stemmer

- A word has one of the four forms:
 - ▶ CVCV ... C
 - ▶ CVCV ... V
 - ▶ VCVC ... C
 - ▶ VCVC ... V
- Which can be represented as:
 - ▶ [C]VCVC ... [V]
 - ▶ [C] (VC)^m [V]
 - ▶ m = measure

30

The Porter Stemmer

- m=0: TR, EE, TREE, Y, BY
- m=1: TROUBLE, OATS, TREES, IVY
- m=2: TROUBLES, PRIVATE, OATEN, ORRERY
- TREE = C(VC)⁰V
- TREES = C(VC)¹
- TROUBLE = C(VC)²

31

The Porter Stemmer

- Rules format: (condition) S₁ → S₂
- e.g. (m > 1) EMENT → null
 - ▶ REPLACEMENT → REPLAC
- Always use the longest matching S₁
 - ▶ CARESSES → CARESS
 - ▶ CARESS → CARESS
 - ▶ CARES → CARE

Rules:

SSES	→ SS
IES	→ I
SS	→ SS
S	→ null

32

$[C][VC]^m[V]$
 (m>1) enent → null
 regularer → replace
 ester → CVCVC
 \Downarrow
 $[C][VC]^m$
 ement → null
 replace ✗

COMP90042

L2

The Porter Stemmer

- Step 1: plurals and past participles *inflectional morphology*

	Rule	Positive Example	Negative Example
a	SSES → SS	caresses → caress	
	IES → I	ponies → poni	
	SS → SS	caress → caress	
	S → null	cats → cat	
b	(m>0) EED → EE	agreed → agree	feed → feed
	(*v*) ED → null *v* = stem has vowel	plastered → plaster	bled → bled
	(*v*) ING →	motoring → motor	sing → sing
b+	AT → ATE	confat(ed) → confiate	
c	(*v*) Y → I	happy → happi	

33

COMP90042

L2

The Porter Stemmer

- Step 2, 3, 4: derivational inflections

	Rule	Positive Example
2	(m>0) ATIONAL → ATE	relational → relate
	(m>0) TIONAL → TION	conditional → condition
	(m>0) ENCI → ENCE	valenci → valence
	(m>0) ANCI → ANCE	hesitanci → hesitance
3	(m>0) ICATE → IC	triplicate → triplic
	(m>0) ATIVE → null	formative → form
	(m>0) ALIZE → AL	formalize → formal
4	(m>1) AL → null	revival → reviv
	(m>1) ER → null	airliner → airlin
	(m>1) ATE → null	activate → activ

34

COMP90042

L2

The Porter Stemmer

- Step 5: tidying up

normalise

	Rule	Positive Example
5	(m>1) E → null	probate → probat
	(m=1 and not *o) E → null *o = stem ends cvc, and second c is not w, x or y (e.g. -WIL, -HOP)	cease → ceas
	(m>1 and *d and *L) null → single letter	
	*d = stem ends with double consonant (e.g. -TT) *L = stem ends with 'l'	controll → control

35

The Porter Stemmer

- computational → comput
 - ▶ step 2: ATIONAL → ATE: computate
 - ▶ step 4: ATE → null: comput
- computer → comput
 - ▶ step 4: ER → null: comput

36

Fixing Spelling Errors

- Why fix them?
 - ▶ Spelling errors create new, rare types
 - ▶ Disrupt various kinds of linguistic analysis
 - ▶ Very common in internet corpora
 - ▶ In web search, particularly important in queries
- How?
 - ▶ String distance (Levenshtein, etc.)
 - ▶ Modelling of error types (phonetic, typing etc.)
 - ▶ Use an n -gram language model

37

Other Word Normalisation

- Normalising spelling variations
 - ▶ Normalize → Normalise (or vice versa)
 - ▶ U r so coool! → you are so cool
- Expanding abbreviations
 - ▶ US, U.S. → United States
 - ▶ imho → in my humble opinion

38

Stopword Removal

39

Stop Words

- Definition: a list of words to be removed from the document
 - Typical in bag-of-word (BOW) representations
 - Not appropriate when sequence is important
- How to choose them? *order of words*
 - All closed-class or function words
 - E.g. *the, a, of, for, he, ...*
 - Any high frequency words
 - NLTK, spaCy NLP toolkits

40

A Final Word

- Preprocessing unavoidable in text analysis
- Can have a major effect on downstream applications
- Exact steps may vary depending on corpus, task
- Simple rule-based systems work well, but rarely perfectly
- Language-dependent

41

Further Reading

- J&M3 Ch 2. on Normalisation
(includes a review of regex and Levenshtien distance)
- Details on the Porter Stemmer algorithm <http://snowball.tartarus.org/algorithms/porter/stemmer.html>

42



13-ngram

N-gram Language Models

COMP90042
Natural Language Processing
Lecture 3



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L3

Language Models

- NLP is about *explaining language*
 - ▶ Why some sentences are more **fluent** than others
- E.g. in speech recognition:
 - *recognise speech > wreck a nice beach*
- We measure ‘goodness’ using **probabilities** estimated by language models
- Language model can also be used for **generation**

2

Talk to Transformer

See how a modern neural network completes your text. Type a custom snippet or try one of the examples. [Learn more](#) below.

[Follow @AdamDanielKing](#) for more neat neural networks.

Custom prompt

Today we have a lecture on Natural Language Processing.

[GENERATE ANOTHER](#)

Completion

Today we have a lecture on Natural Language Processing. It's a very unusual subject, but a very interesting subject, and this is the first time that it's been covered in a Livefist podcast. And, of course, we're going to talk about what makes Natural Language Processing a little different from other language processing areas: namely its social nature. We're going to talk about the question of artificial intelligence, which is something that we've spoken about here and a lot of people on the Livefist messageboard talk about a lot. What is the nature of AI? Is it an essentially self-regulating system, an ultimate mind, the best machine of its kind? What is the nature of intelligence, and how does it work?

3

Language Models

- Useful for
 - ▶ **Speech recognition**
 - ▶ **Spelling correction**
 - ▶ **Query completion**
 - ▶ **Optical character recog.**
- Other generation tasks
 - ▶ **Machine translation**
 - ▶ **Summarisation**
 - ▶ **Dialogue systems**

- 🔍 lots of
- 🔍 lots of **fins**
- 🔍 lots of **love**
- 🔍 lots of **discharge**
- 🔍 lots of **synonym**

4

Outline

- Deriving n -gram language models
- Smoothing to deal with sparsity
- Generating Language
- Evaluating language models

5

Probabilities: Joint to Conditional

Our goal is to get a probability for an arbitrary sequence of m words

$$P(w_1, w_2, \dots, w_m)$$

First step is to apply the chain rule to convert joint probabilities to conditional ones

$$P(w_1, w_2, \dots, w_m) = P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2) \dots P(w_m | w_1 \dots w_{m-1})$$

6

The Markov Assumption

Still intractable, so make a simplifying assumption:

$$P(w_i | w_1 \dots w_{i-1}) \approx P(w_i | w_{i-n+1} \dots w_{i-1})$$

For some small n

When $n = 1$, a unigram model

$$P(w_1, w_2, \dots w_m) = \prod_{i=1}^m P(w_i)$$

Only associated with
n words before it
(w_i)

When $n = 2$, a bigram model

$$P(w_1, w_2, \dots w_m) = \prod_{i=1}^m P(w_i | w_{i-1})$$

When $n = 3$, a trigram model

$$P(w_1, w_2, \dots w_m) = \prod_{i=1}^m P(w_i | w_{i-2} w_{i-1})$$

7

Maximum Likelihood Estimation

How do we calculate the probabilities? Estimate **based on counts** in our corpus:

For unigram models,

$$P(w_i) = \frac{C(w_i)}{M}$$

For bigram models,

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

For n-gram models generally,

$$P(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

8

Book-ending Sequences

- Special tags used to denote start and end of sequence
 - $<\text{s}>$ = sentence start (□ in E18)
 - $</\text{s}>$ = sentence end (■ in E18)

9

Trigram example

Corpus:

$<\text{s}> <\text{s}> \text{yes no no no no yes } </\text{s}>$
 $<\text{s}> <\text{s}> \text{no no no yes yes yes no } </\text{s}>$

What is the probability of

$<\text{s}> <\text{s}> \text{yes no no yes } </\text{s}>$

under a trigram language model?

$(\text{s}) (\text{s}) \text{ yes}$
 $\text{no} \text{ no} \text{ no}$
 $\text{no} \text{ yes} \text{ yes}$
 $\text{no} \text{ yes} \text{ yes}$
 $\text{no} \text{ yes} \text{ yes}$

$P(\text{sent} = \text{yes no no yes})$

$$= P(\text{yes} | <\text{s}> <\text{s}>) * P(\text{no} | <\text{s}> \text{yes}) * P(\text{no} | \text{yes no}) \\ * P(\text{yes} | \text{no no}) * P(</\text{s}> | \text{no yes})$$

$$= \frac{1}{2} * 1 * \frac{1}{2} * \frac{2}{5} * \frac{1}{2} = 0.1$$

5 "no no" bigrams

10

Several Problems

- Language has long distance effects — need large n
 - ▶ The lecture/s that took place last week was/were on preprocessing.
- Resulting probabilities are often very small
 - ▶ Use log probability to avoid numerical underflow
- No probabilities for unseen words
 - ▶ Special symbol to represent them (e.g. <UNK>)
- Words in new contexts
 - ▶ By default, zero count for any n -gram we've never seen before, zero probability for the sentence
 - ▶ Need to smooth the LM!

11

Smoothing

12

Smoothing

- Basic idea: give events you've never seen before some probability

- Must be the case that $P(\text{everything}) = 1$

- Many different kinds of smoothing

- Laplacian (add-one) smoothing
- Add-k smoothing
- Jelinek-Mercer interpolation
- Katz backoff
- Absolute discounting
- Kneser-Ney
- And others...

Sparsity

13

Laplacian (Add-one) Smoothing

- Simple idea: pretend we've seen each n -gram once more than we did.

For unigram models (V = the vocabulary),

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |V|}$$

For bigram models,

$$P_{add1}(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i) + 1}{C(w_{i-1}) + |V|}$$

14

Add-one Example

< s > the rat ate the cheese < /s >

~~1 + 1
1 + 5~~

What's the bigram probability $P(\text{ate}|\text{rat})$ under add-one smoothing?

$$= \frac{C(\text{rat ate}) + 1}{C(\text{rat}) + |V|} = \frac{2}{6}$$

$V = \{ \text{the}, \text{rat}, \text{ate}, \text{cheese}, \langle \text{s} \rangle \}$
Never predict < /s >

What's the bigram probability $P(\text{ate}|\text{cheese})$ under add-one smoothing?

$$= \frac{C(\text{cheese ate}) + 1}{C(\text{cheese}) + |V|} = \frac{1}{6}$$

15

Add-k Smoothing

- Adding one is often too much
- Instead, add a fraction k
- AKA Lidstone Smoothing

take away too much effective counts
from observed n-grams
make rare events too frequent

$$P_{addk}(w_i | w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + k}{C(w_{i-2}, w_{i-1}) + k |V|}$$

- Have to choose k  hyperparameter, we need to tune
 - ▶ number of "classes" is huge (n-grams), so can have a big effect

16

Lidstone Smoothing

Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

Lidstone Smoothing

Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$	
			effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391
<i>offense</i>	5	0.25	4.928	0.246
<i>damage</i>	4	0.2	3.961	0.198
<i>deficiencies</i>	2	0.1	2.029	0.101
<i>outbreak</i>	1	0.05	1.063	0.053
<i>infirmitiy</i>	0	0	0.097	0.005
<i>cephalopods</i>	0	0	0.097	0.005
	20	1.0	20	1.0

$(8 + 0.1) / (20 + 7 \times 0.1)$

17

Lidstone Smoothing

Context = *alleged*

- 5 observed n-grams
- 2 unobserved n-grams

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$	
			effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391
<i>offense</i>	5	0.25	4.928	0.246
<i>damage</i>	4	0.2	3.961	0.198
<i>deficiencies</i>	2	0.1	2.029	0.101
<i>outbreak</i>	1	0.05	1.063	0.053
<i>infirmitiy</i>	0	0	0.097	0.005
<i>cephalopods</i>	0	0	0.097	0.005
	20	1.0	20	1.0

$(8 + 0.1) / (20 + 7 \times 0.1)$

$(0 + 0.1) / (20 + 7 \times 0.1)$

18

Lidstone Smoothing

Context = *alleged*

- 5 observed n-grams
- 2 unobserved n-grams

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$	
			effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391
<i>offense</i>	5	0.25	4.928	0.246
<i>damage</i>	4	0.2	3.961	0.198
<i>deficiencies</i>	2	0.1	2.029	0.101
<i>outbreak</i>	1	0.05	1.063	0.053
<i>infirmity</i>	0	0	0.097	0.005
<i>cephalopods</i>	0	0	0.097	0.005
	20	1.0	20	1.0
		0.391×20		
			$(8 + 0.1) / (20 + 7 \times 0.1)$	19
			$(0 + 0.1) / (20 + 7 \times 0.1)$	

Absolute Discounting

- ‘Borrows’ a **fixed** probability mass from observed n-gram counts
- Redistributions it to unseen n-grams

Absolute Discounting

Context = *alleged*

- 5 observed n-grams
- 2 unobserved n-grams

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$	Discounting, $d = 0.1$		
			effective counts	smoothed probability	effective counts	smoothed probability
<i>impropriety</i>	8	0.4	7.826	0.391	7.9	0.395
<i>offense</i>	5	0.25	4.928	0.246	4.9	0.245
<i>damage</i>	4	0.2	3.961	0.198	3.9	0.195
<i>deficiencies</i>	2	0.1	2.029	0.101	1.9	0.095
<i>outbreak</i>	1	0.05	1.063	0.053	0.9	0.045
<i>infirmity</i>	0	0	0.097	0.005	0.25	0.013
<i>cephalopods</i>	0	0	0.097	0.005	0.25	0.013

20

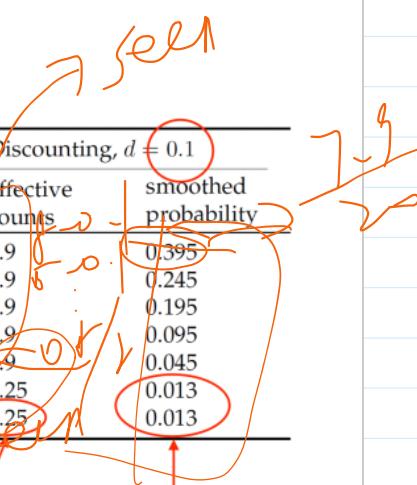
$$8 - 0.1$$

$$(0.1 \times 5) / 2$$

$$\text{total amount of discounted probability mass}$$

$$(0.1 \times 5) / 20$$

21



Backoff

- Absolute discounting redistributes the probability mass **equally** for all unseen n-grams
- Katz Backoff: redistributes the mass based on a lower order model (e.g. unigram)

$$P_{\text{katz}}(w_i | w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j: C(w_{i-1}, w_j) = 0} P(w_j)}, & \text{otherwise} \end{cases}$$

↑
sum unigram probabilities for all words that do not co-occur with context w_{i-1}

↑
unigram probability for w_i

the amount of probability mass that has been discounted for context w_{i-1}
 $((0.1 \times 5) / 20$ in previous slide)

22

Issues with Katz Backoff

$$P_{katz}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j: C(w_{i-1}, w_j) = 0} P(w_j)}, & \text{otherwise} \end{cases}$$

- *I can't see without my reading* _____
- $C(\text{reading}, \text{glasses}) = C(\text{reading}, \text{Francisco}) = 0$
- $C(\text{Francisco}) > C(\text{glasses})$
- Katz backoff will give higher probability to
Francisco

23

Kneser-Ney Smoothing

- Redistribute probability mass based on how many number of different contexts word w has appeared in
- This measure is called “continuation probability”

24

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \underline{P_{cont}(w_i)}, & \text{otherwise} \end{cases}$$

where

$$P_{cont}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}$$

unique context of words
 has occurred before w_i
 sum of all possible w_i

- High continuation counts for *glasses* (*men's glasses*, *black glasses*, *buy glasses*, *prescription glasses*, etc)
- Low continuation counts for *Franciso* (*San Francisco*)

25

Interpolation

- A better way to combine different order n -gram models
- Weighted sum of probabilities across progressively shorter contexts
- Interpolated trigram model:

$$p_{\text{Interpolation}}(w_m | w_{m-1}, w_{m-2}) = \lambda_3 p_3^*(w_m | w_{m-1}, w_{m-2}) + \lambda_2 p_2^*(w_m | w_{m-1}) + \lambda_1 p_1^*(w_m).$$

Learned based on held out data

trigram
 bigram
 unigram

$\sum_{n=1}^{n_{\max}} \lambda_n = 1$

26

Interpolated Kneser-Ney Smoothing

- Interpolation instead of back-off

$$P_{IKN}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) - D}{C(w_{i-1})} + \beta(w_{i-1}) P_{cont}(w_i)$$

- where $\beta(w_{i-1}) =$

▶ normalising constant so that $P_{IKN}(w_i|w_{i-1})$
sums to 1.0

27

In Practice

- Commonly used Kneser-Ney language models use 5-grams as max order
- Has different discount values for each n-gram order.

based on data

28

Generating Language

29

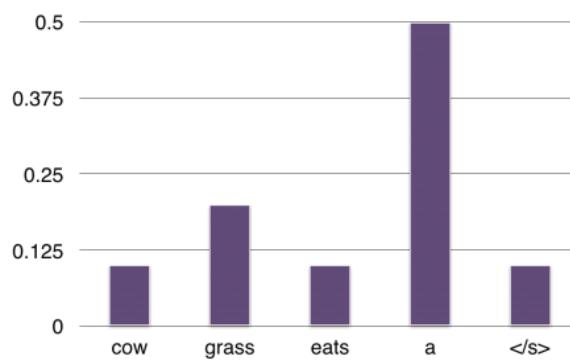
Generation

- Given an initial word, draw the next word according to the probability distribution defined by the language model.
- Include (n-1) start tokens for n-gram model, to provide context to generate first word
 - never generate <s>
 - generating </s> terminates the sequence

30

Generation (Bigram LM)

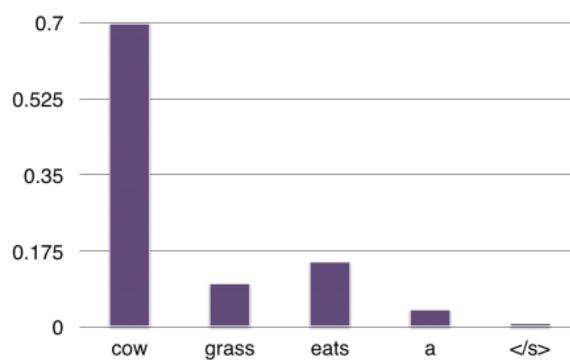
- Sentence = <s>
- $P(\cdot | \langle s \rangle) = "a"$



31

Generation (Bigram LM)

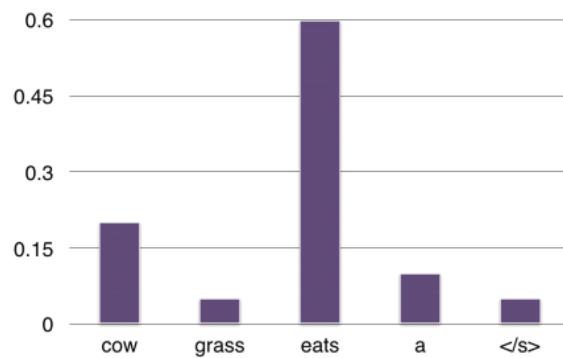
- Sentence = <s> a
- $P(\cdot | a) = "cow"$



32

Generation (Bigram LM)

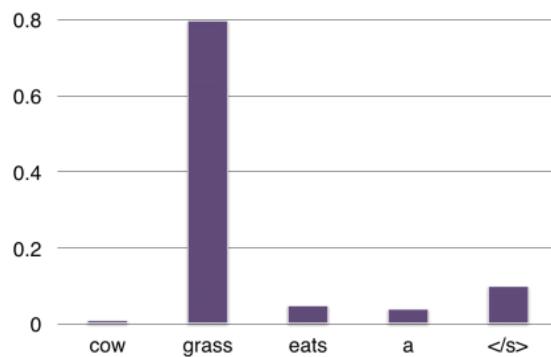
- Sentence = <s> a cow
- $P(\text{?} | \text{cow}) = \text{"eats"}$



33

Generation (Bigram LM)

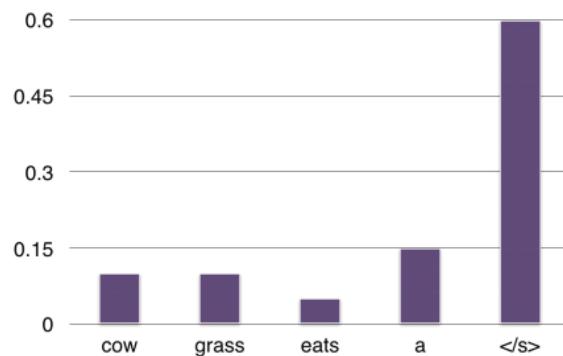
- Sentence = <s> a cow eats
- $P(\text{?} | \text{eats}) = \text{"grass"}$



34

Generation (Bigram LM)

- Sentence = <s> a cow eats grass
- $P(\text{?} | \text{grass}) = \langle\text{/s}\rangle$



35

Generation (Bigram LM)

- Sentence = <s> a cow eats grass </s>
- Done!

36

How to Select Next Word?

- Argmax: takes highest probability word each turn
 - ▶ Greedy search
- Beam search decoding:
 - ▶ Keeps track of top-N highest probability words each turn
 - ▶ Produces sentences with near-optimal sentence probability
- Randomly samples from the distribution (e.g. temperature sampling)

37

Evaluating Language Models

38

Evaluation

- Extrinsic → look at downstream tasks' performance
 - E.g. spelling correction, machine translation
- Intrinsic
 - Perplexity on held-out test set

39

Perplexity

- Inverse probability of entire test set
 - Normalized by number of word tokens (including </s>)
- The lower the better

$$PP(w_1, w_2, \dots, w_m) = \sqrt[m]{\frac{1}{P(w_1, w_2, \dots, w_m)}}$$

equivalently

$$\underline{PP(w_1, w_2, \dots, w_m)} = 2^{-\frac{\log_2 P(w_1, w_2, \dots, w_m)}{m}}$$

- Unknown (OOV) words a problem (omit)

40

Example Perplexity Scores

	Unigram	Bigram	Trigram
Perplexity	962	170	109

- Corpus: Wall Street Journal
- Train partition: 38 million word tokens, almost 20K word types (vocabulary)
- Test partition: 1.5 million word tokens

41

A Final Word

- N -gram language models are a simple but effective way to capture the predictability of language
- Information can be derived in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective, due to sparsity

42

Reading

- E18 Chapter 6 (skip 6.3)



Text Classification

COMP90042

Natural Language Processing

Lecture 4



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L4

Outline

- Fundamentals of classification
- Text classification tasks
- Algorithms for classification
- Evaluation

2

Classification

- Input
 - A document d
 - Often represented as a vector of *features*
 - A fixed output set of classes $C = \{c_1, c_2, \dots, c_k\}$
 - Categorical, not continuous (regression) or ordinal (ranking)
- Output
 - A predicted class $c \in C$
 - *close set*

3

Text Classification Tasks

- Obviously more than can be enumerated here
- But let's briefly look at some major examples:
 - Topic classification
 - Sentiment analysis
 - Authorship attribution
 - Native-language identification
 - Automatic fact-checking
- Not necessarily a full-sized “text”
 - E.g. sentence or tweet-level polarity classification

4

Topic Classification

underline topics of given document

- Motivation: library science, information retrieval
- Classes: Topic categories, e.g. “jobs”, “international news”
- Features
 - Unigram bag of words (BOW), with stop-words removed
 - Longer n-grams (bigrams, trigrams) for phrases
- Examples of corpora
 - Reuters news corpus (RCV1, see NLTK sample)
 - Pubmed abstracts
 - Tweets with hashtags

5

Topic Classification Example

Is the topic of this text from the Reuters news corpus
acquisitions or earnings?

LIEBERT CORP APPROVES MERGER

Liebert Corp said its shareholders approved the merger of a wholly-owned subsidiary of Emerson Electric Co. Under the terms of the merger, each Liebert shareholder will receive .3322 shares of Emerson stock for each Liebert share.

ANSWER: ACQUISITIONS

6

Sentiment Analysis

- Motivation: opinion mining, business analytics
- Classes: Positive/Negative/(Neutral)
- Features
 - N-grams
 - Polarity lexicons
- Examples of corpora
 - Polarity movie review dataset (in NLTK)
 - SEMEVAL Twitter polarity datasets

7

Sentiment Analysis Example

What is the **polarity (+/-)** of this tweet from the SEMEVAL dataset?

anyone having problems with Windows 10? may be coincidental but since i downloaded, my WiFi keeps dropping out. Itunes had a malfunction

ANSWER: NEGATIVE

8

Authorship Attribution

- Motivation: forensic linguistics, plagiarism detection
- Classes: Authors (e.g. Shakespeare)
- Features
 - Frequency of function words
 - Character n -grams
 - Discourse structure
- Examples of corpora
 - Project Gutenberg corpus (see NLTK sample)

9

Author Attribution Example

Which famous **novelist** wrote this text from Project Gutenberg?

Mr. Dashwood's disappointment was, at first, severe; but his temper was cheerful and sanguine; and he might reasonably hope to live many years, and by living economically, lay by a considerable sum from the produce of an estate already large, and capable of almost immediate improvement. But the fortune, which had been so tardy in coming, was his only one twelvemonth. He survived his uncle no longer; and ten thousand pounds, including the late legacies, was all that remained for his widow and daughters.

ANSWER: JANE AUSTEN

10

Native-Language Identification

- Motivation: forensic linguistics, educational applications
 - Classes: first language of author (e.g. Chinese)
 - Features
 - Word N-grams
 - Syntactic patterns (POS, parse trees)
 - Phonological features
 - Examples of corpora
 - TOEFL/IELTS essay corpora
- how arrange words
in native language

11

Native-Language Identification

What is the **native language** of the writer of this text?

Now a festival of my university is being held, and my club is joining it by offering a target practice game using bows and arrows of archery. I'm a manager of the attraction, so I have worked to make it succeed. I found it puzzled to manage a event or a program efficiently without generating a free rider. The event is not free, so we earn a lot of money.

ANSWER: JAPANESE

12

Automatic Fact-checking

- Motivation: social media, journalism (fake news)
- Classes: True/False/(Can't be sure)
- Features
 - N-grams
 - Non-text metadata *beyond textual info*
- Examples of corpora
 - Emergent, LIAR: political statements
 - FEVER

13

Automatic Fact-checking

Is this statement **true** or **false**?

Austin is burdened by the fastest-growing tax increases of any major city in the nation.

ANSWER:FALSE

14

Building a Text Classifier

1. Identify a task of interest
2. Collect an appropriate corpus
3. Carry out annotation → label documents
4. Select features
5. Choose a machine learning algorithm
6. Tune hyperparameters using held-out development data
7. Repeat earlier steps as needed
8. Train final model
9. Evaluate model on held-out test data

15

Algorithms for Classification

16

Choosing a Classification Algorithm

- Bias vs. Variance
 - Bias: assumptions we made in our model
 - Variance: sensitivity to training set
- Underlying assumptions, e.g., independence
- Complexity *high bias . low variance
linear model*
- Speed

17

Naïve Bayes

- Finds the class with the highest likelihood under Bayes law
 - i.e. probability of the class times probability of features given the class
- Naïvely assumes features are independent

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$p(c_n | f_1 \dots f_m) = \prod_{i=1}^m p(f_i | c_n) p(c_n)$$

18

$$\begin{aligned}
 c_n &= \arg \max_c P(C_i | f_1 \dots f_m) \\
 &= \arg \max_c P(f_1 \dots f_m | C_i) P(C_i) \\
 &\quad P(f_1 \dots f_m) \rightarrow \text{constant} \\
 &= \arg \max_c P(f_1 \dots f_m | C_i) P(C_i) \\
 &\quad \text{independence assumption} \\
 &= \arg \max_c P(C_i) \prod_{j=1}^m P(f_j | C_i) \\
 &= \arg \max_c \prod_{j=1}^m P(f_j | C_i) P(C_i)
 \end{aligned}$$

Naïve Bayes

- Pros: Fast to “train” and classify; robust, low-variance; good for low data situations; optimal classifier if independence assumption is correct; extremely simple to implement.
- Cons: Independence assumption rarely holds; low accuracy compared to similar methods in most situations; smoothing required for unseen class/feature combinations

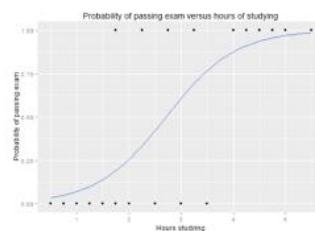
19

Logistic Regression

softmax

- A classifier, despite its name
- A linear model, but uses softmax “squashing” to get valid probability

$$p(c_n | f_1 \dots f_m) = \frac{1}{Z} \cdot \exp\left(\sum_{i=0}^m w_i f_i\right)$$



- Training maximizes probability of training data subject to regularization which encourages low or sparse weights

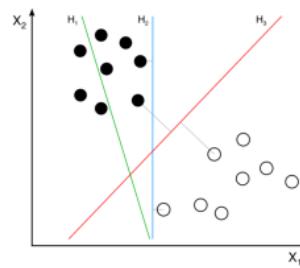
20

Logistic Regression

- Pros: Unlike Naïve Bayes not confounded by diverse, correlated features
- Cons: High bias; slow to train; some feature scaling issues; often needs a lot of data to work well; choosing regularisation a nuisance but important since overfitting is a big problem

21

Support Vector Machines



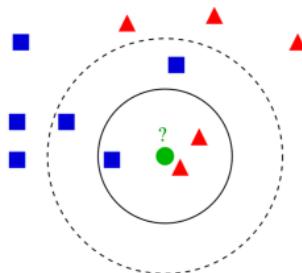
- Finds hyperplane which separates the training data with maximum margin
 - Allows for some misclassification
- Pros: fast and accurate linear classifier; can do non-linearity with kernel trick; works well with huge feature sets
- Cons: Multiclass classification awkward; feature scaling can be tricky; deals poorly with class imbalances; uninterpretable

| vs rest

22

K-Nearest Neighbour

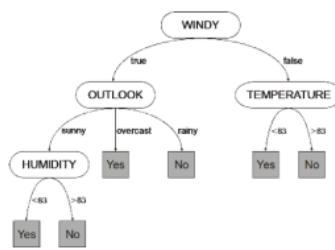
- Classify based on majority class of k -nearest training examples in feature space
- Definition of nearest can vary
 - Euclidean distance
 - Cosine distance
- Pros: Simple, effective; no training required; inherently multiclass; optimal with infinite data
- Cons: Have to select k ; issues with unbalanced classes; often slow (need to find those k -neighbours); features must be selected carefully



23

Decision tree

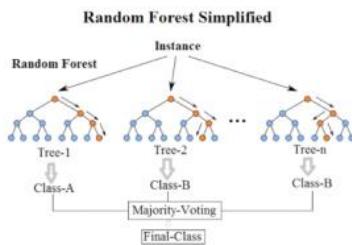
- Construct a tree where nodes correspond to tests on individual features
- Leaves are final class decisions
- Based on greedy maximization of mutual information
- Pros: in theory, very interpretable; fast to build and test; feature representation/scaling irrelevant; good for small feature sets, handles non-linearly-separable problems
- Cons: In practice, often not that interpretable; highly redundant sub-trees; not competitive for large feature sets



24

Random Forests

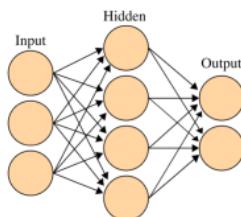
- An *ensemble* classifier
- Consists of decision trees trained on different subsets of the training and feature space
- Final class decision is majority vote of sub-classifiers
- Pros: Usually more accurate and more robust than decision trees, a great classifier for small- to moderate-sized feature sets; training easily parallelised
- Cons: Same negatives as decision trees: too slow with large feature sets



25

Neural Networks

- An interconnected set of nodes typically arranged in layers
- Input layer (features), output layer (class probabilities), and one or more hidden layers
- Each node performs a linear weighting of its inputs from previous layer, passes result through activation function to nodes in next layer
- Pros: Extremely powerful, state-of-the-art accuracy on many tasks in natural language processing and vision
- Cons: Not an off-the-shelf classifier, very difficult to choose good parameters; slow to train; prone to overfitting



26

Hyperparameter Tuning

- Dataset for tuning
 - Development set
 - Not the training set or the test set
 - *k*-fold cross-validation
- Specific hyperparameters are classifier specific
 - E.g. tree depth for decision trees
- But many hyperparameters relate to regularization
 - Regularization hyperparameters penalize model complexity

- But many hyperparameters relate to regularization
 - Regularization hyperparameters penalize model complexity
 - Used to prevent overfitting
- For multiple hyperparameters, use grid search

27

COMP90042

L4

Evaluation

28

COMP90042

L4

Evaluation: Accuracy

Class	Classified As	
	A	B
A	79	13
B	8	10

Accuracy = correct classifications/total classifications

$$= (79 + 10) / (79 + 13 + 8 + 10)$$

$$= 0.81$$

0.81 looks good, but most common class baseline accuracy is

$$= (79 + 13) / (79 + 13 + 8 + 10) = 0.84$$

29

Evaluation: Precision & Recall

Class	Classified As		False Positives (fp)
	A	B	
A	79	13	
B	8	10	True Positives (tp)

False Negatives (fn)

B as "positive class"

$$\begin{aligned} \text{Precision} &= \frac{\text{correct classifications of B (tp)}}{\text{total classifications as B (tp + fp)}} \\ &= 10/(10 + 13) = 0.43 \end{aligned}$$

$$\begin{aligned} \text{Recall} &= \frac{\text{correct classifications of B (tp)}}{\text{total instances of B (tp + fn)}} \\ &= 10/(10 + 8) = 0.56 \end{aligned}$$

30

		labelled	
		positive	negative
real	positive	TP	FN
	negative	FP	TN

multi-class classification

macro-avg

micro-avg

accuracy = $\frac{TP + TN}{TP + FN + FP + FN}$

precision = $\frac{TP}{TP + FP}$

recall = $\frac{TP}{TP + FN}$

F1 score = $\frac{2 \times \text{pre} * \text{re}}{\text{pre} + \text{re}}$

Evaluation: F(1)-score

- Harmonic mean of precision and recall
$$F1 = 2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$$
- Like precision and recall, defined relative to a specific positive class
- But can be used as a general multiclass metric
 - Macroaverage: Average F-score across classes
 - Microaverage: Calculate F-score using sum of counts

31

A Final Word

- Lots of algorithms available to try out on your task of interest (see scikit-learn)
- But if good results on a new task are your goal, then well-annotated, plentiful datasets and appropriate features often more important than the specific algorithm used

32

Further Reading

- ▶ E18 Ch 4.1, 4.3-4.4.1
- ▶ E18 Chs 2 & 3: reviews linear and non-linear classification algorithms

33

Part of speech tagging

COMP90042

Natural Language Processing

Lecture 5



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

What is Part-of-Speech (POS)?

- AKA word classes, morphological classes, syntactic categories
- Nouns, verbs, adjective, etc
- POS tells us quite a bit about a word and its neighbours:
 - ▶ nouns are often preceded by determiners
 - ▶ verbs preceded by nouns
 - ▶ content as a noun pronounced as *CONtent*
 - ▶ content as a adjective pronounced as *conTENT*

6

Authorship Attribution Revisited

- Training data:
 - ▶ “The lawyer convinced the jury.” → Sam
 - ▶ “Ruby travelled around Australia.” → Sam
 - ▶ “The hospital was cleaned by the janitor.” → Max
 - ▶ “Lunch was served at 12pm.” → Max
- “The bookstore was opened by the manager.” → ?
- Similar structure (passive voice).
 - ▶ Not captured by simple BOW representations.
- How to ensure a computer knows/learns this?

7

Information Extraction

- Given this:
 - ▶ “Brasilia, the Brazilian capital, was founded in 1960.”
- Obtain this:
 - ▶ capital(Brazil, Brasilia)
 - ▶ founded(Brasilia, 1960)
- Many steps involved but first need to know **nouns** (Brasilia, capital), **adjectives** (Brazilian), **verbs** (founded) and **numbers** (1960).

8

Outline

Parts of speech, tagsets
Automatic tagging

9

POS Open Classes

Open vs closed classes: how readily do POS categories take on new words? Just a few **open** classes:

form new words

- **Nouns**

- Proper (*Australia*) versus common (*wombat*)
- Mass (*rice*) versus count (*bowls*)

- **Verbs**

- Rich inflection (*go/goes-going/gone/went*)
- Auxiliary verbs (*be, have, and do* in English)
- Transitivity (*wait* versus *hit* versus *give*)
 - number of arguments

10

POS Open Classes

- **Adjectives**

- Gradable (*happy*) versus non-gradable (*computational*)

- **Adverbs**

- Manner (*slowly*)
- Locative (*here*)
- Degree (*really*)
- Temporal (*yesterday*)

11

POS Closed Classes (English)

- **Prepositions** (*in, on, with, for, of, over, ...*)
 - ▶ *on the table*
- **Particles**
 - ▶ *brushed himself off*
- **Determiners**
 - ▶ Articles (*a, an, the*)
 - ▶ Demonstratives (*this, that, these, those*)
 - ▶ Quantifiers (*each, every, some, two, ...*)
- **Pronouns**
 - ▶ Personal (*I, me, she, ...*)
 - ▶ Possessive (*my, our, ...*)
 - ▶ Interrogative or Wh (*who, what, ...*)

basic have static
words

12

POS Closed Classes (English)

- **Conjunctions**
 - ▶ Coordinating (*and, or, but*)
 - ▶ Subordinating (*if, although, that, ...*)
- **Modal verbs**
 - ▶ Ability (*can, could*)
 - ▶ Permission (*can, may*)
 - ▶ Possibility (*may, might, could, will*)
 - ▶ Necessity (*must*)
- And some more...
 - ▶ negatives, politeness markers, etc

13

Ambiguity

- Many word types belong to multiple classes
- Compare:
 - Time flies like an arrow*
 - Fruit flies like a banana*

Time	flies	like	an	arrow
noun	verb	preposition	determiner	noun

Fruit	flies	like	a	banana
noun	noun	verb	determiner	noun

14

POS Ambiguity in News Headlines

- British Left Waffles on Falkland Islands
- Juvenile Court to Try Shooting Defendant
- Teachers Strike Idle Kids
- Eye Drops Off Shelf

15

POS Ambiguity in News Headlines

- [British Left] [Waffles] [on] [Falkland Islands]
- Juvenile Court to Try Shooting Defendant
- Teachers Strike Idle Kids
- Eye Drops Off Shelf

16

POS Ambiguity in News Headlines

- [British Left] [Waffles] [on] [Falkland Islands]
- [Juvenile Court] [to] [Try] [Shooting Defendant]
- Teachers Strike Idle Kids
- Eye Drops Off Shelf

17

POS Ambiguity in News Headlines

- [British Left] [Waffles] [on] [Falkland Islands]
- [Juvenile Court] [to] [Try] [Shooting Defendant]
- [Teachers Strike] [Idle Kids]
- Eye Drops Off Shelf

18

POS Ambiguity in News Headlines

- [British Left] [Waffles] [on] [Falkland Islands]
- [Juvenile Court] [to] [Try] [Shooting Defendant]
- [Teachers Strike] [Idle Kids]
- [Eye Drops] [Off Shelf]

19

Tagsets

- A compact representation of POS information
 - ▶ Usually ≤ 4 capitalized characters
 - ▶ Often includes inflectional distinctions
- Major English tagsets
 - ▶ Brown (87 tags)
 - ▶ Penn Treebank (45 tags)
 - ▶ CLAWS/BNC (61 tags)
 - ▶ “Universal” (12 tags)
- At least one tagset for all major languages

20

Major Penn Treebank Tags

NN noun

VB verb

JJ adjective

RB adverb

DT determiner

CD cardinal number

IN preposition

PRP personal pronoun

MD modal

CC coordinating conjunction

RP particle

WH wh-pronoun

TO *to*

21

Penn Treebank Derived Tags

NN: NNS (plural, *wombats*), NNP (proper, *Australia*),
NNPS (proper plural, *Australians*)

VB: VB (infinitive, *eat*), VBP (1st /2nd person present, *eat*),
VBZ (3rd person singular, *eats*), VBD (past tense, *ate*),
VBG (gerund, *eating*), VBN (past participle, *eaten*)

JJ: JJR (comparative, *nicer*), JJS (superlative, *nicest*)

RB: RBR (comparative, *faster*), RBS (superlative,
fastest)

PRP: PRP\$ (possessive, *my*)

WH: WH\$ (possessive, *whose*), WDT(*wh*-determiner,
who), WRB (*wh*-adverb, *where*)

22

Tagged Text Example

The/DT limits/NNS to/TO legal/JJ absurdity/NN
stretched/VBD another/DT notch/NN this/DT week/
NN

when/WRB the/DT Supreme/NNP Court/NNP
refused/VBD to/TO hear/VB an/DT appeal/VB from/
IN a/DT case/NN that/WDT says/VBZ corporate/JJ
defendants/NNS must/MD pay/VB damages/NNS
even/RB after/IN proving/VBG that/IN they/PRP
could/MD not/RB possibly/RB have/VB
caused/VBN the/DT harm/NN ./.

23

Why Automatically POS tag?

- Important for morphological analysis, e.g. lemmatisation
- For some applications, we want to focus on certain POS
 - ▶ E.g. nouns are important for information retrieval, adjectives for sentiment analysis
- Very useful features for certain classification tasks
 - ▶ E.g. genre classification
- POS tags can offer word sense disambiguation
 - ▶ E.g. cross/*NN* cross/*NB* cross/*JJ*
- Can use them to create larger structures (parsing)

24

Automatic Taggers

- Rule-based taggers
- Statistical taggers
 - ▶ Unigram tagger
 - ▶ Classifier-based taggers
 - ▶ Hidden Markov Model (HMM) taggers

25

Rule-based tagging

- Typically starts with a list of possible tags for each word
 - ▶ From a lexical resource, or a corpus
- Often includes other lexical information, e.g. verb *subcategorisation* (its arguments)
- Apply rules to narrow down to a single tag
 - ▶ E.g. If DT comes before word, then eliminate VB
 - ▶ Relies on some unambiguous contexts
- Large systems have 1000s of constraints

26

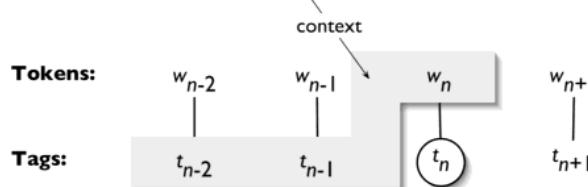
Unigram tagger

- Assign most common tag to each word type
- Requires a corpus of tagged words
- “Model” is just a look-up table
- But actually quite good, ~90% accuracy
 - ▶ Correctly resolves about 75% of ambiguity
- Often considered the baseline for more complex approaches

27

Classifier-Based Tagging

- Use a standard discriminative classifier (e.g. logistic regression, neural network), with features:
 - ▶ Target word
 - ▶ Lexical context around the word
 - ▶ Already classified tags in sentence
- Among the best sequential models
 - ▶ But can suffer from error propagation: wrong predictions from previous steps affect the next ones



28

Hidden Markov Models

- A basic sequential (or structured) model
- Like sequential classifiers, use both previous tag and lexical evidence
- Unlike classifiers, treat previous tag(s) evidence and lexical evidence as independent from each other
 - ▶ Less sparsity
 - ▶ Fast algorithms for sequential prediction, i.e. finding the best tagging of entire word sequence

29

Unknown Words

- Huge problem in morphologically rich languages (e.g. Turkish)
- Can use things we've seen only once (hapax legomena) to best guess for things we've never seen before
 - ▶ Tend to be nouns, followed by verbs
 - ▶ Unlikely to be determiners
- Can use sub-word representations to capture morphology (look for common affixes)

30

A Final Word

- Part of speech is a fundamental intersection between linguistics and automatic text analysis
- A fundamental task in NLP, provides useful information for many other applications
- Methods applied to it are typical of language tasks in general, e.g. probabilistic, sequential machine learning

31

Reading

- JM3 Ch. 8 8.1-8.3, 8.5.1

Lecture 6

Wednesday, April 22, 2020 17:56



l6-hmm

Sequence Tagging: Hidden Markov Models

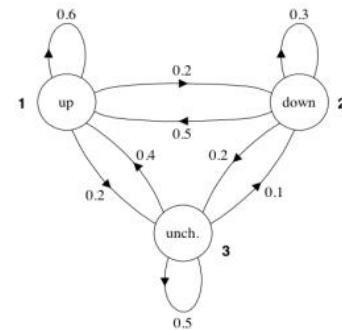
COMP90042

Natural Language Processing

Lecture 6



THE UNIVERSITY OF
MELBOURNE



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

POS Tagging Recap

- Janet will back the bill
- Janet/NNP will/MB back/VP the/DT bill/NN
- Local classifier: prone to error propagation
- What about treating the full sequence as a “class”?
 - ▶ Output: “NNP_MB_VP_DT_NN”
- Problems:
 - ▶ Exponentially many combinations: $|Tags|^M$, for length M
 - ▶ How to tag sequences of different lengths?

2

A Better Approach

- Tagging is a sentence-level task but as humans we decompose it into small word-level tasks.
 - ▶ Janet/NNP will/MB back/VP the/DT bill/NN
- Solution:
 - ▶ Define a model that decomposes process into individual word level steps
 - ▶ But that takes into account the whole sequence when learning and predicting (no error propagation)
- This is the idea of **sequence labelling**, and more general, **structured prediction**.

3

A Probabilistic Model

- Goal: obtain best tag sequence t from sentence w

$$\hat{t} = \operatorname{argmax}_t P(t | w)$$

$$\hat{t} = \operatorname{argmax}_t \frac{P(w | t) P(t)}{P(w)} = \operatorname{argmax}_t P(w | t) P(t)$$

[Bayes]

- Let's decompose:

$$P(w | t) = \prod_{i=1}^n P(w_i | t_i) \quad [\text{Prob. of a word depends only on the tag}]$$

$$P(t) = \prod_{i=1}^n P(t_i | t_{i-1}) \quad [\text{Prob. of a tag depends only on the previous tag}]$$

- These are **independence assumptions** (bigram language models?)
- This is a Hidden Markov Model (HMM)

4

Hidden Markov Model

$$\hat{t} = \operatorname{argmax}_t P(w | t) P(t)$$

$$P(w | t) = \prod_{i=1}^n P(w_i | t_i)$$

$$P(t) = \prod_{i=1}^n P(t_i | t_{i-1})$$

- Why "Markov"?
 - Because it assumes the sequence follows a Markov chain: probability of an event (tag) depends only on the previous event (last tag)
- Why "Hidden"?
 - Because the events (tags) are not seen: goal is to find the best sequence

5

HMMs - Training

- Parameters are the individual probabilities $P(w_i | t_i)$ and $P(t_i | t_{i-1})$
 - Respectively, **emission (O)** and **transition (A)** probabilities
- Training uses Maximum Likelihood Estimation (MLE)**
 - In Naïve Bayes & n-gram LMs, this is done by simply counting word frequencies according to the class.
- We do **exactly the same** in HMMs!

$$P(\text{like} | \text{VB}) = \frac{\text{count(VB, like)}}{\text{count(VB)}}$$

$$P(\text{NN} | \text{DT}) = \frac{\text{count(DT, NN)}}{\text{count(DT)}}$$

6

HMMs - Training

- What about the first tag?
 - Assume we have a symbol “<s>” that represents the start of your sentence.
$$P(\text{NN} | \text{<s>}) = \frac{\text{count(<s>, NN)}}{\text{count(<s>)}}$$
- What about the last tag?
 - Assume we have a symbol “</s>” that represents the end of sentence.
- What about unseen (word,tag) and (tag, previous) combinations?
 - Smoothing techniques, like NB/n-gram LMs

7

condition

Transition Matrix

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

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

8

Emission (Observation) Matrix

	Janet	will	back	the	bill
NNP	0.000022	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

9

HMMs – Prediction (Decoding)

$$\hat{t} = \operatorname{argmax}_t P(w | t) P(t)$$

$$= \operatorname{argmax}_t \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

- Simple idea: for each word, take the tag that maximises $P(w_i | t_i) P(t_i | t_{i-1})$. Do it left-to-right, in greedy fashion.
- This is wrong! We are looking for argmax_t , not individual $\operatorname{argmax}_{t_i}$ terms. *global optimum*
 - This is a local classifier: error propagation
- Correct way: take all possible tag combinations, evaluate them, take the max (like Naïve Bayes)
 - Problem: exponential number of sequences.

10

The Viterbi Algorithm

- Dynamic Programming to the rescue!
 - We can still proceed sequentially, as long as we are careful.
- “can play” -> can/MD play/VB
- Best tag for “can” is easy: $\operatorname{argmax}_t P(\text{can} | t) P(t | \langle s \rangle)$
 - We can do that because first “tag” is always “⟨s⟩”
- Suppose best tag for “can” is NN. To get the tag for “play”, we can take $\operatorname{argmax}_t P(\text{play} | t) P(t | \text{NN})$ but this is wrong.
- Instead, we keep track of scores for each tag for “can” and check what would happen if “can” had a different tag.

11

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP					
MD					
VB					
JJ					
NN					
RB					
DT					

12

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	$P(\text{Janet NNP}) * P(\text{NNP <s>})$				
MD	$P(\text{Janet MD}) * P(\text{MD <s>})$				
VB	...				
JJ	...				
NN	...				
RB	...				
DT	...				

13

Transition and Emission Matrix

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

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

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

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

14

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	0.000032 * 0.2767				
MD	0 * 0.0006				
VB	...				
JJ	...				
NN	...				
RB	...				
DT	...				

15

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •				
MD	0				
VB	0				
JJ	0				
NN	0				
RB	0				
DT	0				

16

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	$P(\text{will} \text{NNP}) * P(\text{NNP} t_{\text{Janet}}) * s(t_{\text{Janet}} t_{\text{Janet}})$			
MD	0	...	Max		
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

17

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	$P(\text{will} \text{NNP}) * P(\text{NNP} \text{Janet}) * s(\text{NNP} \text{Janet})$			
MD	0	...			
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

Calculate this for all tags,
take the max.

$\max(P(\text{will} | \text{NNP}) * P(\text{NNP} | \text{NNP}) * s(\text{NNP} | \text{Janet}),$
 $P(\text{will} | \text{NNP}) * P(\text{NNP} | \text{MD}) * s(\text{MD} | \text{Janet}),$
 \dots
 $P(\text{will} | \text{NNP}) * P(\text{NNP} | \text{DT}) * s(\text{DT} | \text{Janet}))$

18

Transition and Emission Matrix

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

Figure 8.7 The A transition probabilities $P(t_i | t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB | MD)$ is 0.7968.

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

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

19

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	$0 * P(NNP t_{Janet}) * s(t_{Janet} Janet)$			
MD	0	...			
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

20

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0			
MD	0	$P(will MD) * P(MD NNP) * s(NNP Janet)$			
VB	0	...	$\max(P(will MD) * P(MD NNP) * s(NNP Janet),$ $P(will MD) * P(MD MD) * s(MD Janet),$... $P(will MD) * P(MD DT) * s(DT Janet))$		
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

21

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	...			
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

22

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	2.231e-13			
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

23

Transition and Emission Matrix

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

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

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

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

24

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	2.231e-13			
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

25

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	$P(\text{back} \text{VB}) * P(\text{VB} t_{\text{will}}) * s(t_{\text{will}} \text{will})$		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

26

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	MD: 1.6e-11 VB: 7.5e-19 NN: 9.7e-17		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

27

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	MD: 1.6e-11 VB: 7.5e-19 NN: 9.7e-17		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

28

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

29

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0	5.1e-15		
NN	0	1.034e-10	5.4e-15		
RB	0	0	5.3e-11		
DT	0	0	0		

30

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0	2.5e-17	
MD	0	3.004e-8	0	0	
VB	0	2.231e-13	1.6e-11	0	
JJ	0	0	5.1e-15	5.2e-16	
NN	0	1.034e-10	5.4e-15	5.9e-18	
RB	0	0	5.3e-11	0	
DT	0	0	0	1.8e-12	

31

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

(NNP, MD, VB, DT, NN)

32

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

33

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

34

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06 •	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

35

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

36

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

37

The Viterbi Algorithm

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

38

The Viterbi Algorithm

	Janet/NNP	will/MD	back/VB	the/DT	bill/NN
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-08	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	2.0e-15
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

39

The Viterbi Algorithm

- Complexity: $O(T^2N)$, where T is the size of the tagset and N is the length of the sequence.
 - $T * N$ matrix, each cell performs T operations.
- Why does it work?
 - Because of the independence assumptions that decompose the problem (specifically, the Markov property). Without these, we cannot apply DP.

40

Viterbi Pseudocode

```

alpha = np.zeros(M, T)
for t in range(T):
    alpha[1, t] = pi[t] * O[w[1], t]

for i in range(2, M):
    for t_i in range(T):
        for t_last in range(T):
            s = alpha[i-1, t_last] * A[t_last, t_i] * O[w[i], t_i]
            if s > alpha[i, t_i]:
                alpha[i, t_i] = s
                back[i, t_i] = t_last

best = np.max(alpha[M-1,:])
return backtrace(best, back)

```

Invert
Score

- Good practice: work with **log** probabilities to prevent underflow (multiplications become sums)
- Vectorisation (use matrix-vector operations)

41

HMMs In Practice

- We saw HMM taggers based on **bigrams**. State-of-the-art use tag **trigrams**.

$$\rightarrow P(t) = \prod_{i=1}^n P(t_i | t_{i-1}, t_{i-2}) \text{ Viterbi now } O(T^3N)$$

- Need to deal with sparsity: some tag trigram sequences might not be present in training data
 - Backoff: $P(t_i | t_{i-1}, t_{i-2}) = \lambda_3 \hat{P}(t_i | t_{i-1}, t_{i-2}) + \lambda_2 \hat{P}(t_i | t_{i-1}) + \lambda_1 \hat{P}(t_i)$
 - $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- With additional features, reach 96.5% accuracy on Penn Treebank (Brants, 2000)

42

Other Variant Taggers

- HMM is **generative**

- allows for unsupervised HMMs: learn model without any tagged data!

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T P(W|T)P(T) \\ &= \operatorname{argmax}_T \prod_i P(\text{word}_i|\text{tag}_i) \prod_i P(\text{tag}_i|\text{tag}_{i-1})\end{aligned}$$

43

Other Variant Taggers

$$\hat{T} = \operatorname{argmax}_T P(T|W)$$

Other Variant Taggers

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

can add more features

- **Discriminative** models describe $P(t|w)$ directly
 - ▶ supports richer feature set, generally better accuracy when trained over large supervised datasets
 - ▶ E.g., Maximum Entropy Markov Model (MEMM), Conditional random field (CRF), Connectionist Temporal Classification (CTC)
 - ▶ Most *deep learning* models of sequences are discriminative (e.g., encoder-decoders for translation), similar to an MEMM

44

HMMs in NLP

- **HMMs are highly effective for part-of-speech tagging**
 - ▶ trigram HMM gets 96.5% accuracy (TnT)
 - ▶ related models are state of the art
 - ▶ MEMMs 97%
 - ▶ CRFs 97.6%
 - ▶ Deep CRF 97.9%
 - ▶ *English Penn Treebank* tagging accuracy [https://aclweb.org/aclwiki/index.php?title=POS_Tagging_\(State_of_the_art\)](https://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art))
- **Apply out-of-the box to other sequence labelling tasks**
 - ▶ named entity recognition, shallow parsing, alignment ...
 - ▶ In other fields: DNA, protein sequences, image lattices...

45

A Final Word

- HMMs are a simple, yet effective way to perform sequence labelling.
- Can still be competitive, and fast. Natural baseline for other sequence labelling tasks.
- Main drawback: not very flexible in terms of feature representation, compared to MEMMs and CRFs.

46

Readings

- JM3 Appendix A A.1-A.2, A.4
- See also E18 Chapter 7.3
- References:
 - ▶ Rabiner's HMM tutorial <http://tinyurl.com/2hqaf8>
 - ▶ Lafferty et al, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)

47

Deep Learning for NLP: Feedforward Networks

COMP90042

Natural Language Processing

Lecture 7



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

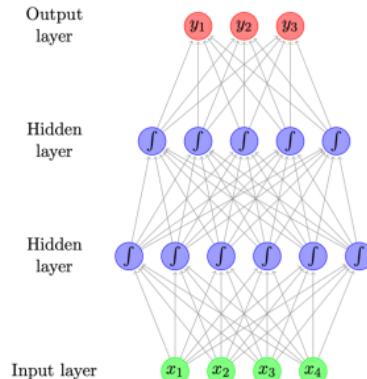
Deep Learning

- A branch of machine learning
- Re-branded name for neural networks
- Neural networks: historically inspired by the way computation works in the brain
 - Consists of computation units called neurons
- Why deep? Many layers are chained together in modern deep learning models

4

Feed-forward NN

- Aka multilayer perceptrons
- Each arrow carries a weight, reflecting its importance
- Sigmoid function represents a non-linear function



5

NN Units

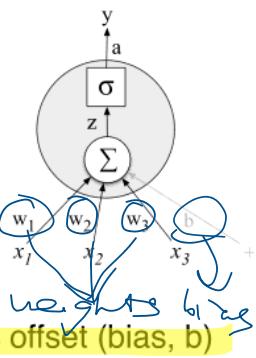
- Each “unit” is a function

- given input x , computes real-value (scalar) h

$$h = \tanh \left(\sum_j w_j x_j + b \right)$$

is a non-linear activation function

- scales input (with weights, w) and adds offset (bias, b)
- applies non-linear function, such as logistic sigmoid, hyperbolic sigmoid (\tanh), or rectified linear unit



6

Matrix Vector Notation

- Typically have several hidden units, i.e.

$$h_i = \tanh \left(\sum_j w_{ij} x_j + b_i \right)$$

- Each with its own weights (w_i) and bias term (b_i)

- Can be expressed using matrix and vector operators

$$\vec{h} = \tanh (W \vec{x} + \vec{b}) \quad \xrightarrow{\text{vector representation}}$$

- Where W is a matrix comprising the weight vectors, and b is a vector of all bias terms

- Non-linear function applied element-wise

7

Output Layer

- **Binary classification problem** (e.g. classify whether a tweet is positive or negative in sentiment):

► sigmoid activation function (aka logistic function) $\rightarrow 2 \text{ classes}$

- **Multi-class classification problem** (e.g. classify the topics of a document)

► softmax ensures probabilities > 0 and sum to 1

$$\left[\frac{\exp(v_1)}{\sum_i \exp(v_i)}, \frac{\exp(v_2)}{\sum_i \exp(v_i)}, \dots, \frac{\exp(v_m)}{\sum_i \exp(v_i)} \right]$$

8

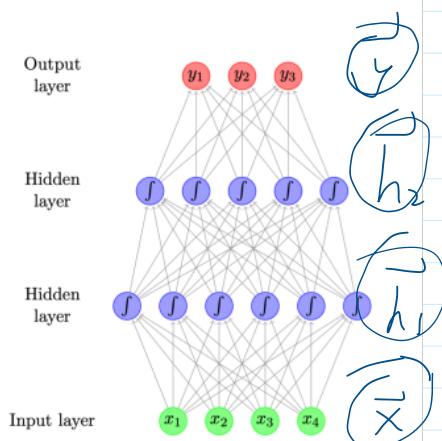
Feed-forward NN

$$\vec{h}_1 = \tanh(W_1 \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = \tanh(W_2 \vec{h}_1 + \vec{b}_2)$$

$$\vec{y} = \text{softmax}(W_3 \vec{h}_2)$$

- **Matrices and biases = parameters of the model**



9

Learning from Data

- How to learn the parameters from data?
- Consider how well the model “fits” the training data, in terms of the probability it assigns to the correct output

$$L = \prod_{i=0}^m P(y_i|x_i)$$

- ▶ want to **maximise** total probability, L
- ▶ equivalently **minimise** $-\log L$ with respect to parameters

- Trained using gradient descent

- ▶ tools like *tensorflow*, *pytorch*, *dynet* use autodiff to compute gradients automatically

→ define loss function properly

10

Topic Classification

- Given a document, classify it into a predefined set of topics (e.g. economy, politics, sports)
- Input: bag-of-words

	love	cat	dog	doctor
doc 1	0	2	3	0
doc 2	2	0	2	0
doc 3	0	0	0	4
doc 4	3	0	0	2

11

Topic Classification

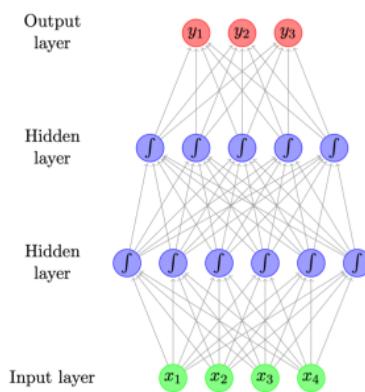
$$\vec{h}_1 = \tanh(W_1 \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = \tanh(W_2 \vec{h}_1 + \vec{b}_2)$$

$$\vec{y} = \text{softmax}(W_3 \vec{h}_2)$$

- $x = [0, 2, 3, 0]$ for the first document
- $y = [0.1, 0.6, 0.3]$: probability distribution over the 3 classes

argmax



12

Topic Classification - Improvements

- + Bag of bigrams as input
- Preprocess text to lemmatise words and remove stopwords
- Instead of raw counts, we can weight words using TF-IDF or indicators (0 or 1 depending on presence of words)

13

Authorship Attribution

- Given a document, infer the identity of its author or characteristics of the author (e.g. gender, age, native language)
- Stylistic properties of text are more important than content words in this task
 - POS tags and function words (e.g. *on, of, the, and*)
- Good approximation of function words: top-300 most frequent words in a large corpus
- Input: bag of function words, bag of POS tags, bag of POS bigrams, trigrams
- Word weighting: density (e.g. ratio between no. of function words and content words in a window of text)
- Other features: distribution of distances between consecutive function words

14

Language model (Recap)

- Assign a probability to a sequence of words
- Framed as “sliding a window” over the sentence, predicting each word from finite context
 - E.g., $n = 3$, a trigram model

$$P(w_1, w_2, \dots, w_m) = \prod_{i=1}^m P(w_i | w_{i-2}, w_{i-1})$$

- Training (estimation) from frequency counts
 - Difficulty with rare events → smoothing

15

Language Models as Classifiers

LMs can be considered simple classifiers, e.g.
trigram model

$$P(w_i | w_{i-2} = \text{"cow"}, w_{i-1} = \text{"eats"})$$

classifies the likely next word in a sequence.

16

Feed-forward NN Language Model

- Use neural network as a classifier to model
 $P(w_i | w_{i-2} = \text{"cow"}, w_{i-1} = \text{"eats"})$
 - input features = the previous two words
 - output class = the next word
- How to represent words? Embeddings

17

Word Embeddings

- Maps discrete word symbols to continuous vectors in a relatively low dimensional space
- Word embeddings allow the model to capture similarity between words
 - dog vs. cat
 - walking vs. running
- Alleviates data-sparsity problems

(100, 200, ...)

dog + cat = walking
dog - cat = running

18

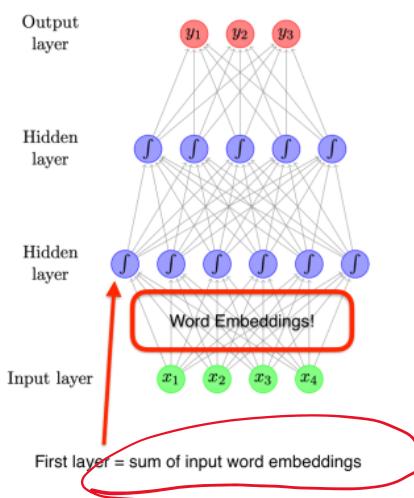
Topic Classification

$$\vec{h}_1 = \tanh(W_1 \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = \tanh(W_2 \vec{h}_1 + \vec{b}_2)$$

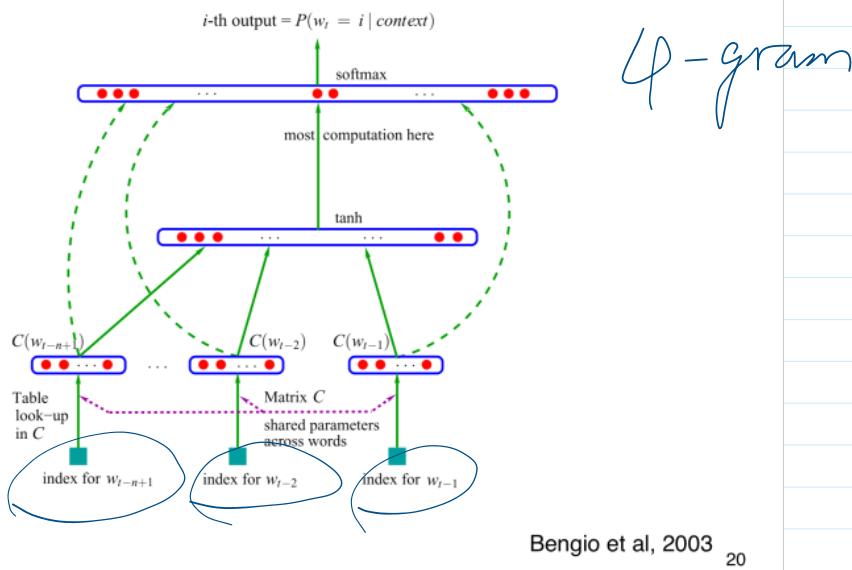
$$\vec{y} = \text{softmax}(W_3 \vec{h}_2)$$

- $x = [0, 2, 3, 0]$ for the first document
- $y = [0.1, 0.6, 0.3]$: probability distribution over the 3 classes



19

Language Model: Architecture



Example

$$P(w_i = \text{"grass"} \mid w_{i-2} = \text{"cow"}, w_{i-1} = \text{"eats"})$$

- Lookup word embeddings for cow and eats

rabbit	grass	eats	hunts	cow
0.9	0.2	-3.3	-0.1	-0.5
0.2	-2.3	0.6	-1.5	1.2
-0.6	0.8	1.1	0.3	-2.4
1.5	0.8	0.1	2.5	0.4

- Concatenate them and feed it to the network

$$\vec{x} = v_{\text{cow}} \oplus v_{\text{eats}}$$

$$\vec{h}_1 = \tanh(W_1 \vec{x} + b_1)$$

$$\vec{y} = \text{softmax}(W_2 \vec{h}_1)$$

hidden
output

Output

- y gives the probability distribution over all words in the vocabulary

0.01	0.80	0.05	0.10	0.04
rabbit	grass	eats	hunts	cow

$$P(w_i = \text{"grass"} | w_{i-2} = \text{"cow"}, w_{i-1} = \text{"eats"}) = 0.8$$

- Most parameters are in the word embeddings (size = $d \times |\mathcal{V}|$) and the output embeddings (size = $|\mathcal{V}| \times d$)

22

Example

$$P(w_i = \text{"grass"} | w_{i-2} = \text{"cow"}, w_{i-1} = \text{"eats"})$$

- Lookup word embeddings for cow and eats

rabbit	grass	eats	hunts	cow	
0.9	0.2	-3.3	-0.1	-0.5	
0.2	-2.3	0.6	-1.5	1.2	
-0.6	0.8	1.1	0.3	-2.4	
1.5	0.8	0.1	2.5	0.4	

Word embeddings
 $d \times |\mathcal{V}|$

- Concatenate them and feed it to the network

$$\vec{x} = v_{\text{cow}} \oplus v_{\text{eats}}$$

$$\vec{h}_1 = \tanh(W_1 \vec{x} + \vec{b}_1)$$

$$\vec{y} = \text{softmax}(W_2 \vec{h}_1)$$

Output word embeddings
 $|\mathcal{V}| \times d$

23

Why Bother?

- Ngram LMs
 - cheap to train (just compute counts)
 - problems with sparsity and scaling to larger contexts
 - don't adequately capture properties of words (grammatical and semantic similarity), e.g., film vs movie
- NNLMs more robust
 - force words through low-dimensional embeddings
 - automatically capture word properties, leading to more robust estimates
 - flexible: minor change to adapt to other tasks (tagging)

24

POS Tagging

- POS tagging can also be framed as classification:
 $P(t_i | w_{i-1} = "cow", w_i = "eats")$
 classifies the likely POS tag for "eats".
- Why not use a fancier classifier? (Neural net)
- NNLM architecture can be adapted to the task directly

25

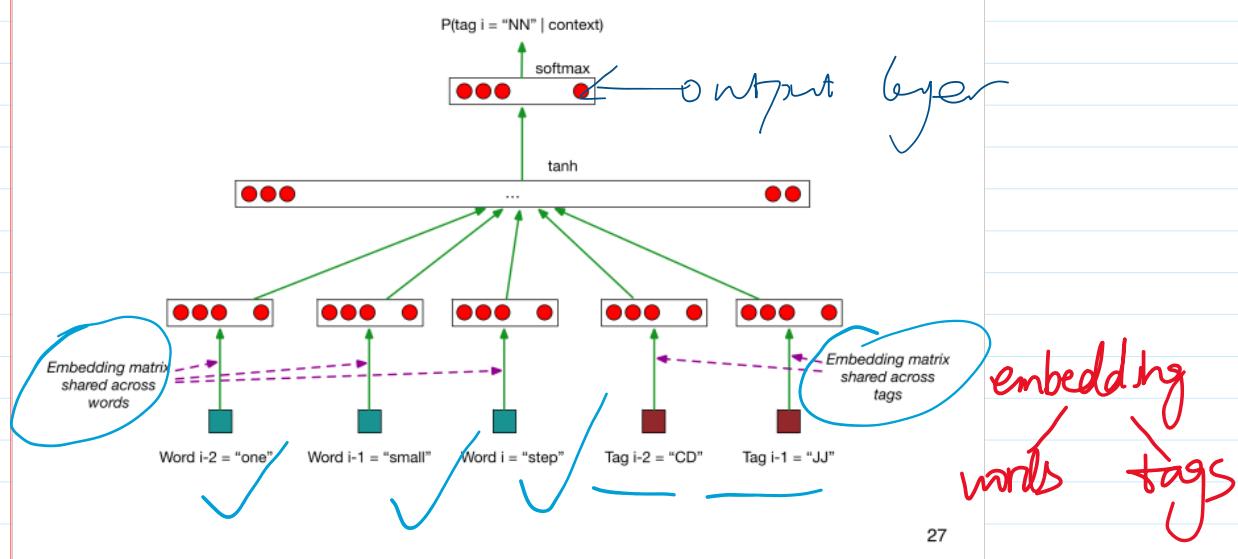
Feed-forward NN for Tagging

- MEMM tagger takes as input:
 - recent words w_{i-2}, w_{i-1}, w_i
 - recent tags t_{i-2}, t_{i-1}
- And outputs: current tag t_i
- Frame as neural network with
 - 5 inputs: 3 x word embeddings and 2 x tag embeddings
 - 1 output: vector of size ITI, using softmax
- Train to minimise

$$-\sum_i \log P(t_i | w_{i-2}, w_{i-1}, w_i, t_{i-2}, t_{i-1})$$

26

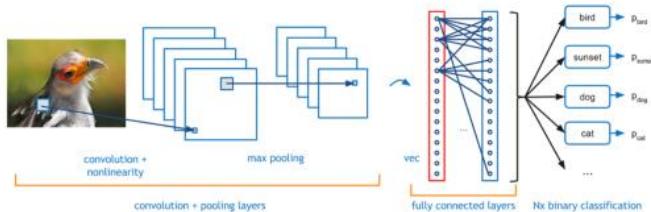
FF-NN for Tagging



27

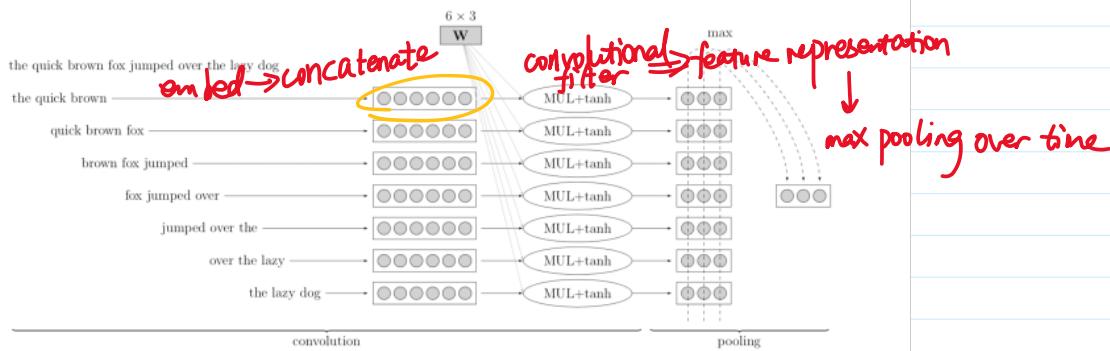
Convolutional Networks

- Commonly used in computer vision
- Identify indicative local predictors
- Combine them to produce a fixed-size representation



<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/> 28

Convolutional Networks for NLP



- Sliding window (e.g. 3 words) over sequence
- W = convolution filter (linear transformation+tanh)
- max-pool to produce a fixed-size representation

29

Final Words

- Neural networks
 - Robust to word variation, typos, etc
 - Excellent generalization
 - Flexible — customised architecture for different tasks
- Cons
 - Much slower than classical ML models... but GPU acceleration
 - Lots of parameters due to vocabulary size
 - Data hungry, not so good on tiny data sets
 - Pre-training on big corpora helps

30

Readings

- Feed-forward network: G15, section 4
- Convolutional network: G15, section 9

31



Deep Learning for NLP: Recurrent Networks

COMP90042

Natural Language Processing

Lecture 8



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L8

N-gram Language Models

- Can be implemented using counts (with smoothing) *sparsity*
- Can be implemented using feed-forward neural networks *predict next word given some context* *fixed*
- Generates sentences like (trigram model):
 - *I saw a table is round and about*

2

COMP90042

L8

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - *I saw a table is round and about*

3

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - *I saw a table is round and about*

4

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - *I saw a table is round and about*

5

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - *I saw a table is round and about*

6

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - ▶ *I saw a table is round and about*

7

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
 - ▶ *I saw a table is round and about*
- Problem: **limited context**
generate long sentences
→ not related to the front of sentence

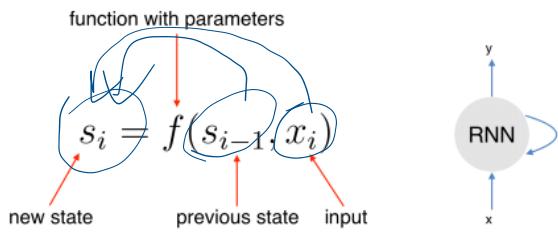
8

Recurrent Neural Network (RNN)

- **RNNs allow representing arbitrarily sized inputs**
- Core Idea: **processes the input sequence one at a time, by applying a recurrence formula**
- **Uses a state vector to represent contexts that have been previously processed**

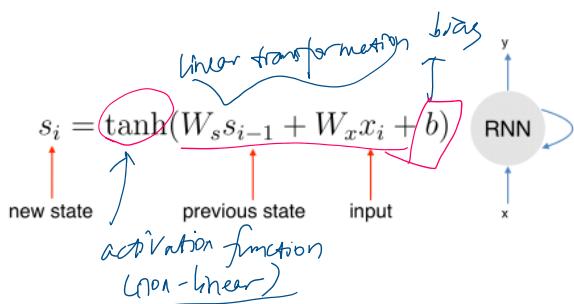
9

Recurrent Neural Network (RNN)



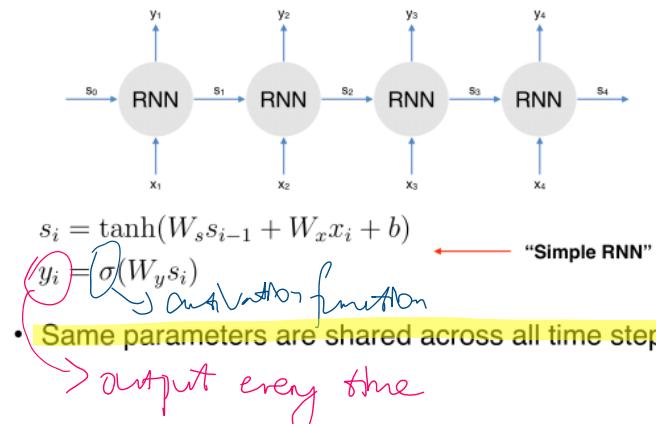
10

Recurrent Neural Network (RNN)



11

RNN Unrolled



12

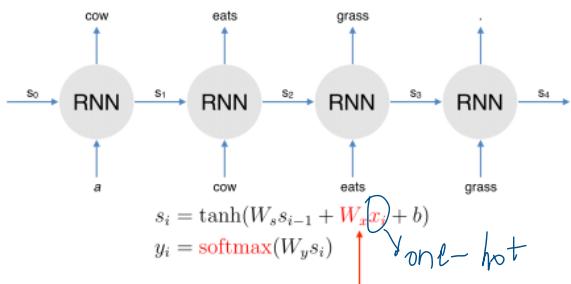
RNN Training

- An unrolled RNN is just a very deep neural network
- But same parameters are shared across many time steps
- To train RNN, we just need to create the unrolled computation graph given an input sequence
- And use backpropagation algorithm to compute gradients as usual
- This procedure is called backpropagation through time

$$\begin{aligned}
 s_4 &= R(s_3, x_4) \\
 &= R(\overbrace{R(s_2, x_3)}^{s_2}, x_4) \\
 &= R(R(\overbrace{R(s_1, x_2)}^{s_1}, x_3), x_4) \\
 &= R(R(R(\overbrace{R(s_0, x_1)}^{s_0}, x_2), x_3), x_4)
 \end{aligned}$$

13

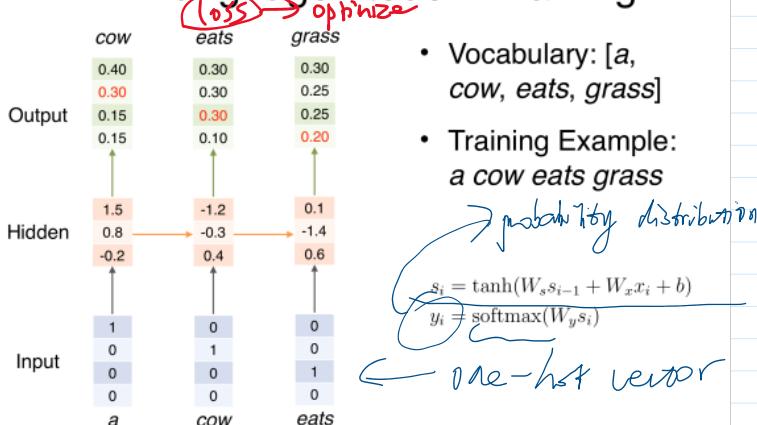
(Simple) RNN for Language Model



- Input words mapped to an embedding
- Output = next word

14

RNN Language Model - Training



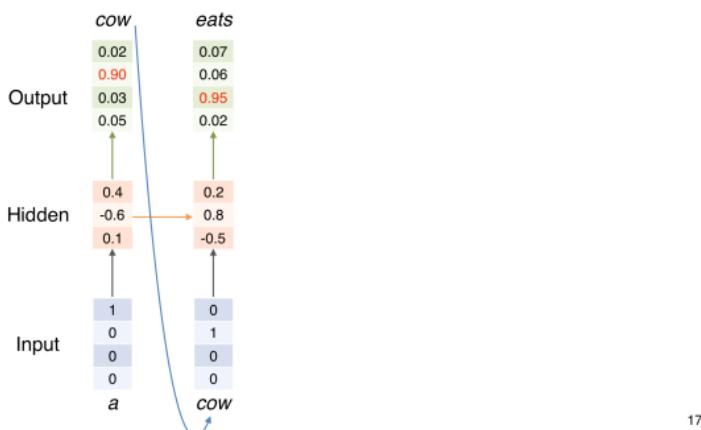
15

RNN Language Model - Generation



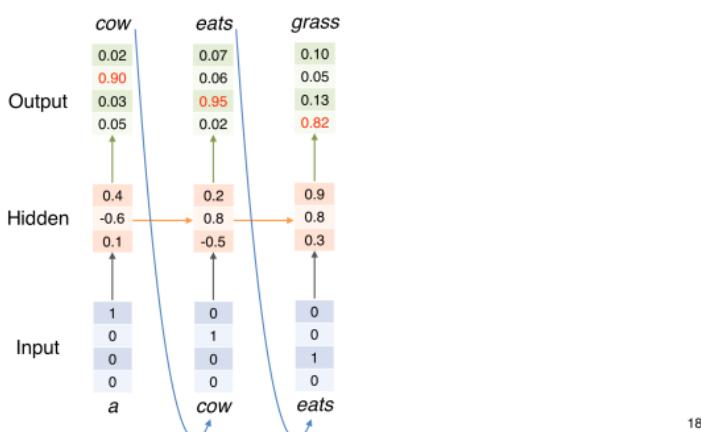
16

RNN Language Model - Generation



17

RNN Language Model - Generation



18

Language Model... Solved?

- RNN has the capability to model infinite context
- But can it actually capture long-range dependencies in practice?
- No... due to "vanishing gradients"
- Gradients in later steps diminish quickly during backpropagation
- Earlier inputs do not get much update

19

Long Short-term Memory (LSTM)

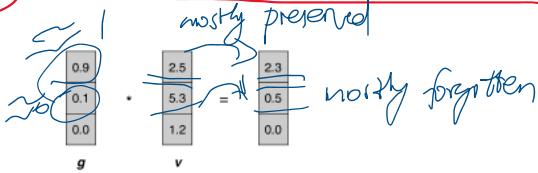
- LSTM is introduced to solve vanishing gradients
- Core idea: have "memory cells" that preserve gradients across time
- Access to the memory cells is controlled by "gates"
- For each input, a gate decides:
 - ▶ how much the new input should be written to the memory cell
 - ▶ and how much content of the current memory cell should be forgotten

20

→ tackle vanishing problem

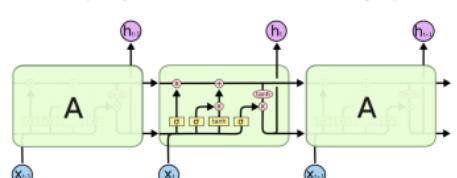
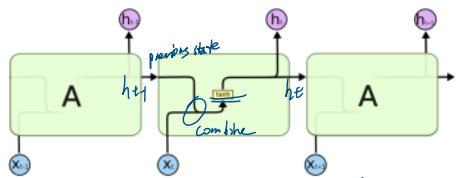
Long Short-term Memory (LSTM)

- A gate g is a vector
 - ▶ each element has values between 0 to 1
- g is multiplied component-wise with vector v , to determine how much information to keep for v
- Use sigmoid function to keep values of g close to either 0 or 1



21

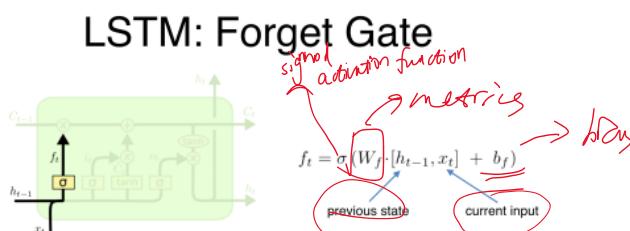
LSTM vs. Simple RNN



<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

22

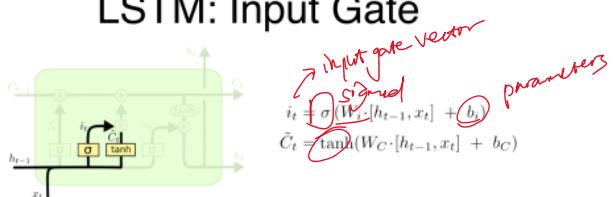
LSTM: Forget Gate



- Controls how much information to "forget" in the memory cell (C_{t-1})
- The cats that the boy likes
- Memory cell was storing pronoun information (cats)
- The cell should now forget cats and store boy to correctly predict the singular verb likes

23

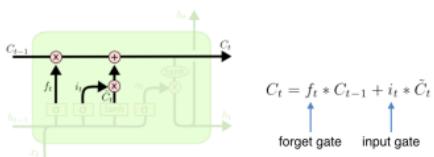
LSTM: Input Gate



- Input gate controls how much new information to put to memory cell
- \tilde{C}_t = new distilled information to be added
 - e.g. information about boy

24

LSTM: Update Memory Cell



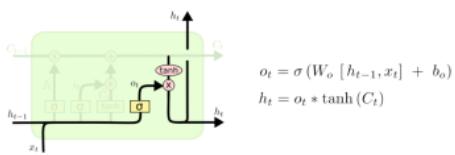
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

forget gate input gate

- Use the forget and input gates to update memory cell

25

LSTM: Output Gate



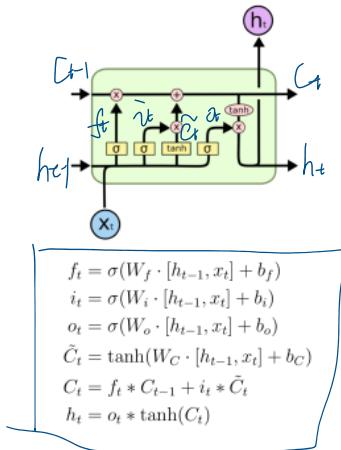
$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

- Output gate controls how much to distill the content of the memory cell to create the next state (h_t)

26

LSTM: Summary



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

27

Variants

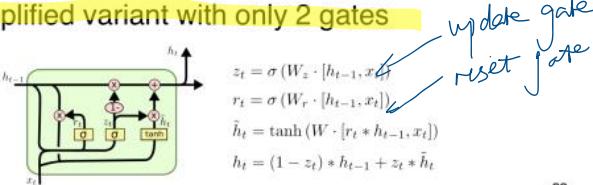
- Peephole connections

- ▶ Allow gates to look at cell state

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

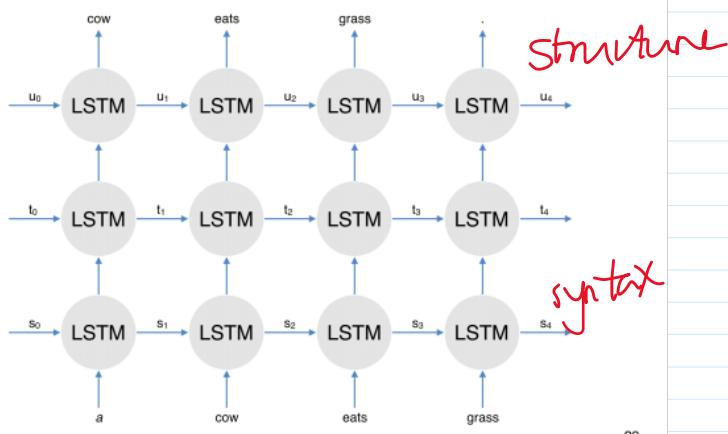
- Gated recurrent unit (GRU)

- ▶ Simplified variant with only 2 gates



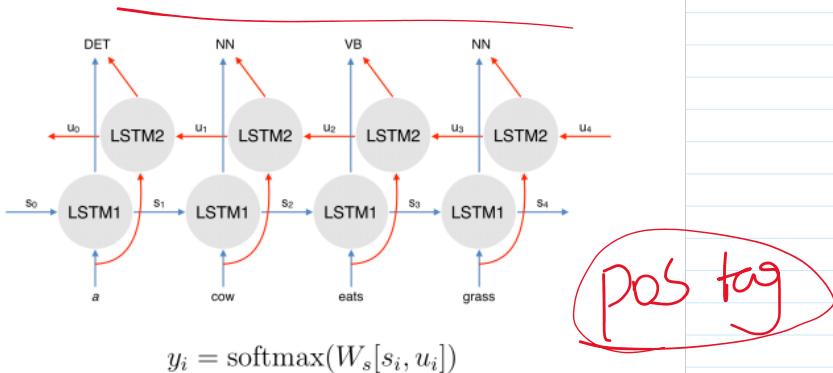
28

Multi-layer LSTM



29

Bidirectional LSTM



30

Shakespeare Generator

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:

- Training data = all works of Shakespeare
- Model: 3-layer **character RNN**, hidden dimension = 512

Shakespeare Generator

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nus begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

- Training data = all works of Shakespeare
- Model: 3-layer **character RNN**, hidden dimension = 512

generate by character

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/> 31

Wikipedia Generator

```
Nationalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangsham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immigrants]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrals]], [[Cynth's Daoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slot of Syrian influence that
was a famous German movement based on a more popular servacious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)|http://www.human.yahoo.com/guardian.cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]
```

- Training data = 100MB of Wikipedia raw data

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/> 32

Code Generator

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /* The kernel blank will could it to userspace.
        */
        if (as->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_8B(in.addr);
    selector = seg / 16;
    setup.works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i+1];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    w->name = "GetJobregs";
    bprm_self_clear(&w->version);
    reg->new = blocks(BPF_STATS << info->historidac) | PFMB_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/> 33

Deep-Speare

- Generates Shakespearean sonnets

```
python sonnet_gen.py -m trained_model/ -d 1
```

```
Temperature = 0.6 ~ 0.8
01 [0.43] with joyous gambols gay and still array
02 [0.44] no longer when he twas, while in his day
03 [0.00] at first to pass in all delightful ways
04 [0.40] around him, charming and of all his days
```

```
python sonnet_gen.py -m trained_model/ -d 2
```

```
Temperature = 0.6 ~ 0.8
01 [0.44] shall i behold him in his cloudy state
02 [0.00] for just but tempteth me to stop and pray
03 [0.00] a cry: if it will drag me, find no way
04 [0.40] from pardon to him, who will stand and wait
```

<https://github.com/jblas/deepspeare> 34

Computers produce poetry by the metre



Sonnets written by a computer are superior to Shakespeare's in more respects – and the general public is unable to tell the difference between the two, researchers say.

Computer scientists have trained a "neural network" using 2600 sonnets taken from a free online database and put the system to work

Daily Mail AUSTRALIA

Can YOU spot the real Shakespearean sonnet? The AI learning how write its own poetry

- Researchers at IBM used 2600 sonnets to train their AI
- Researchers asked workers from a crowdsourcing website to rate them
- Found they were unable to distinguish them from the real deal
- However, experts were more critical and said they still lacked real emotion

By MARK PRYDE FOR DAILYMAIL.COM ©

PUBLISHED: 05:25 AEST, 26 July 2018 | UPDATED: 05:56 AEST, 26 July 2018

View comments

50

Share

Print

Comment

Like

Link

Embed

Report

Save

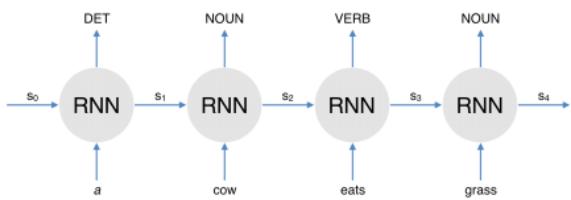
Comment

Like

Link

Sequence Labeling

- RNNs work particularly for sequence labelling problems, e.g. POS tagging



37

Final Words

- Pros
 - Has the ability to capture long range contexts
 - Excellent generalisation
 - Just like feedforward networks: flexible, so it can be used for all sorts of tasks
 - Common component in a number of NLP tasks
- Cons
 - Slower than feedforward networks due to sequential processing
 - In practice still doesn't capture long range dependency very well (evident when generating long text)

38

Readings

- G15, Section 10 & 11

39



Lexical Semantics

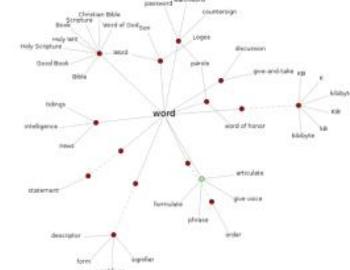
COMP90042

Natural Language Processing

Lecture 9



THE UNIVERSITY OF
MELBOURNE



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

Sentiment Analysis

- Bag of words, kNN classifier. Training data:

- ▶ “This is a good movie.” → 😊
- ▶ “This is a great movie.” → 😊
- ▶ “This is a terrible film.” → 😞

- “This is a wonderful film.” → ?

- Two problems:

- ▶ The model does not know that “movie” and “film” are synonyms. Since “film” appears only in negative examples the model learns that it is a negative word.
- ▶ “wonderful” is not in the vocabulary (OOV – Out-Of-Vocabulary).

2

Sentiment Analysis

- Comparing words directly will not work. How to make sure we compare word meanings instead?
- Solution: add this information explicitly through a lexical database.

3

Word Semantics

- Lexical semantics (this lecture)
 - ▶ How the meanings of words connect to one another.
 - ▶ Manually constructed resources: lexicons, thesauri, ontologies, etc.
- Distributional semantics (next)
 - ▶ How words relate to each other in the text.
 - ▶ Automatically created resources from corpora.

4

What Do Words Mean?

- Referents in the physical or social world
 - ▶ But not usually useful in text analysis
- Their dictionary definition
 - ▶ But dictionary definitions are necessarily circular
 - ▶ Only useful if meaning is already understood
 - red* *n.* the color of blood or a ruby.
 - blood* *n.* the red liquid that circulates in the heart, arteries and veins of animals.
- Their relationships with other words
 - ▶ Also circular, but more practical

5

Word Senses

- A word sense describes one aspect of the meaning of a word

mouse¹ : a *mouse* controlling a computer system in 1968.

mouse² : a quiet animal like a *mouse*

bank¹ : ...a *bank* can hold the investments in a custodial account ...

bank² : ...as agriculture burgeons on the east *bank*, the river ...

6

Word Glosses

- Gloss: textual definition of a sense, given by a dictionary
- *Bank*:
 - ▶ financial institution that accepts deposits and channels the money into lending activities
 - ▶ sloping land (especially the slope beside a body of water)
- If a word has multiple senses, it is polysemous

7

Meaning Through Relations

- Another way to define meaning: by looking at how it relates to other words
- **Synonymy:** near identical meaning
 - ▶ *vomit* vs. *throw up*
 - ▶ *big* vs. *large*
- **Antonymy:** opposite meaning
 - ▶ *long* vs. *short*
 - ▶ *big* vs. *little*

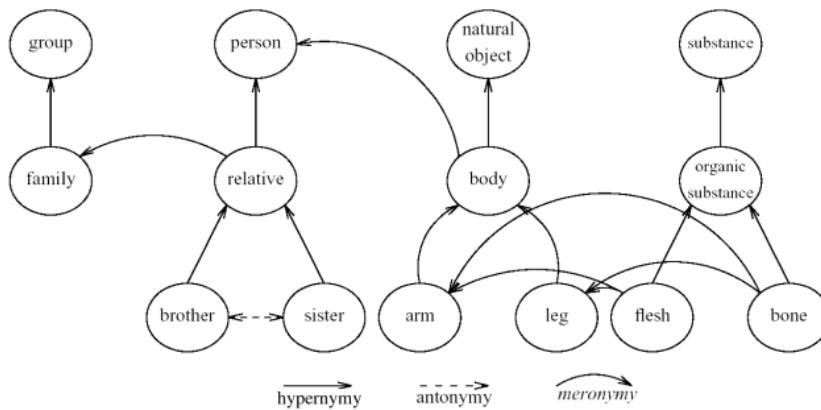
8

Meaning Through Relations (2)

- **Hypernymy:** *is-a* relation
 - ▶ *cat* is an *animal*
 - ▶ *mango* is a *fruit*
- **Meronymy:** part-whole relation
 - ▶ *leg* is part of a *chair*
 - ▶ *wheel* is part of a *car*

9

Meaning Through Relations (3)



10

WordNet

- A database of lexical relations
- English WordNet includes ~120,000 nouns, ~12,000 verbs, ~21,000 adjectives, ~4,000 adverbs
- On average: noun has 1.23 senses; verbs 2.16
- WordNets available in most major languages (www.globalwordnet.org, <https://babelnet.org/>)
- English version freely available (accessible via NLTK)

11

WordNet Example

The noun “bass” has 8 senses in WordNet.

1. bass¹ - (the lowest part of the musical range)
2. bass², bass part¹ - (the lowest part in polyphonic music)
3. bass³, basso¹ - (an adult male singer with the lowest voice)
4. sea bass¹, bass⁴ - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass¹, bass⁵ - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass⁶, bass voice¹, basso² - (the lowest adult male singing voice)
7. bass⁷ - (the member with the lowest range of a family of musical instruments)
8. bass⁸ - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

J&M3 Ch. 19

12

Synsets

- Nodes of WordNet are not words or lemmas, but senses
- There are represented by sets of synonyms, or synsets
- Bass synsets:
 - ▶ {bass¹, deep⁶}
 - ▶ {bass⁶, bass voice¹, basso²}
- Another synset:
 - ▶ {chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²}
 - ▶ Gloss: a person who is gullible and easy to take advantage of

13

Synsets (2)

```
>>> nltk.corpus.wordnet.synsets('bank')
```

[Synset('bank.n.01'), Synset('depository_financial_institution.n.01'), Synset('bank.n.03'),
 Synset('bank.n.04'), Synset('bank.n.05'), Synset('bank.n.06'), Synset('bank.n.07'),
 Synset('savings_bank.n.02'), Synset('bank.n.09'), Synset('bank.n.10'), Synset('bank.v.01'),
 Synset('bank.v.02'), Synset('bank.v.03'), Synset('bank.v.04'), Synset('bank.v.05'), Synset('deposit.v.02'),
 Synset('bank.v.07'), Synset('trust.v.01')]

```
>>> nltk.corpus.wordnet.synsets('bank')[0].definition()
```

u'sloping land (especially the slope beside a body of water)'

```
>>> nltk.corpus.wordnet.synsets('bank')[1].lemma_names()
```

[u'depository_financial_institution', u'bank', u'banking_concern', u'banking_company']

14

Lexical Relations in WordNet

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	<i>breakfast</i> ¹ → <i>meal</i> ¹
Hyponym	Subordinate	From concepts to subtypes	<i>meal</i> ¹ → <i>lunch</i> ¹
Instance Hypernym	Instance	From instances to their concepts	<i>Austen</i> ¹ → <i>author</i> ¹
Instance Hyponym	Has-Instance	From concepts to their instances	<i>composer</i> ¹ → <i>Bach</i> ¹
Part Meronym	Has-Part	From wholes to parts	<i>table</i> ² → <i>leg</i> ³
Part Holonym	Part-Of	From parts to wholes	<i>course</i> ⁷ → <i>meal</i> ¹
Antonym		Semantic opposition between lemmas	<i>leader</i> ¹ ↔ <i>follower</i> ¹
Derivation		Lemmas w/same morphological root	<i>destruction</i> ¹ ↔ <i>destroy</i> ¹

15

Hypernymy Chain

bass³, basso (an adult male singer with the lowest voice)

=> singer, vocalist, vocalizer, vocaliser

=> musician, instrumentalist, player

=> performer, performing artist

=> entertainer

=> person, individual, someone...

=> organism, being

=> living thing, animate thing,

=> **whole, unit**

=> object, physical object

=> physical entity

=> entity

bass⁷ (member with the lowest range of a family of instruments)

=> musical instrument, instrument

=> device

=> instrumentality, instrumentation

=> artifact, artefact

=> **whole, unit**

=> object, physical object

=> physical entity

=> entity

16

Word Similarity

17

Word Similarity

- Synonymy: *film* vs. *movie*
- What about *show* vs. *film*? *opera* vs. *film*?
- Unlike synonymy (which is a binary relation), word similarity is a spectrum
- We can use lexical database (e.g. WordNet) or thesaurus to estimate word similarity

18

Word Similarity with Paths

- Given WordNet, find similarity based on path length
- $\text{pathlen}(c_1, c_2) = 1 + \text{edge length in the shortest path between sense } c_1 \text{ and } c_2$
- similarity between two senses:
 - ▶ $\text{simpath}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}$
- similarity between two words
 - ▶ $\text{wordsim}(w_1, w_2) = \max_{c_1 \in \text{senses}(w_1), c_2 \in \text{senses}(w_2)} \text{simpath}(c_1, c_2)$

19

Examples

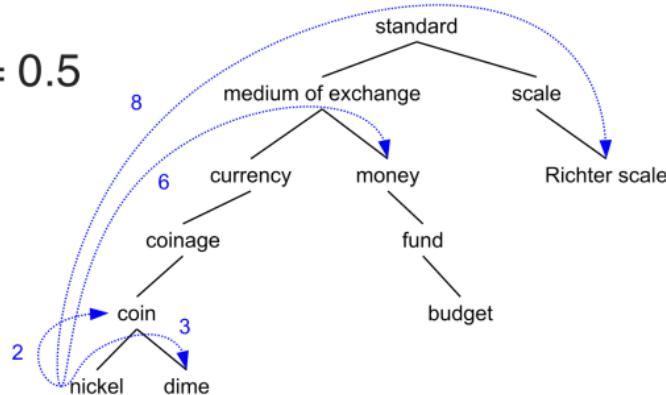
$$\text{simpath}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}$$

$$\text{simpath}(\text{nickel}, \text{coin}) = 1/2 = 0.5$$

$$\begin{aligned} \text{simpath}(\text{nickel}, \text{currency}) \\ = 1/4 = 0.25 \end{aligned}$$

$$\begin{aligned} \text{simpath}(\text{nickel}, \text{money}) \\ = 1/6 = 0.17 \end{aligned}$$

$$\begin{aligned} \text{simpath}(\text{nickel}, \text{Richter scale}) \\ = 1/8 = 0.13 \end{aligned}$$



20

Beyond Path Length

- Problem: edges vary widely in actual semantic distance
 - Much bigger jumps near top of hierarchy
- Solution 1: include depth information (Wu & Palmer)
 - Use path to find lowest common subsumer (LCS)
 - Compare using depths

$$\text{simwup}(c_1, c_2) = \frac{2 * \text{depth}(\text{LCS}(c_1, c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}$$

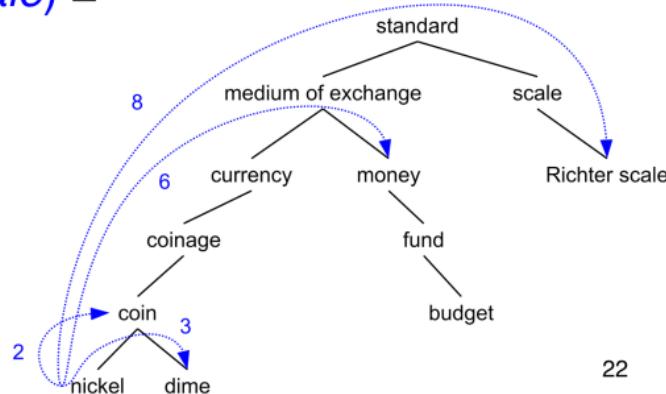
21

Examples

$$\text{simwup}(c_1, c_2) = \frac{2 * \text{depth}(\text{LCS}(c_1, c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}$$

$$\text{simwup}(\text{nickel}, \text{money}) = 2 * 2 / (3 + 6) = 0.44$$

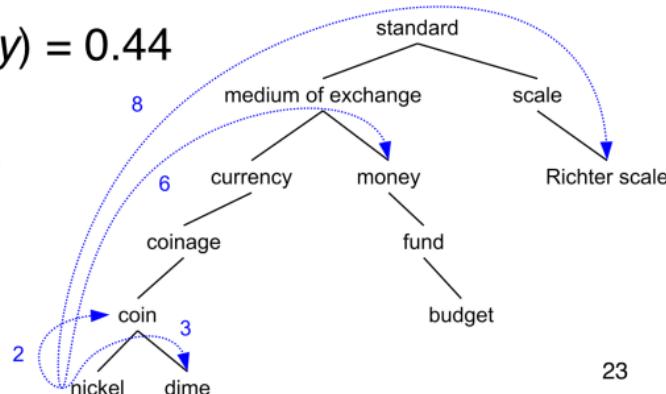
$$\text{simwup}(\text{nickel}, \text{Richter scale}) = \\ 2 * 1 / (3 + 6) = 0.22$$



22

Abstract Nodes

- But count of edges or node depth is still poor semantic distance metric
- Nodes high in the hierarchy is very abstract/general
- How do we make words that connect through very abstract nodes much less similar
 - ▶ $\text{simwup}(\text{nickel}, \text{money}) = 0.44$
 - ▶ $\text{simwup}(\text{nickel}, \text{Richter scale}) = 0.22$

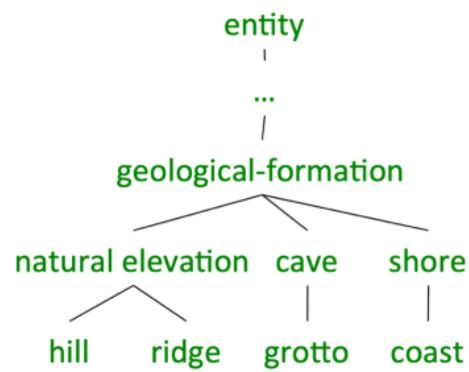


23

Concept Probability

$$P(c) = \frac{\sum_{w \in \text{words}(c))} \text{count}(w)}{N}$$

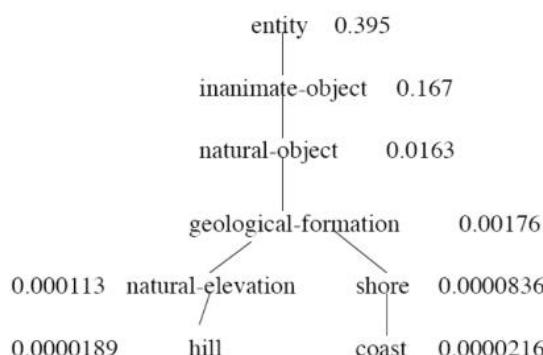
- $P(c)$: probability that a randomly selected word in a corpus is an instance of concept c
- $\text{words}(c)$: set of all words that are children of c
- $\text{words}(\text{geological-formation}) = \{\text{hill}, \text{ridge}, \text{grotto}, \text{coast}, \text{natural elevation}, \text{cave}, \text{shore}\}$
- $\text{words}(\text{natural elevation}) = \{\text{hill}, \text{ridge}\}$



24

Example

- Abstract nodes higher in the hierarchy has a higher $P(c)$



25

Similarity with Information Content

$$IC(c) = -\log P(c)$$

use IC instead of depth in simwup

$$\text{simlin}(c_1, c_2) = \frac{2 \times IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}$$

$$\text{simlin}(\text{hill}, \text{coast}) = \frac{2 \times -\log P(\text{geological-formation})}{-\log P(\text{hill}) - \log P(\text{coast})}$$

$$= \frac{-2 \log 0.00176}{-\log 0.0000189 - \log 0.0000216}$$

$$= 0.587$$

if LCS node is very high up in the hierarchy
(say $P(c) = 0.99$), then IC will be very low
(0.01 in this case)

entity	0.395
inanimate-object	0.167
natural-object	0.0163
geological-formation	0.00176
0.000113 natural-elevation	
0.0000189 hill	
0.0000836 shore	
0.0000216 coast	

26

Sentiment Analysis Revisited

- “This is a great movie.” → 😊
- “This is a wonderful film.” → ?
- Comparing words using WordNet paths work well if our classifier is based on word similarities (such as kNN)
- But what if we want sense as a general feature representation, so we can employ other classifiers?
- Solution: map words in text to senses in WordNet explicitly.

27

Word Sense Disambiguation

- Task: selects the correct sense for words in a sentence
- Baseline:
 - ▶ Assume the most popular sense
- Good WSD potentially useful for many tasks in NLP
 - ▶ In practice, often ignored because good WSD too hard
 - ▶ Active research area

28

Supervised WSD

- Apply standard machine classifiers
- Feature vectors typically words and syntax around target
 - ▶ But context is ambiguous too!
 - ▶ How big should context window be? (typically very small)
- Requires sense-tagged corpora
 - ▶ E.g. SENSEVAL, SEMCOR (available in NLTK)
 - ▶ Very time consuming to create!

29

Less Supervised Approaches

Less Supervised Approaches

- Lesk: Choose sense whose dictionary gloss from WordNet most overlaps with the context
- The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.
- **bank**: 2 overlapping non-stopwords, *deposits* and *mortgage*
- **bank**: 0

bank ¹	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank ²	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

30

Other Databases - FrameNet

- Based on **frame semantics**
 - ▶ *Mary bought a car from John*
 - ▶ *John sold a car to Mary*
 - ▶ Same situation (semantic frame), just different perspective
- A lexical database of **frames**, typically prototypical situations
 - ▶ E.g. "commerce_buy", "apply_heat"

31

FrameNet

- Includes lists of *lexical units* that evoke the frame
 - ▶ E.g. *cook*, *fry*, *bake*, *boil*, etc.
- Lists of *semantic roles* or *frame elements*
 - ▶ E.g. “the cook”, “the food”, “the container”, “the instrument”
- Semantic relationships among frames
 - ▶ “apply_heat” is Causative of “absorb_heat”, is Used by “cooking_creation”

32

Moving On To The Corpus

- Manually-tagged lexical resources an important starting point for text analysis
- But much modern work attempts to derive semantic information directly from corpora, without human intervention
- Distributional semantics!

33

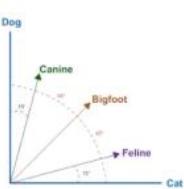
Reading

- JM3 Ch 19.1-19.3, 19.4.1, 19.5.1



Distributional Semantics

COMP90042
Natural Language Processing
Lecture 10



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L10

Lexical Databases - Problems

- Manually constructed
 - Expensive
 - Human annotation can be biased and noisy
- Language is dynamic
 - New words: slang, terminology, etc.
 - New senses
- The Internet provides us with massive amounts of text. Can we use that to obtain word meanings?

2

Distributional Hypothesis

- “You shall know a word by the company it keeps” (Firth, 1957)
- Document co-occurrence often indicative of topic
(document as context)
 - ▶ E.g. voting and politics
- Local context reflects a word’s semantic class
(word window as context)
 - ▶ E.g. eat a pizza, eat a burger

3

Guessing Meaning from Context

- Learn unknown word from its usage
- E.g., *tezgüino*
 - (14.1) A bottle of _____ is on the table.
 - (14.2) Everybody likes _____.
 - (14.3) Don’t have _____ before you drive.
 - (14.4) We make _____ out of corn.
- Look at other words in same (or similar) contexts

	(14.1)	(14.2)	(14.3)	(14.4)	...
<i>tezgüino</i>	1	1	1	1	
<i>loud</i>	0	0	0	0	
<i>motor oil</i>	1	0	0	1	
<i>tortillas</i>	0	1	0	1	
<i>choices</i>	0	1	0	0	
<i>wine</i>	1	1	1	0	

4

Word Vectors

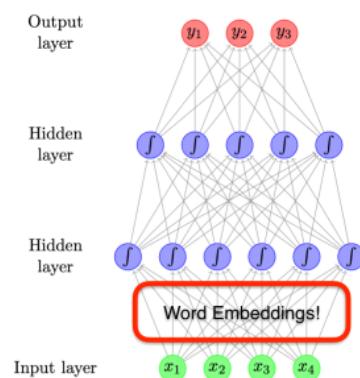
	(14.1)	(14.2)	(14.3)	(14.4)	...
<i>tezgūino</i>	1	1	1	1	
<i>loud</i>	0	0	0	0	
<i>motor oil</i>	1	0	0	1	
<i>tortillas</i>	0	1	0	1	
<i>choices</i>	0	1	0	0	
<i>wine</i>	1	1	1	0	

- Each row can be thought of a **word vector**
- It describes the **distributional properties**
- Capture all sorts of semantic relationships
(synonymy, analogy, etc)

5

Word Embeddings?

- We've seen word vectors before: word embeddings!
- Here we will learn other ways to **produce word vectors**
 - ▶ Count-based methods
 - ▶ More efficient neural methods designed just for learning word vectors



6

Count-Based Methods

7

The Vector Space Model

- Fundamental idea: represent meaning as a vector
- Consider documents as context
- One matrix, two viewpoints
 - ▶ Documents represented by their words
 - ▶ Words represented by their documents

	...	state	fun	heaven	...
...					
425		0	1	0	
426		3	0	0	
427		0	0	0	
.....					

8

Manipulating the VSM

- Weighting the values (beyond frequency)
- Creating low-dimensional dense vectors
- Comparing vectors

↓
sparcity

9

Tf-idf

- Standard weighting scheme for information retrieval
- Discounts common words

	...	the	country	hell	...
...					
425		43	5	1	
426		24	1	0	
427		37	0	3	
...					
df		500	14	7	

tf matrix

	...	the	country	hell	...
...					
425		0	26.0	6.2	
426		0	5.2	0	
427		0	0	18.6	
...					

tf-idf matrix

10

Dimensionality Reduction

- Term-document matrices are very sparse
- Dimensionality reduction: create shorter, denser vectors
- More practical (less features)
- Remove noise (less overfitting)

11

Singular Value Decomposition

$$A = U \Sigma V^T$$

(term-document matrix)

$|D|$

$|V|$

$$\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

A

$|U|$

U

(new term matrix)

m

Σ

(singular values)

m

V^T

(new document matrix)

$|D|$

$|V|$

$$\begin{bmatrix} 2.2 & 0.3 & \dots & 8.7 \\ 5.5 & -2.8 & \dots & 0.1 \\ -1.3 & 3.7 & \dots & 3.5 \\ \vdots & & \ddots & \vdots \\ 2.9 & -2.1 & \dots & -1.9 \end{bmatrix}$$

m

$$\begin{bmatrix} 9.1 & 0 & 0 & \dots & 0 \\ 0 & 4.4 & 0 & \dots & 0 \\ 0 & 0 & 2.3 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0.1 \end{bmatrix}$$

m

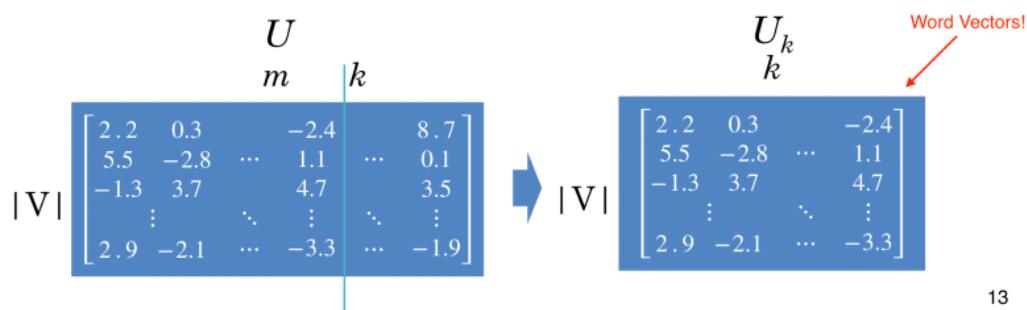
$$\begin{bmatrix} -0.2 & 4.0 & \dots & -1.3 \\ -4.1 & 0.6 & \dots & -0.2 \\ 2.6 & 6.1 & \dots & 1.4 \\ \vdots & & \ddots & \vdots \\ -1.9 & -1.8 & \dots & 0.3 \end{bmatrix}$$

$m = \text{Rank}(A)$

12

Truncating – Latent Semantic Analysis

- Truncating U , Σ , and V to k dimensions produces best possible k rank approximation of original matrix
- So truncated, U_k (or V_k^T) is a new low dimensional representation of the word
- Typical values for k are 100-5000



Words as Context

- Lists how often words appear with other words
 - In some predefined context (usually a window)
- The obvious problem with raw frequency: dominated by common words

	...	the	country	hell	...
...					
state		1973	10	1	
fun		54	2	0	
heaven		55	1	3	
.....					

14

Pointwise Mutual Information

- For two events x and y , PMI computes the discrepancy between:

- Their joint distribution
- Their individual distributions (assuming independence)

$$PMI(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$

15

Calculating PMI

	...	the	country	hell	...	Σ
...		1973	10	1		12786
state		54	2	0		633
fun		55	1	3		627
heaven						
...						
Σ		1047519	3617	780		15871304

$$p(x,y) = \text{count}(x,y) / \Sigma$$

$$p(x) = \Sigma_x / \Sigma$$

$$p(y) = \Sigma_y / \Sigma$$

$$x = state, y = country$$

$$p(x,y) = 10/15871304 = 6.3 \times 10^{-7}$$

$$p(x) = 12786/15871304 = 8.0 \times 10^{-4}$$

$$p(y) = 3617/15871304 = 2.3 \times 10^{-4}$$

$$PMI(x,y) = \log_2(6.3 \times 10^{-7}) / ((8.0 \times 10^{-4})(2.3 \times 10^{-4})) \\ = 1.78$$

16

PMI Matrix

- PMI does a better job of capturing interesting semantics
 - ▶ E.g. *heaven* and *hell*
- But it is obviously biased towards rare words
- And doesn't handle zeros well

	...	the	country	hell	...
...					
state		1.22	1.78	0.63	
fun		0.37	3.79	-inf	
heaven		0.41	2.80	6.60	
.....					

17

PMI Tricks

- Zero all negative values (PPMI)
 - ▶ Avoid -inf and unreliable negative values
- Counter bias towards rare events
 - ▶ Smooth probabilities

18

SVD

	...	the	country	hell	...
...					
425	0	26.0	6.2		
426	0	5.2	0		
427	0	0	18.6		
...					

tf-idf matrix

	...	the	country	hell	...
...					
state		1.22	1.78	0.63	
fun		0.37	3.79	0	
heaven		0.41	2.80	6.60	
.....					

PPMI matrix

- Regardless of whether we use document or word as context, SVD can be applied to create dense vectors

19

Similarity

- Word similarity = comparison between word vectors (e.g. cosine similarity)
- Find synonyms, based on proximity in vector space
 - ▶ automatic construction of lexical resources
- Use vectors as features in classifier — more robust to different inputs (movie vs film)

20

Neural Methods

21

Word Embeddings

- We've seen word embeddings used in neural networks (feedforward or recurrent)
- But these models are designed for other tasks:
 - ▶ Classification
 - ▶ Language modelling
- Word embeddings is just one part of the model

22

Neural Models for Embeddings

- Can we design neural networks whose goal is to learn word embeddings?
- Desiderata:
 - Unsupervised
 - Efficient
 - Useful representation

23

Word2Vec

- Neural network inspired approaches seek to learn vector representations of words and their contexts
- Key idea
 - Word embeddings should be similar to embeddings of neighbouring words
 - And dissimilar to other words that don't occur nearby
- Using vector dot product for vector 'comparison'
 - $u \cdot v = \sum_j u_j v_j$

24

Word2Vec

- Framed as learning a classifier

- ▶ **Skip-gram:** predict words in local context surrounding given word



- ▶ **CBOW:** predict word in centre, given words in the local surrounding context

- Local context means words within L positions, L=2 above

25

Skip-gram Model

- Generates each word in context given centre word



- Total probability defined as

- Where subscript denotes position in running text

$$\prod_{l \in -L, \dots, -1, 1, \dots, L} P(w_{t+l} | w_t)$$

target

- Using a logistic regression model

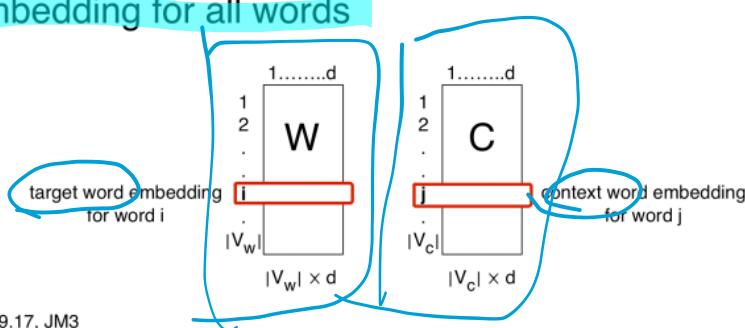
$$P(w_k | w_j) = \frac{\exp(c_{w_k} \cdot v_{w_j})}{\sum_{w' \in V} \exp(c_{w'} \cdot v_{w_j})}$$

26

*word embedding
dot product*

Embedding parameterisation

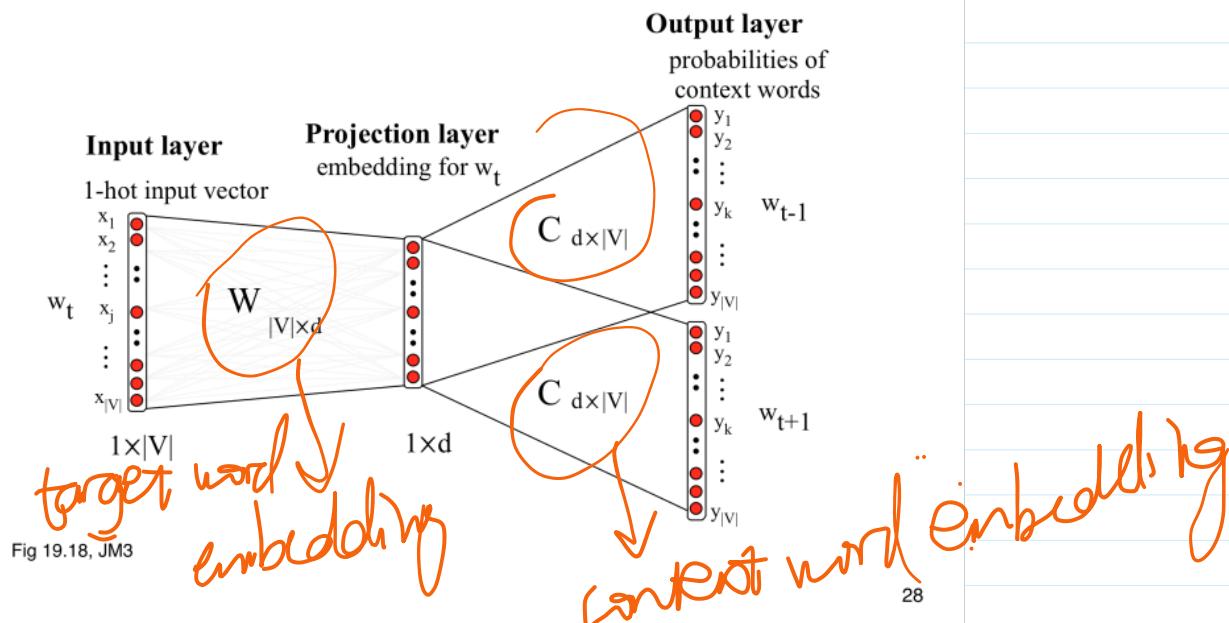
- Two parameter matrices, with **d-dimensional embedding for all words**



- Words are numbered**, e.g., by sorting vocabulary and using word location as its index

27

Skip-gram model



28

Training the skip-gram model

- Train to maximise likelihood of raw text
- Too slow in practice, due to normalisation over $|V|$

$$P(w_k|w_j) = \frac{\exp(c_{w_k} \cdot v_{w_j})}{\sum_{w' \in V} \exp(c_{w'} \cdot v_{w_j})}$$

Sum up all vocabulary

- Reduce problem to binary classification, distinguish real context words from non-context words aka "negative samples"
 - words drawn randomly from V

29

Negative Sampling

window=2

... lemon, a [tablespoon of apricot jam, a] pinch ...

c1	c2	t	c3	c4
----	----	---	----	----

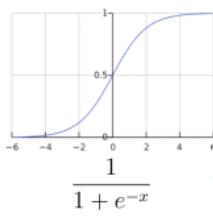
positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

randomly sample



$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

maximise similarity between target word and real context words

$$P(-|t, c) = 1 - \frac{1}{1 + e^{-t \cdot c}}$$

minimise similarity between target word and non-context words

30

Skip-gram Loss

$$L(\theta) = \sum_{(t,c) \in +} \log P(+|t, c) + \sum_{(t,c) \in -} \log P(-|t, c)$$

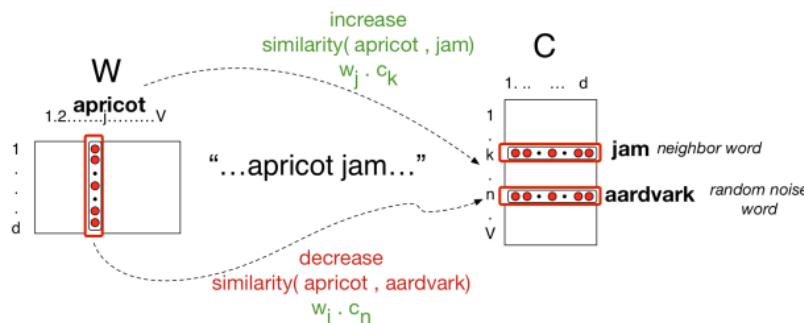
- In practice, use k negative examples

$$L(\theta) = \log P(+|t, c) + \sum_{i=1}^k \log P(-|t, n_i)$$

31

Training Illustration

- Iterative process (stochastic gradient descent)
 - each step moves embeddings closer for context words
 - and moves embeddings apart for noise samples



Source: JM3 Ch 6

32

Desiderata

- Unsupervised
 - Raw, unlabelled corpus
- Efficient
 - Negative sampling (avoid softmax over full vocabulary)
 - Scales to very very large corpus
- Useful representation:
 - How do we evaluate word vectors?

33

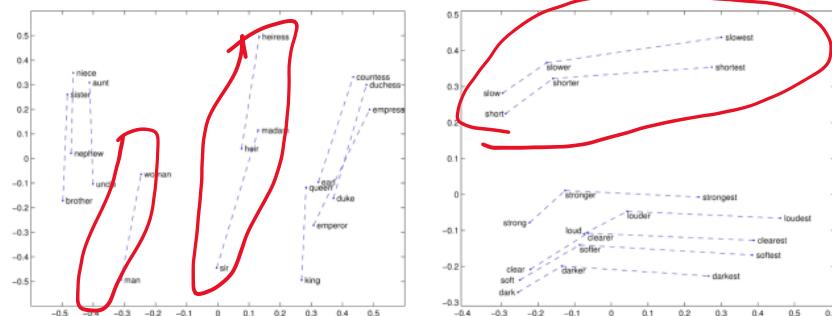
Evaluating Word Vectors

- Lexicon style tasks
 - WordSim-353 are pairs of nouns with judged relatedness
 - SimLex-999 also covers verbs and adjectives
 - TOEFL asks for closest synonym as multiple choice
 - ...
- Test compatibility of word pairs using cosine similarity in vector space

Correlation

34

Embeddings Exhibit Meaningful Geometry



- Word analogy task

- ▶ *Man* is to *King* as *Woman* is to ???
- ▶ $v(\text{Man}) - v(\text{King}) = v(\text{Woman}) - v(\text{???})$
- ▶ $v(\text{??}) = v(\text{Woman}) - v(\text{Man}) + v(\text{King})$

35

Evaluating Word Vectors

- Best evaluation is in other downstream tasks
 - ▶ Use bag-of-word embeddings as a feature representation in a classifier
 - ▶ First layer of most deep learning models is to embed input text; use pre-trained word vectors as embeddings
- Recently contextual word vectors shown to work even better
 - ▶ ELMO & BERT (next lecture!)

36

Pointers to Software

- Word2Vec
 - C implementation of Skip-gram and CBOW
<https://code.google.com/archive/p/word2vec/>
- GenSim
 - Python library with many methods include LSI, topic models and Skip-gram/CBOW
<https://radimrehurek.com/gensim/index.html>
- GLOVE
 - <http://nlp.stanford.edu/projects/glove/>

37

Further Reading

- JM3, Ch 6

38



Contextual Representation

COMP90042

Natural Language Processing

Lecture 11



meaning of words
in context

COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L11

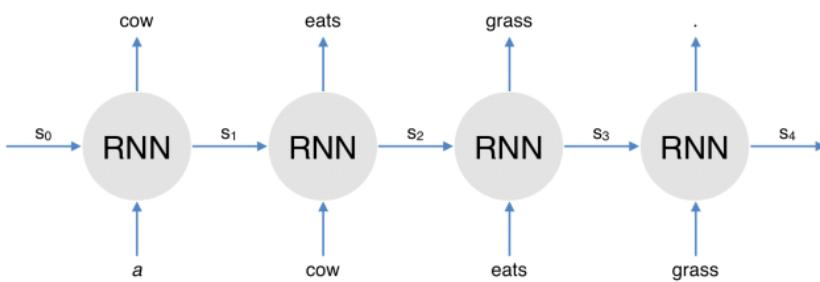
Word Vectors/Embeddings

- Each word type has one representation
 - Word2Vec
- Always the same representation regardless of the context of the word → only have one representation
- Does not capture multiple senses of words
- Contextual representation = representation of words based on context
- Pre-trained contextual representations work *really well* for downstream applications!



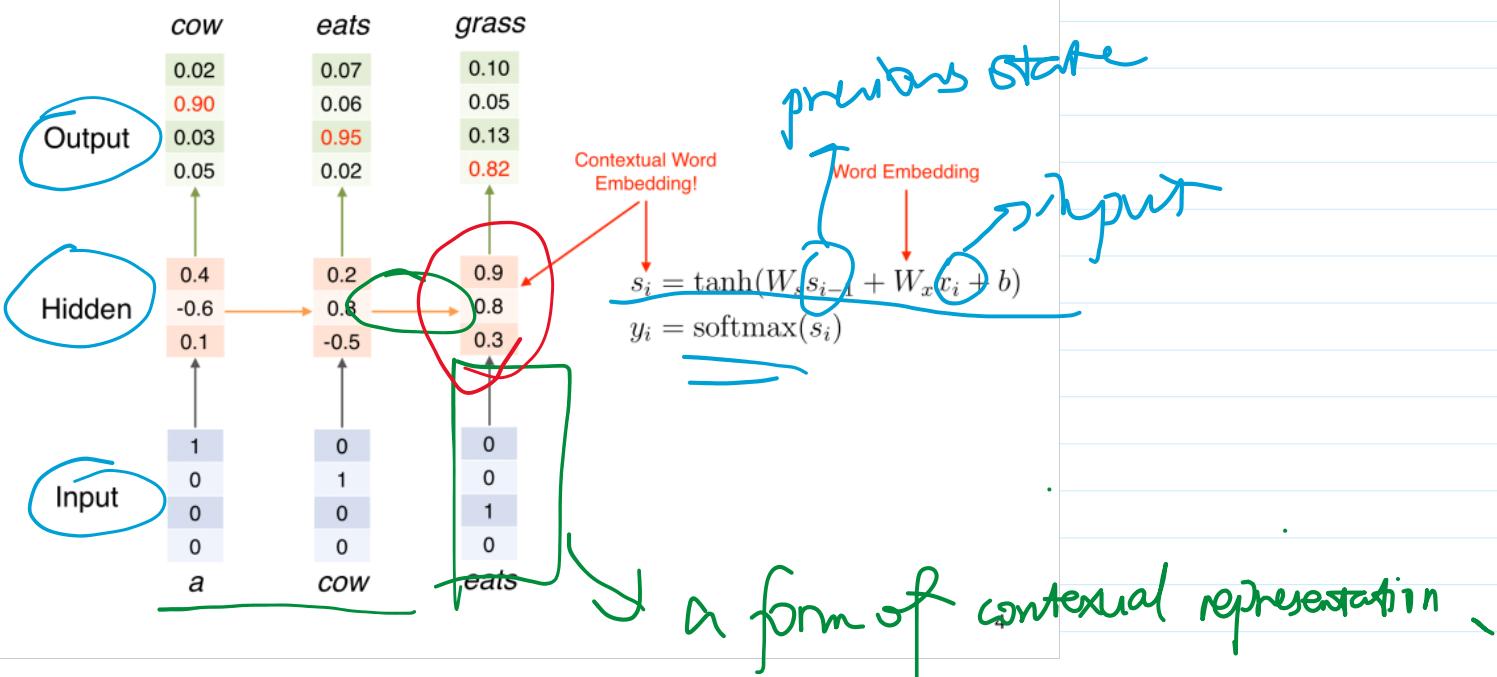
2

RNN Language Model



3

RNN Language Model



Solved?

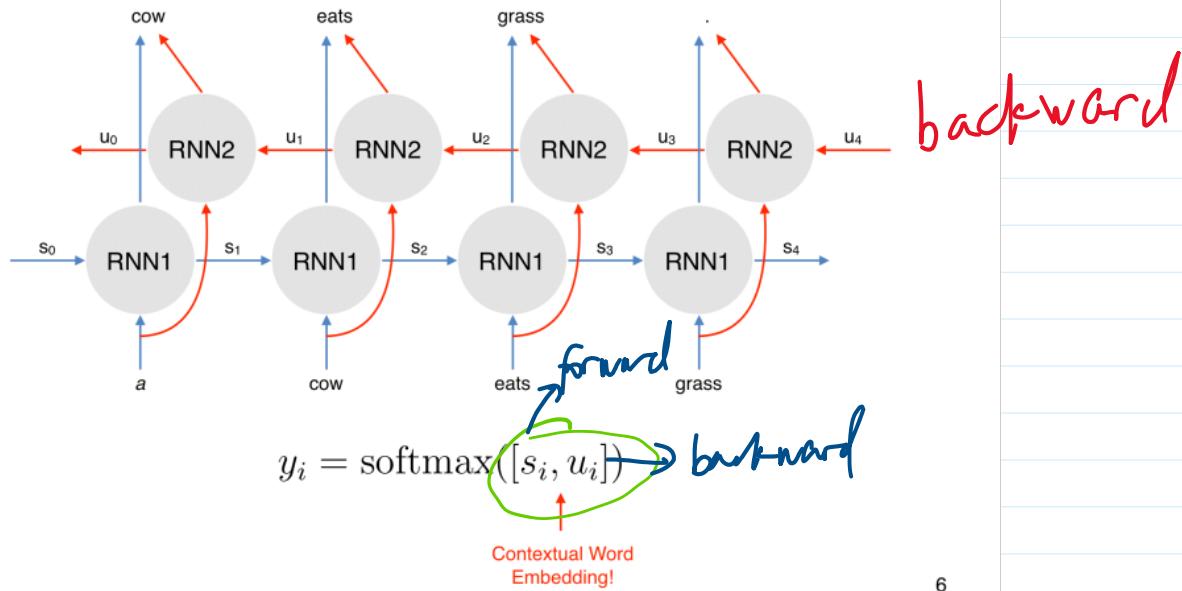
- Almost, but the contextual representation only captures context to the left
- Solution: use a bidirectional RNN instead!**

Solved?

- Almost, but the contextual representation only captures context to the left
- Solution: use a bidirectional RNN instead!**

5

Bidirectional RNN



6

ELMo



7

ELMo: Embeddings from Language Models

- Peters et al. (2018): <https://arxiv.org/abs/1802.05365v2>
- Trains a bidirectional, multi-layer LSTM language model over 1B word corpus
- Combine hidden states from multiple layers of LSTM for downstream tasks
 - Prior studies use only top layer information
- Improves task performance significantly!

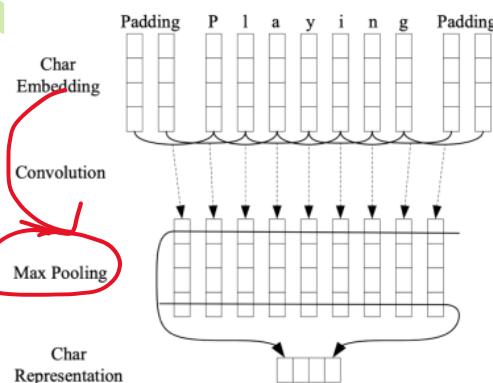
8

ELMo

- Number of LSTM layers = 2
- LSTM hidden dimension = 4096
- Character convolutional networks (CNN) to create word embeddings

► No unknown words

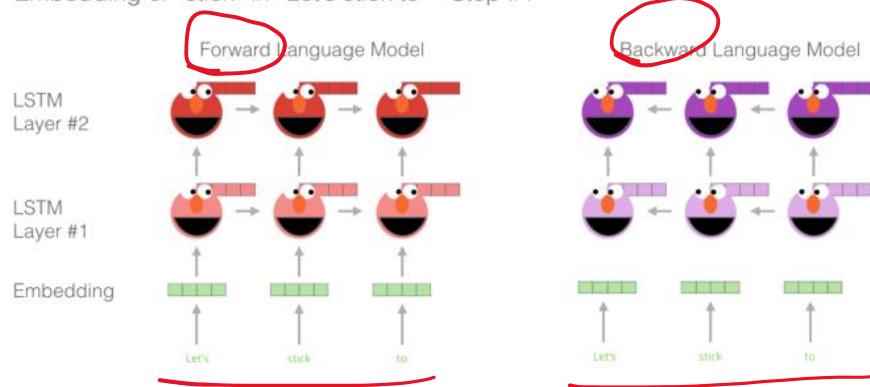
Solve



<https://www.aclweb.org/anthology/P16-1101.pdf> 9

Extracting Contextual Representation

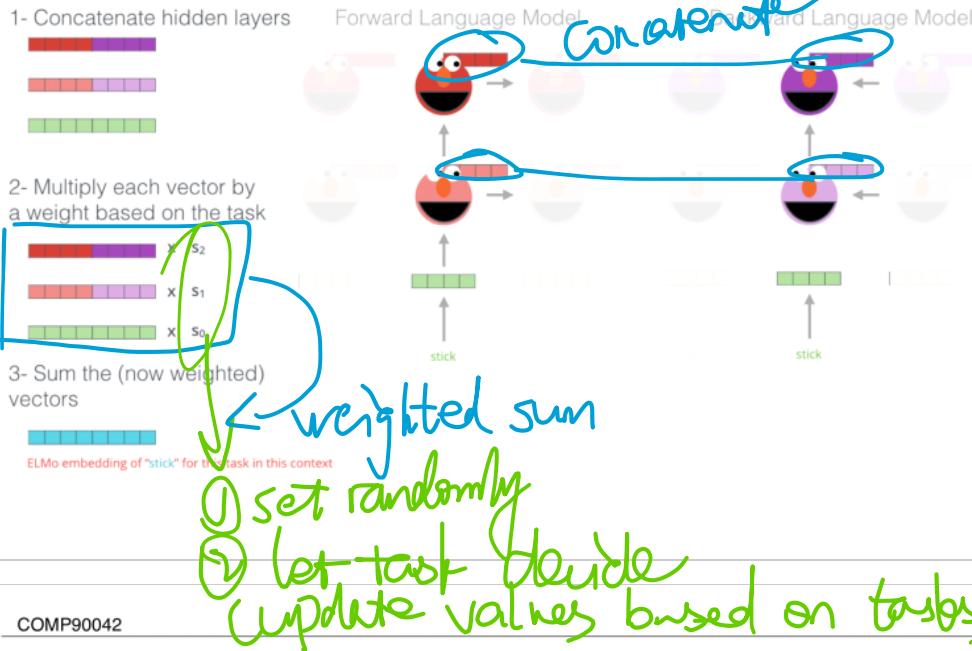
Embedding of "stick" in "Let's stick to" - Step #1



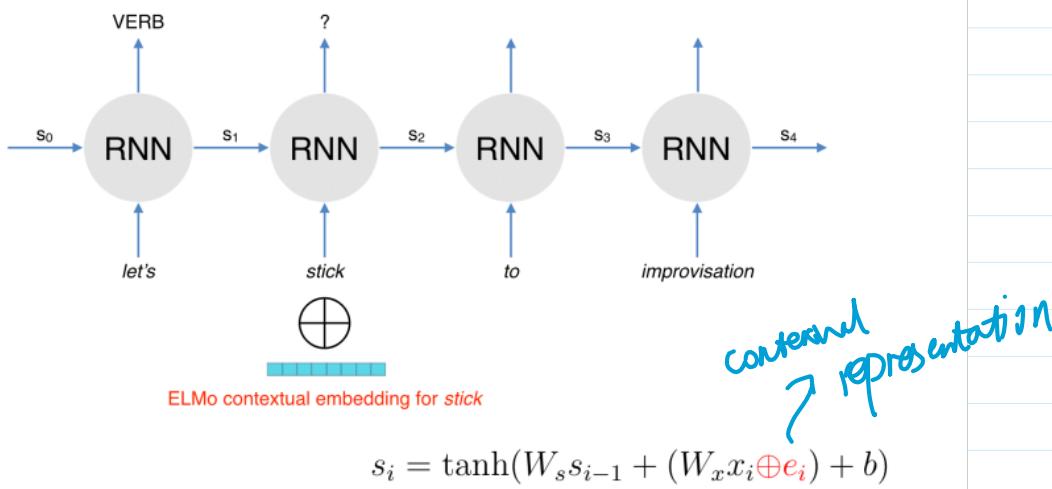
<http://jalammar.github.io/illustrated-bert/> 10

Extracting Contextual Representation

Embedding of "stick" in "Let's stick to" - Step #2



Downstream Task: POS Tagging



How Good is ELMo?

TASK	PREVIOUS SOTA	OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8 4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17 0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6 3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4 3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10 2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5 3.3 / 6.8%

- SQuAD: QA
- SNLI: textual entailment
- SRL: semantic role labelling
- Coref: coreference resolution
- NER: named entity recognition
- SST-5: sentiment analysis

13

Other Findings

- Lower layer representation = captures syntax
 - ▶ good for POS tagging, NER
- Higher layer representation = captures semantics
 - ▶ good for QA, textual entailment, sentiment analysis

14

Contextual vs. Non-contextual

Source	Nearest Neighbors
GloVe play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM Chico Ruiz made a spectacular <u>play</u> on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent <u>play</u> .
	{...} they were actors who had been handed fat roles in a successful <u>play</u> , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to “play” using GloVe and the context embeddings from a biLM.

15

Disadvantages of RNNs

(last word?)

- Sequential processing: difficult to scale to very large corpus or models
- RNN language models run left to right (captures only one side of context)
 - ▶ Produces well-formed sentence probability
- Bidirectional RNNs help, but they only capture surface bidirectional representations

16



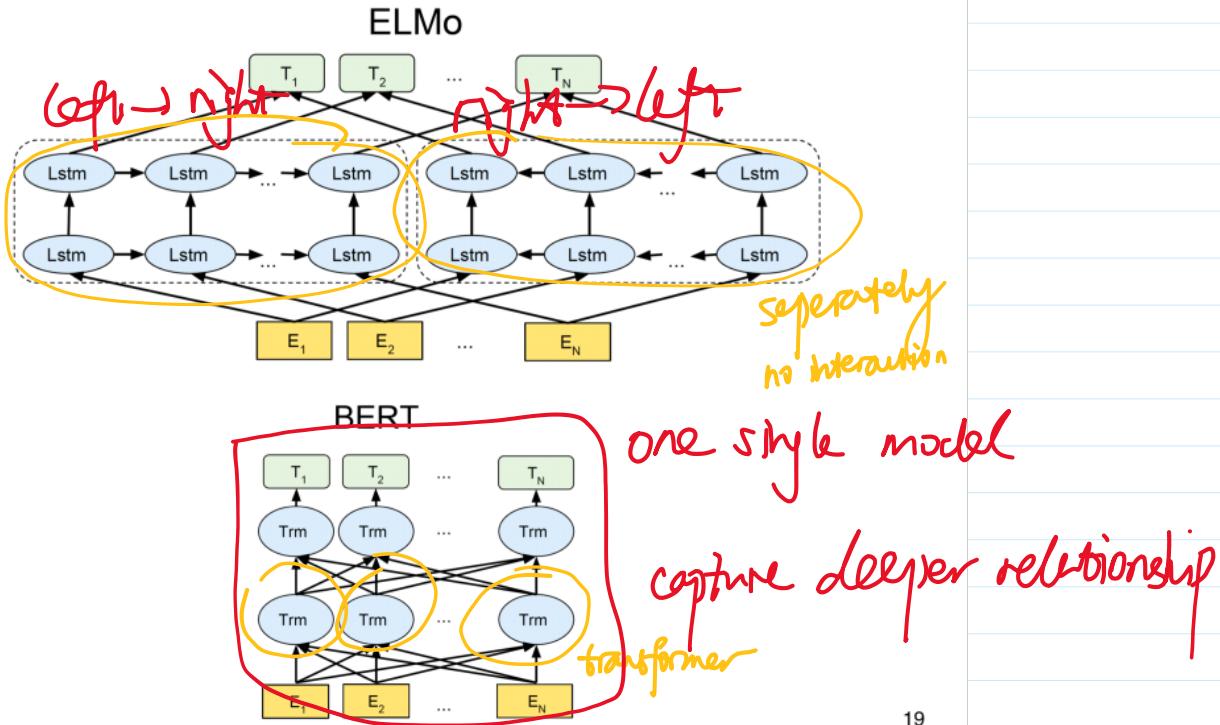
BERT

17

BERT: Bidirectional Encoder Representations from Transformers

- Devlin et al. (2019): <https://arxiv.org/abs/1810.04805>
- Uses self-attention networks (aka **Transformers**) to capture dependencies between words
 - ▶ No sequential processing
- **Masked language model** objective to capture deep bidirectional representations
- Loses the ability to generate language
- Not an issue if the goal is to learn contextual representations

18



Objective 1: Masked Language Model

- ‘Mask’ out k% of tokens at random
- Objective: predict the masked words

lecture



and



Today we have a [MASK] on contextual representations [MASK] it's interesting

Objective 2: Next Sentence Prediction

- Learn relationships between sentences
- Predicts whether sentence B follows sentence A
- Useful pre-training objective for downstream applications that analyse sentence pairs (e.g. textual entailment)

Sentence A: Today we have a lecture on NLP.

Sentence B: It is an interesting lecture.

Label: IsNextSentence

Sentence A: Today we have a lecture on NLP.

Sentence B: Polar bears are white.

Label: NotNextSentence

21

Training/Model Details

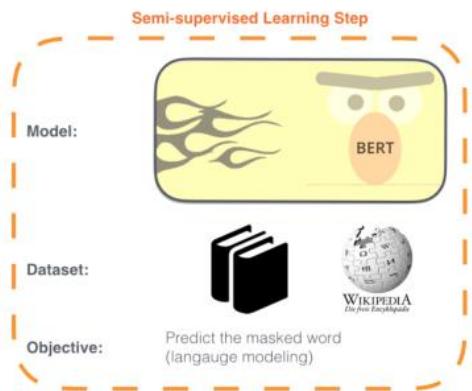
- WordPiece (subword) Tokenisation
- Multiple layers of transformers to learn contextual representations
- Train models trained on Wikipedia+BookCorpus
- Training takes multiple GPUs over several days

22

Fine-Tuning for BERT

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

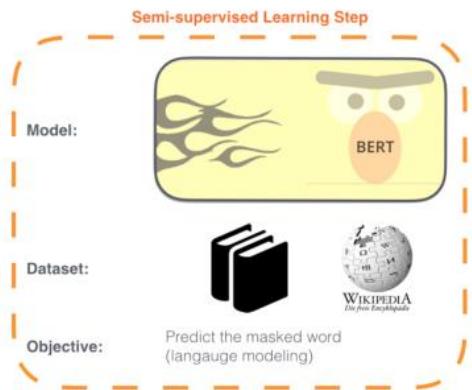


<http://jalammar.github.io/illustrated-bert/> 23

Fine-Tuning for BERT

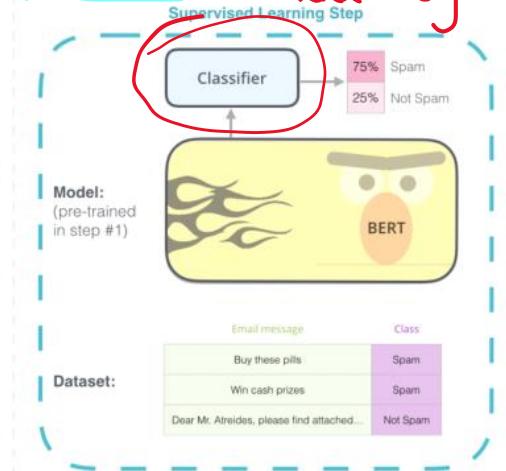
1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



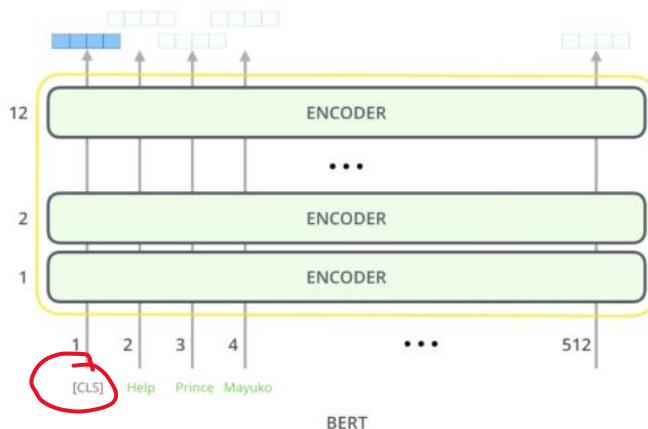
2 - Supervised training on a specific task with a labeled dataset.

add a layer



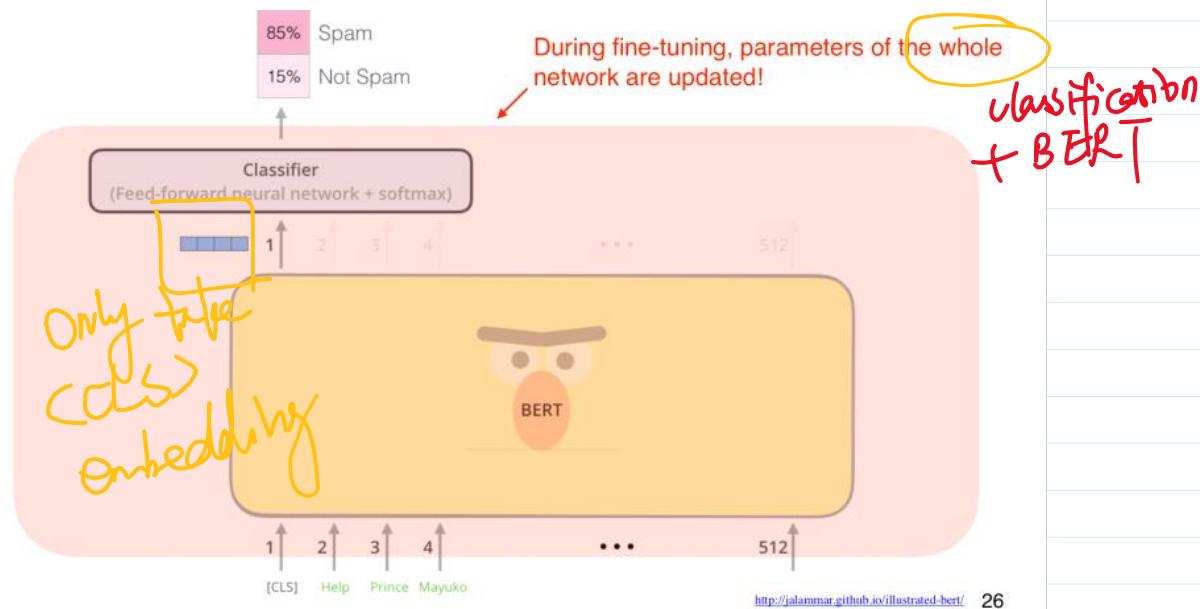
<http://jalammar.github.io/illustrated-bert/> 24

Example: Spam Detection



<http://jalammar.github.io/illustrated-bert/> 25

Example: Spam Detection

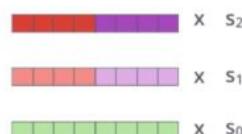


<http://jalammar.github.io/illustrated-bert/> 26

BERT vs. ELMo

- ELMo provides only the contextual representations
- Downstream applications has their own network architecture
- ELMo parameters are fixed when applied to downstream applications
 - ▶ Only the weights to combine states from different LSTM layers are learned

2- Multiply each vector by a weight based on the task



27

BERT vs. ELMo

- BERT adds a classification layer for downstream tasks
 - ▶ No task-specific model needed
- BERT updates all parameters during fine-tuning

28

How Good is BERT?

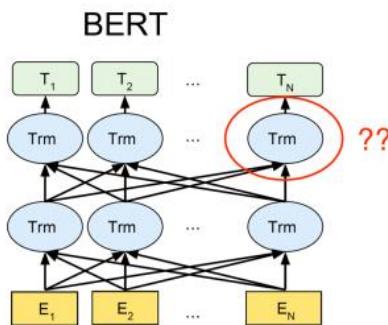
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

- MNLI, RTE: textual entailment
- QQP, STS-B, MRPC: sentence similarity
- QNLP: answerability prediction
- SST: sentiment analysis
- COLA: sentence acceptability prediction

29

Transformers

- What are transformers, and how do they work?

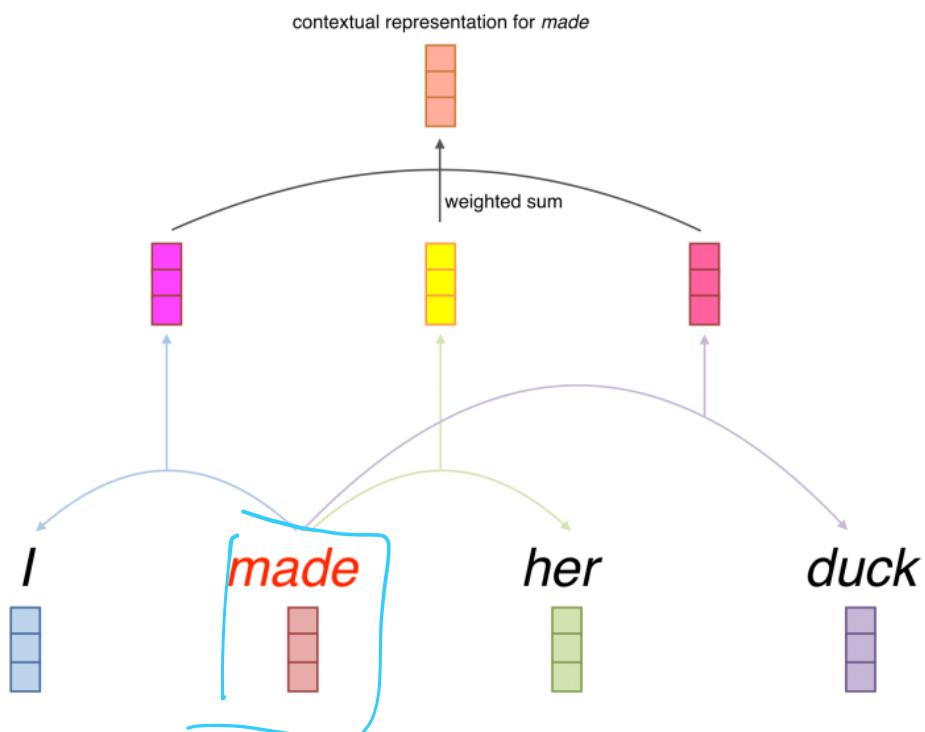


30

Attention is All You Need

- Vaswani et al. (2017): <https://arxiv.org/abs/1706.03762>
- Use attention instead of using RNNs (or CNNs) to capture dependencies between words

31



32

Self-Attention: Implementation

- Input:
 - query q (e.g. *made*)
 - key k and value v (e.g. *her*)
- Query, key and value are all **vectors**
 - linear projections from embeddings

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} \times v_i$$

soft max

$$c_{\text{made}} = 0.1v_{\text{I}} + 0.6v_{\text{her}} + 0.3v_{\text{duck}}$$

33

Self-Attention: Implementation

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} \times v_i$$

- Multiple queries, stack them in a matrix

$$A(Q, K, V) = \text{softmax}(QK^\top)V$$

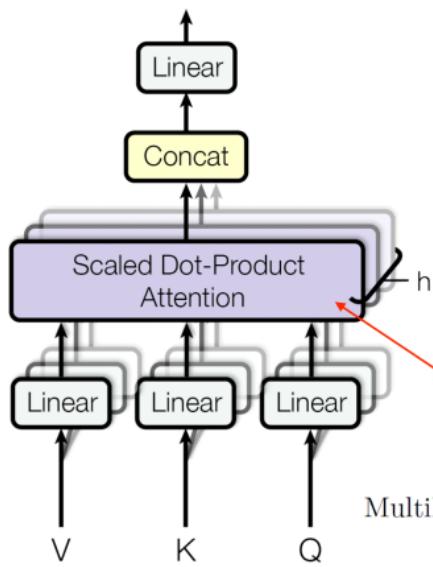
- Uses scaled dot-product to prevent values from growing too large

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

dimension of query and key vectors

34

Multi-Head Attention



- Only one attention for each word pair
- Uses multi-head attention to allow multiple interactions

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

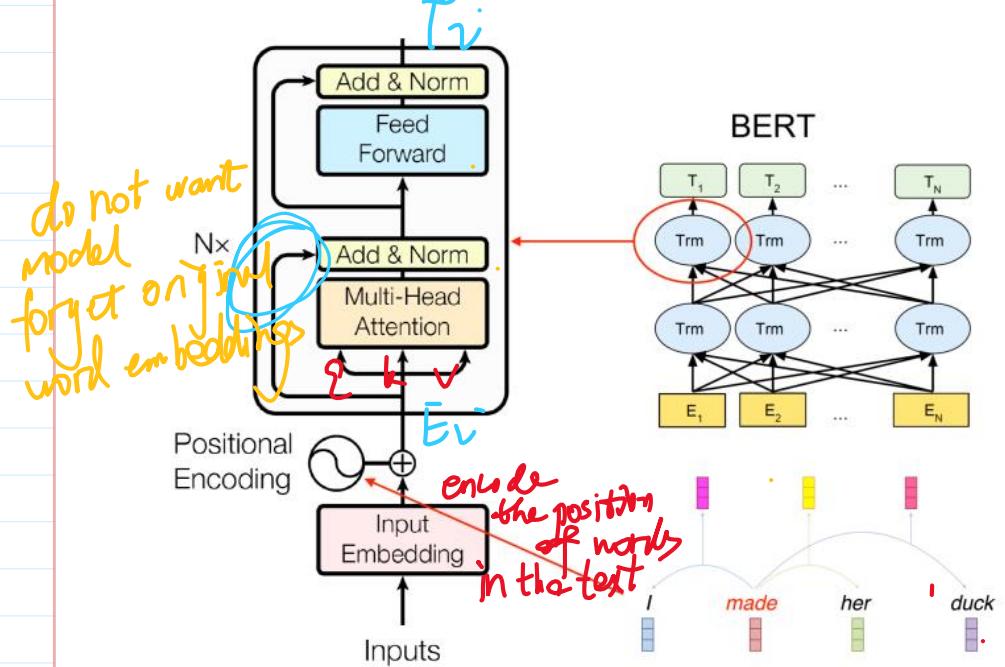
multiple times
concatenate

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

35

Transformer Block



A Final Word

- Contextual representations are very useful
- Pre-trained on very very large corpus
 - Builds up some knowledge about language
 - Uses unsupervised objectives
- When we use them for downstream tasks, we are no longer starting from “scratch”

37

Further Reading

- ELMo: <https://arxiv.org/abs/1802.05365v2>
- BERT: <https://arxiv.org/abs/1810.04805>
- Transformer: <http://nlp.seas.harvard.edu/2018/04/03/attention.html>

38

Discourse

organize sentences
in documents

COMP90042

Natural Language Processing

Lecture 12



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L12

Discourse

- Most tasks/models we learned operate at word or sentence level:
 - ▶ POS tagging → word/sentence
 - ▶ Language models → sentence
 - ▶ Lexical/distributional semantics
- But NLP often deals with documents
- Discourse: understanding how sentences relate to each other in a document

2

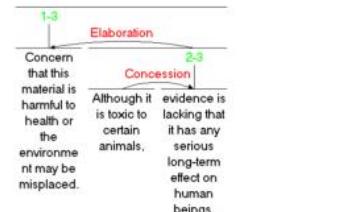
Three Key Discourse Tasks

- Discourse segmentation



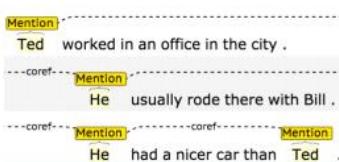
partition document
↓
individual chapters

- Discourse parsing



free structure

- Anaphora resolution



ambiguous pronouns

3

Discourse Segmentation

4

Discourse Segmentation

- A document can be viewed as a sequence of segments
- A segment: a span of cohesive text
- Cohesion:
 - ▶ organised around a particular topic or function
 - Wikipedia biographies: early years, major events, impact on others
 - Scientific articles: introduction, related work, experiments

5

Unsupervised Approaches

- TextTiling algorithm: looking for points of low lexical cohesion between sentences
- For each sentence gap:
 - ▶ Create two BOW vectors consisting of words from k sentences on either side of gap
 - ▶ Use cosine to get a similarity score (sim) for two vectors
 - ▶ For gap i , calculate a depth score, insert boundaries when depth is greater than some threshold t
$$\text{depth}(\text{gap}_i) = (\text{sim}_{i-1} - \text{sim}_i) + (\text{sim}_{i+1} - \text{sim}_i)$$

6

Text Tiling Example ($k=1$, $t=0.9$)

$d=0.7-0.9=-0.2$	sim: 0.9	He walked 15 minutes to the tram stop.
$d=(0.9-0.7)+(0.1-0.7)=-0.4$	sim: 0.7	Then he waited for another 20 minutes, but the tram didn't come.
$d=(0.7-0.1)+(0.5-0.1)=1.0$	sim: 0.1	The tram drivers were on strike that morning.
$d=(0.1-0.5)+(0.8-0.5)=-0.1$	sim: 0.5	So he walked home and got his bike out of the garage.
$d=(0.1-0.5)+(0.8-0.5)=-0.6$	sim: 0.8	He started riding but quickly discovered he had a flat tire
$d=0.8-0.5=0.3$	sim: 0.5	He walked his bike back home.
		He looked around but his wife had cleaned the garage and he couldn't find the bike pump.

$$\text{depth}(\text{gap}_i) = (\text{sim}_{i-1} - \text{sim}_i) + (\text{sim}_{i+1} - \text{sim}_i)$$

Supervised Approaches

- Get labelled data from easy sources
 - Scientific publications
 - Wikipedia articles

Abstract

Language models are typically applied at the sentence level, without access to the broader document context. We present a neural language model that incorporates document context in the form of a topic model-like architecture, thus providing a succinct representation of the broader document context outside of the current sentence. Experiments over a range of datasets demonstrate that our model outperforms a pure sentence-based model in terms of language model perplexity, and leads to topics that are potentially more coherent than those produced by a standard LDA topic model. Our model also has the ability to generate related sentences for a topic, providing another way to interpret topics.

1 Introduction

Topic models provide a powerful tool for extracting the macro-level content structure of a document collection in the form of latent topics (usually in the form of multinomial distributions over terms), with a plethora of applications in NLP (Hall et al., 2008; Newman et al., 2010; Wang and McCallum, 2006). A myriad of variants of the classical LDA method (Blei et al., 2003) have been proposed, including recent work on neural topic models (Cao et al., 2015; Wan et al., 2012; Larochelle and Lai, 2012; Hinton and Salakhutdinov, 2009).

Separately, language models have long been a foundational component of any NLP task involving generation or textual normalisation of a noisy input (including speech, OCR and the processing of social media text). The primary purpose of a language model is to predict the probability of a

span of text, traditionally at the sentence level, under the assumption that sentences are independent of one another, although recent work has started using broader local context such as the preceding sentences (Wang and Cho, 2016; Ji et al., 2016).

In this paper, we combine the benefits of a topic model and language model in proposing a topic-driven language model, whereby we jointly learn topics and word sequence information. This allows us to both sensitise the predictions of the language model to the larger document narrative using topics, and to generate topics which are better sensitised to local context and are hence more coherent and interpretable.

Our model has two components, a language model and a topic model. We implement both components using neural networks, and train them jointly by treating each component as a sub-task in a multi-task learning setting. We show that our model is superior to other language models that leverage additional context, and that the generated topics are potentially more coherent than LDA topics. The architecture of the model provides an extra dimensionality of topic interpretability, in supporting the generation of sentences from a topic (or mix of topics). It is also highly flexible, in its ability to be supervised and incorporate side information, which we show to further improve language model performance. An open source implementation of our model is available at: <https://github.com/jhLau/topic-driven-language-model>

2 Related Work

Griffiths et al. (2004) propose a model that learns topics and word dependencies using a Bayesian framework. Word generation is driven by either LDA or an HMM. For LDA, a word is generated based on a sampled topic in the document. For the

Supervised Discourse Segmenter

- Apply a binary classifier to identify boundaries ✓ ✗
- Or use sequential classifiers HMM / RNN
- Potentially include classification of section types (introduction, conclusion, etc.)
- Integrate a wider range of features, including
 - distributional semantics
 - discourse markers (*therefore, and, etc*)

9

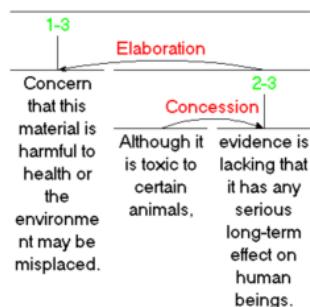
Discourse Parsing

organize the discourse units

10

Discourse Parsing

- Identify discourse units, and the relations that hold between them
- Rhetorical Structure Theory (RST), is a framework to do hierarchical analysis of discourse structure in documents



11

RST

- Basic element: elementary discourse units (EDUs)
 - Typically clauses of a sentence
 - EDUs do not cross sentence boundary
 - [It does have beautiful scenery,] [some of the best since Lord of the Rings.]*
- RST relations between discourse units:
 - conjunction, justify, concession, elaboration, etc*
 - [It does have beautiful scenery,]*
↑(elaboration)
[some of the best since Lord of the Rings.]

12

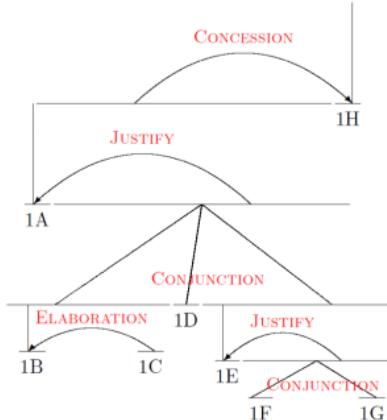
Nucleus vs. Satellite

- Within a discourse relation, one argument is the **nucleus** (the primary argument) → **main point**
- The supporting argument is the **satellite**
 - [*It does have beautiful scenery.*]_{nucleus}
↑ (elaboration)
[*some of the best since Lord of the Rings.*]_{satellite}
- Some relations are equal (e.g. conjunction), and so both arguments are **nuclei**
 - [*He was a likable chap.*]_{nucleus}
↑ (conjunction)
[*and I hated to see him die.*]_{nucleus}

13

RST Tree

- An RST relation combines two or more DUs into composite DUs
- Process of combining DUs is repeated to create an RST tree



[*It could have been a great movie.*]^{1A} [*It does have beautiful scenery.*]^{1B} [*some of the best since Lord of the Rings.*]^{1C} [*The acting is well done.*]^{1D} [*and I really liked the son of the leader of the Samurai.*]^{1E} [*He was a likable chap.*]^{1F} [*and I hated to see him die.*]^{1G} [*But, other than all that, this movie is nothing more than hidden rip-offs.*]^{1H}

merging discourse units iteratively

14

Parsing Using Discourse Markers

- Some discourse markers (cue phrases) explicitly indicate relations
 - Some examples: *although, but, for example, in other words, so, because, in conclusion,...*
- Can be used to build a simple rule-based parser
- However
 - Many relations are not marked by discourse marker at all
 - Many important discourse markers (e.g. *and*) ambiguous
 - Sometimes not a discourse marker
 - Can signal multiple relations

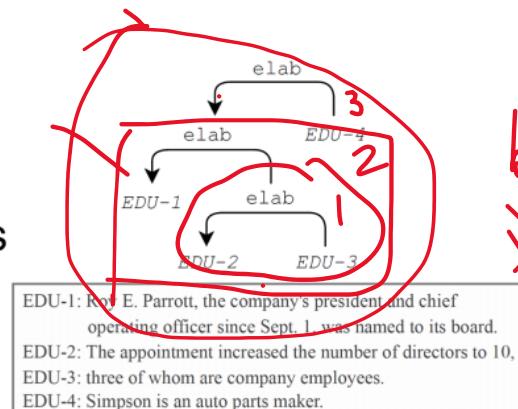
conjunction, justification

15

Parsing Using Machine Learning

- RST Discourse Treebank *data*
 - 300+ documents annotated with RST trees
- Basic idea:
 - Segment document into EDUs
 - Combine adjacent DUs into composite DUs iteratively to create the full RST tree

*classifier
another classifier*



*bottom-up
iterative
merging*

16

Parsing Using Machine Learning

- Transition-based parsing (lecture 16):

- Bottom-up → best-possible
- Greedy, uses shift-reduce algorithm

- CYK/chart parsing algorithm (lecture 14)

- Bottom-up → global optimal
- Global, but some constraints prevent CYK from finding globally optimal tree for discourse parsing

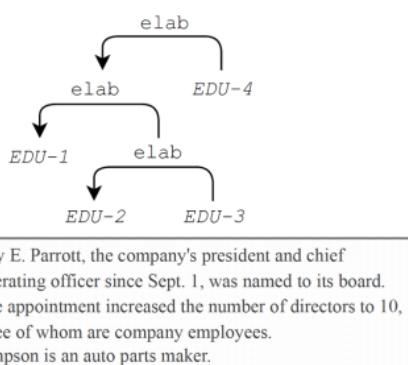
17

Parsing Using Machine Learning

- Top-down parsing

- Sequence labelling problem
- BERT

find cuts



- ②
- ③
- ①

divide and conquer
approach
→ top to bottom

18

Discourse Parsing Features

- Bag of words → words are important
- Discourse markers

- Bag of words → what we import
- Discourse markers
- Starting/ending n-grams start/end sentences with some sets of words → do the same when changing discourse
- Location in the text ↳ different behavior in discourse shifting
- Syntax features
- Lexical and distributional similarities

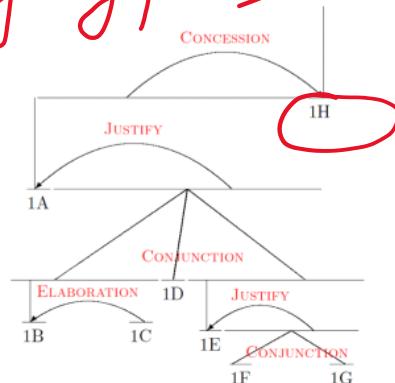
19

COMP90042

L12

Why Discourse Parsing?

- Summarisation → identify key points
- Sentiment analysis
- Argumentation
- Authorship attribution
- Essay scoring



[It could have been a great movie]^{1A} [It does have beautiful scenery.]^{1B} [some of the best since Lord of the Rings.]^{1C} [The acting is well done.]^{1D} [and I really liked the son of the leader of the Samurai.]^{1E} [He was a likable chap.]^{1F} [and I hated to see him die.]^{1G} [But, other than all that, this movie is nothing more than hidden rip-offs.]^{1H}

20

Anaphora Resolution

21

Anaphors

- **Anaphor**: linguistic expressions that refer back to earlier elements in the text
- Anaphors have a **antecedent** in the discourse, often but not always a noun phrase
 - ▶ Yesterday, Ted was late for work. It all started when his car wouldn't start.
- Pronouns are the most common anaphor
- But there are various others
 - ▶ Demonstratives (that problem)

22

Antecedent Restrictions

- Pronouns must agree in *number* with their antecedents
 - ▶ *His coworkers* were leaving for lunch when *Ted* arrived. *They* invited him, but he said no.
- Pronouns must agree in *gender* with their antecedents
 - ▶ *Sue* was leaving for lunch when *Ted* arrived. *She* invited him, but he said no.
- Pronouns whose antecedents are the subject of the same syntactic clause must be *reflexive* (...self)
 - ▶ *Ted* was angry at *him*. [*him* ≠ *Ted*]
 - ▶ *Ted* was angry at *himself*. [*himself* = *Ted*]

23

Antecedent Preferences

- The antecedents of pronouns should be recent
 - ▶ *He waited for another 20 minutes, but the tram didn't come. So he walked home and got his bike out of the garage. He started riding it to work.*
- The antecedent should be salient, as determined by grammatical position
 - ▶ *Subject > object > argument of preposition*
 - ▶ *Ted usually rode to work with Bill. He was never late.*

24

Entities and Reference

- | | |
|--|---|
| (16.1) a. John went to his favorite music store to buy a piano.
b. He had frequented the store for many years.
c. He was excited that he could finally buy a piano.
d. He arrived just as the store was closing for the day | (16.2) a. John went to his favorite music store to buy a piano.
b. It was a store John had frequented for many years.
c. He was excited that he could finally buy a piano.
d. It was closing just as John arrived. |
|--|---|

- Discourse 16.1 (left) more coherent
- Pronouns all refer to John consistently, the protagonist

25

Centering Theory

- A unified account of relationship between discourse structure and entity reference
- Every utterance in the discourse is characterised by a set of entities, known as centers
- Explains preference of certain entities for ambiguous pronouns

26

For an Utterance U_n

- **Forward-looking centers:** (16.1)
 - ▶ All entities in U_n :
 - $C_f(U_n) = [e_1, e_2, \dots]$
 - ▶ $C_f(16.1a) = [\text{John, music store, piano}]$
 - ▶ Ordered by syntactic prominence: subjects > objects
- **Backward-looking center:**
 - ▶ Highest ranked entity in previous utterance's ($C_b(U_{n-1})$) forward-looking centers that is also in current utterance (U_n)
 - ▶ $C_b(16.b) = [\text{John}]$

27

Centering Algorithm

- When resolving entity for anaphora resolution, choose the entity such that the top forward-looking center matches with the backward-looking center

- (16.1)
- a. John went to his favorite music store to buy a piano.
 - b. He had frequented the store for many years.
 - c. He was excited that he could finally buy a piano.
 - d. He arrived just as the store was closing for the day

- (16.2)
- a. John went to his favorite music store to buy a piano.
 - b. It was a store John had frequented for many years.
 - c. He was excited that he could finally buy a piano.
 - d. It was closing just as John arrived.

28

The Centering Algorithm

1. John saw a Ford in the dealership

$$\begin{aligned} C_f(U_1) &= [\text{John}, \text{Ford}, \text{dealership}] \\ C_b(U_1) &= \text{None} \end{aligned}$$

2. He showed it to Bob

$$\begin{aligned} C_f(U_2) &= [\text{John}, \text{Ford}, \text{Bob}] \\ C_b(U_2) &= \text{John} \end{aligned}$$

3. He bought it

If he = John:
 $C_f(U_3) = [\text{John}, \text{Ford}]$
 $C_b(U_3) = \text{John}$

If he = Bob:
 $C_f(U_3) = [\text{Bob}, \text{Ford}]$
 $C_b(U_3) = \text{Ford}$

top forward-looking center = backward-looking center

29

Supervised Anaphor Resolution

- Build a binary classifier for anaphor/antecedent pairs
- Convert restrictions and preferences into features
 - Binary features for number/gender compatibility
 - Position of antecedent in text
 - Include features about type of antecedent
- With enough data, can approximate the centering algorithm
- But also easy to include features which indicate tendencies, rather than rules
 - Like repetition, parallelism

30

Anaphora Resolution Tools

- Stanford CoreNLP includes pronoun coreference models
 - rule-based system does very well
 - considerably faster than learned models

Mention Ted worked in an office in the city .
 coref- Mention He usually rode there with Bill .
 coref- Mention He had a nicer car than Ted .

SYSTEM	LANGUAGE	PREPROCESSING TIME	COREF TIME	TOTAL TIME	F1 SCORE
Deterministic	English	3.87s	0.11s	3.98s	49.5
Statistical	English	0.48s	1.23s	1.71s	56.2
Neural	English	3.22s	4.96s	8.18s	60.0
Deterministic	Chinese	0.39s	0.16s	0.55s	47.5
Neural	Chinese	0.42s	7.02s	7.44s	53.9

Source: <https://stanfordnlp.github.io/CoreNLP/coref.html>
 Evaluated on CoNLL 2012 task.

31

Motivation for Anaphor Resolution

- Essential for deep semantic analysis
 - Very useful for QA, e.g., reading comprehension

Ted's car broke down. So he went over to Bill's house to borrow his car. Bill said that was fine.

Whose car is borrowed?

32

A Final Word

- For many tasks, it is important to consider context larger than sentences
- Traditionally many popular NLP applications has been sentence-focused (e.g. machine translation), but that is beginning to change...

33

Further Reading

- E18, Ch 16

34



I13-formal
-languag...

Formal Language Theory & Finite State Automata

COMP90042

Natural Language Processing

Lecture 13



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L13

What is a Language?

- Methods to process sequence of symbols:
 - ▶ Language Model
 - ▶ Hidden Markov Model
 - ▶ Recurrent Neural Networks
- Nothing is fundamentally linguistic about these models

2

Formal Language Theory

3

Motivation

- Formal language theory studies **classes** of languages and their computational properties
 - ▶ Regular language (this lecture)
 - ▶ Context free language (next lecture)
 - Main goal is to solve the **membership problem**
 - ▶ Whether a string is in a language or not
 - How? By defining its **grammar**

4

Examples

- Binary strings that start with 0 and end with 1
 - ▶ { 01, 001, 011, 0001, ... } ✓
 - ▶ { 1, 0, 00, 11, 100, ... } ✗
- Even-length sequences from alphabet {a, b}
 - ▶ { aa, ab, ba, bb, aaaa, ... } ✓
 - ▶ { aaa, aba, bbb, ... } ✗
- English sentences that start with wh-word and end in ?
 - ▶ { what ?, where my pants ?, ... } ✓

5

Beyond Membership Problem...

- **Membership**
 - ▶ Is the string part of the language? Y/N
- **Scoring**
 - ▶ Graded membership
 - ▶ “How acceptable is a string?” (language models!)
- **Transduction**
 - ▶ “Translate” one string into another (stemming!)

6

Overview

- Regular languages
- Finite state acceptors & transducers
- Modelling word morphology

7

Regular Languages

- Regular language: the simplest class of languages
- Any **regular expression** is a regular language
- The regular expression itself is the grammar
 - ▶ Evaluates whether a string is part of the language

8

Regular Languages

- Formally, a regular expression includes the following operations:

- Symbol drawn from alphabet Σ
- Empty string, ϵ
- Concatenation of two regular expressions, RS
- Alternation of two regular expressions, $R|S$
- Kleene star for 0 or more repeats, R^*
- Parenthesis () to define scope of operations

9

Examples of Regular Languages

- Binary strings that start with 0 and end with 1
 - $0(0|1)^*1$
- Even-length sequences from alphabet { a, b }
 - $((aa)|(ab)|(ba)|(bb))^*$
- English sentences that start with *wh*-word and end in ?
 - $((what)|(where)|(why)|(which)|(whose)|(whom)) \Sigma^* ?$

10

Properties of Regular Languages

- Closure: if we take regular languages L1 and L2 and merge them, is the resulting language regular?
- RLs are closed under the following:
 - ▶ concatenation and union — follows from definition
 - ▶ intersection: strings that are valid in both L1 and L2
 - ▶ negation: strings that are not in L
- Extremely versatile! Can have RLs for different properties of language, and use them together
 - ▶ core algorithms will still apply

11

Finite State Acceptors

- Regular expression defines a regular language
- But it doesn't give an algorithm to check whether a string belongs to the language
- Finite state acceptors (FSA) describes the computation involved for membership checking

12

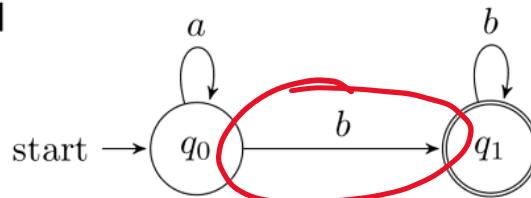
Finite State Acceptors

- FSA consists:
 - alphabet of input symbols, Σ
 - set of states, Q
 - start state, $q_0 \in Q$
 - final states, $F \subseteq Q$
 - transition function
symbol and state → next state
- Accepts strings if there is path from q_0 to a final state with transitions matching each symbol
 - Djisktra's shortest-path algorithm, $O(V \log V + E)$

13

Example FSA

- Input alphabet {a, b}
- States { q_0, q_1 }
- Start, final states $q_0, \{q_1\}$
- Transition function $\{(q_0, a) \rightarrow q_0, (q_0, b) \rightarrow q_1, (q_1, b) \rightarrow q_1\}$
- Note: seeing a in q_1 results in failure
- Accepts a^*bb^*



14

Derivational Morphology

- Use of affixes to change word to another grammatical category
- *grace* → *graceful* → *gracefully*
- *grace* → *disgrace* → *disgracefully*
- *allure* → *alluring* → *alluringly*
- *allure* → **allureful*
- *allure* → **disallure*

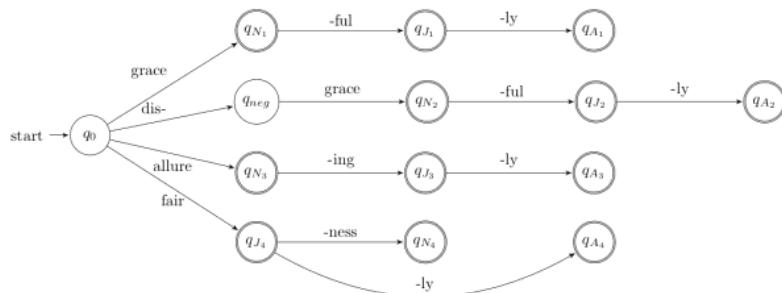
15

FSA for Morphology

- (Fairly) consistent process – can we describe this as a regular language?
 - ▶ want to accept valid forms, and reject invalid ones [flagged with *]
 - ▶ generalise to other words, e.g., nouns that behave like *grace* or *allure*

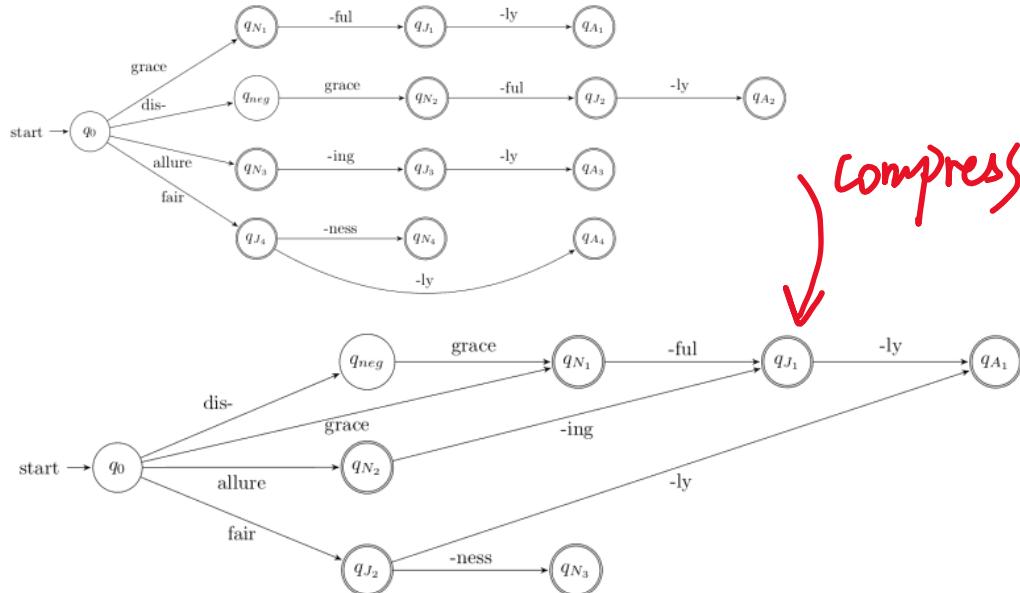
16

FSA for Word Morphology



17

FSA for Word Morphology



18

Weighted FSA

19

Weighted FSA

- Some words are more **possible** than others
 - ▶ *fishful* vs. *disgracelyful*
 - ▶ *musicky* vs. *writey*
- Graded measure of acceptability → weighted FSA adds/changes the following:
 - ▶ set of states Q
 - ▶ alphabet of input symbols Σ
 - ▶ start state weight function, $\lambda: Q \rightarrow \mathbb{R}$
 - ▶ final state weight function, $p: Q \rightarrow \mathbb{R}$
 - ▶ transition function, $\delta: (Q, \Sigma, Q) \rightarrow \mathbb{R}$

} → give the weights

20

WFSA Shortest-Path

- Total score of a path $\pi = t_1, \dots, t_N$ now

$$\lambda(t_0) + \sum_{i=1}^N \delta(t_i) + \rho(t_N)$$

each t is an edge, so more formally using from &/or to states and edge label in score calculation

- Use shortest-path algorithm to find π with min. cost
 - $O(V \log V + E)$, as before

21

N-gram LMs as WFSA

- Recall LM calculates score of string as follows

$$p(w_1, \dots, w_M) \approx \prod_{m=1}^M p_n(w_m | w_{m-1}, \dots, w_{m-n+1}).$$

Unigram language model:

- One state: q_0
- State transition score: $\delta(q_0, \omega, q_0) = \log p_1(\omega)$
- Initial and final state scores = 0
- Path score for w_1, w_2, \dots, w_M :

$$0 + \sum_m^M \delta(q_0, w_m, q_0) + 0 = \sum_m^M \log p_1(w_m).$$

22

Bigram LM

- Bigram sentence probability:

$$P(w_1, w_2, \dots, w_M) = \prod_{i=1}^M P(w_i | w_{i-1}) \text{ (bigram)}$$

- Implemented as WFSA
 - $\Sigma = \text{set of word types}$

- Implemented as WFSA

► $\Sigma = \text{set of word types}$

► $Q = \Sigma$ (no. of states = no. of word types)

$$\delta(q_i, \omega, q_j) = \begin{cases} \log \Pr(w_m = j \mid w_{m-1} = i), & \omega = j \\ -\infty, & \omega \neq j \end{cases}$$

↳

$$\lambda(q_i) = \log \Pr(w_1 = i \mid w_0 = \square)$$

$$\rho(q_i) = \log \Pr(w_{M+1} = \blacksquare \mid w_M = i).$$

↳

23

COMP90042

L13

Finite State Transducer

24

COMP90042

L13

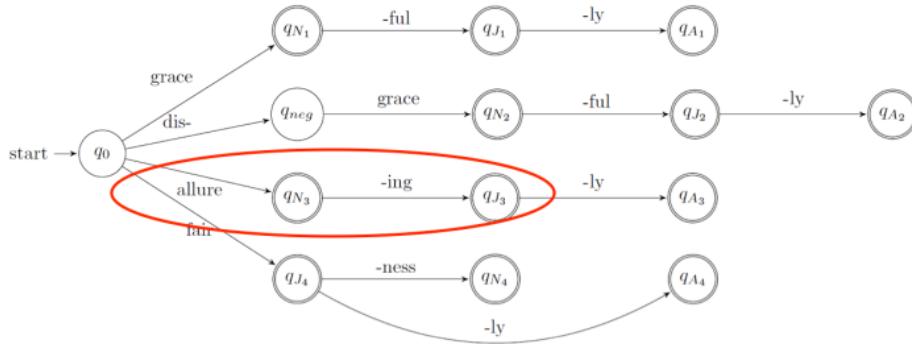
Finite State Transducers (FST)

- Often don't want to just accept or score strings

► want to translate them into another language, correct grammar, parse their structure, etc

25

FSA for Word Morphology



FSA: allure + ing = allureing
 FST: allure + ing = alluring

FSA does not replace anything

26

Finite State Transducers

- FST add string output capability to FSAs
 - ▶ includes an *output alphabet*
 - ▶ and transitions now take *input symbol* and *emit output symbol* (Q, Σ, Δ, Q')
- Can be *weighted* = WFST
 - ▶ Graded scores for transition
- E.g., edit distance as **WFST** which takes one string, and outputs the other
 - ▶ zero cost only if strings are identical

27

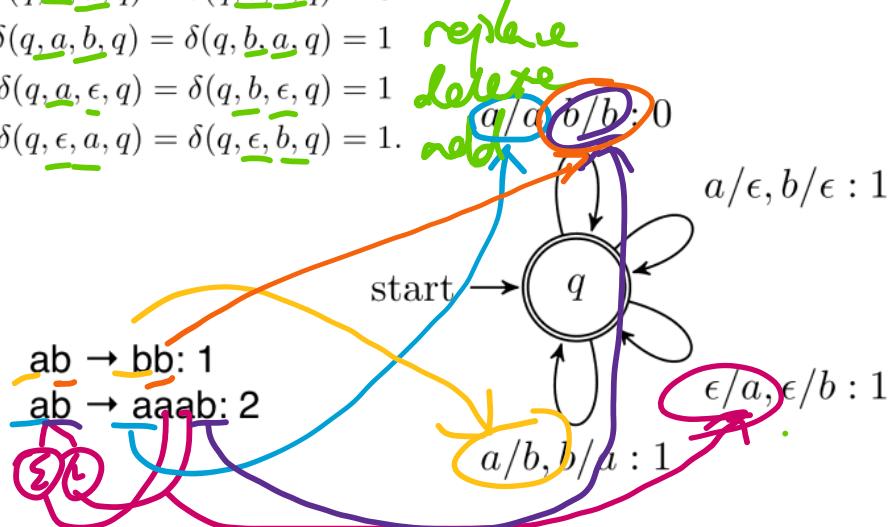
Edit Distance Automata

$$\delta(q, a, a, q) = \delta(q, b, b, q) = 0$$

$$\delta(q, a, b, q) = \delta(q, b, a, q) = 1$$

$$\delta(q, a, \epsilon, q) = \delta(q, b, \epsilon, q) = 1$$

$$\delta(q, \epsilon, a, q) = \delta(q, \epsilon, b, q) = 1.$$



28

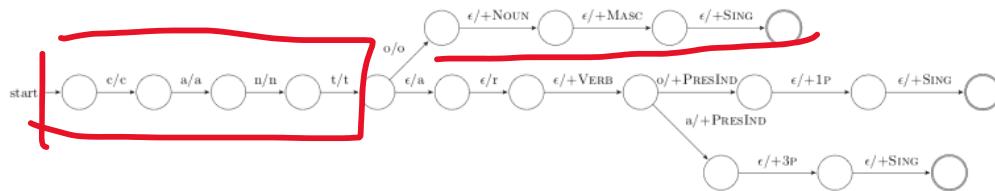
FST for Inflectional Morphology

- Verb inflections in Spanish must match the subject in person & number
- Goal of **morphological analysis**:
 - $canto \rightarrow cantar + \text{VERB} + \text{present} + 1\text{P} + \text{singular}$

	cantar	to sing
1P singular	yo canto	I sing
2P singular	tu cantas	you sing
3P singular	ella canta	she sings
1P plural	nostotros cantamos	we sing
2P plural	vosotros cantáis	you sing
3P plural	ellas cantan	they sing

29

FST for Spanish Inflection



canto → canto+Noun+Masc+Sing

canto → cantar+Verb+PresInd+1P+Sing

30

FST Composition

- Compose two FSTs by taking output of one FST, T1, and giving this as input to FST T2
 - denoted $T_1 \circ T_2$; and results in another FST
 - can also compose FST with FSA, resulting in a FST

31

FST Composition Example

- Allows development of different processes as FSTs:

- -ed added to signal past tense in English
- *cook* → *cooked*, *want* → *wanted*
- But when the word ends with 'e', then we want to avoid producing a spelling with consecutive e's (*bakeed*)
- Solution: build 2 FSTs
- FST T1: *bake+PAST* → *bake+ed*
- FST T2: *bake+ed* → *baked*

32

Is Natural Language Regular?

33

Sometimes...

- Example:

the mouse that ran.

the cat that killed the mouse that ran.

...

the lion that bullied the hyena that bit the dog that chased the cat that killed the mouse that ran

...

- Length is unbounded (*recursive*), but structure is local → can describe with FSA = Regular

- $(\text{Det} \text{ Noun} \text{ Prep} \text{ Verb})^*$

34

Non-Regular Languages

- Arithmetic expressions with balanced parentheses

▶ $(a + (b \times (c / d)))$

▶ Can have arbitrarily many opening parentheses

▶ Need to remember how many open parentheses, to produce the same number of closed parentheses

▶ Can't be done with finite number of states

- $a^n b^n$

35

Center Embedding

- Center embedding of relative clauses
 - The cat loves Mozart
 - The cat **the dog chased** loves Mozart
 - The cat **the dog the rat bit chased** loves Mozart
 - The cat **the dog the rat the elephant admired bit chased** loves Mozart
- Need to remember the *n* subject nouns, to ensure *n* verbs follow (and that they agree etc)
- Requires (at least) **context-free grammar** (next lecture!)

36

Summary

- Concept of a language, and grammar
- Regular languages
- Finite state automata: acceptors, transducers
- Closure properties
- Weighted variants & shortest path inference
- Application to edit distance, morphology

37

Reading

- Reading
 - ▶ E18, Chapter 9.1

14-context
-free-gra...

Context-Free Grammar

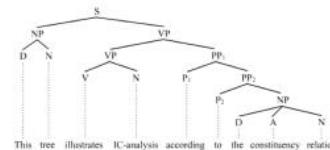
COMP90042

Natural Language Processing

Lecture 14

THE UNIVERSITY OF
MELBOURNE

FUSTERA CRESCAT LAUDE



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L14

Recap

- Center embedding
 - The cat loves Mozart
 - The cat the dog chased loves Mozart
 - The cat the dog the rat bit chased loves Mozart
 - The cat the dog the rat the elephant admired bit chased loves Mozart
- Cannot be captured by regular expressions (S^nV^n)
- **Context-free grammar!**

2

Basics of Context-Free Grammars

- **Symbols**

- **Terminal:** word such as *book*
- **Non-terminal:** syntactic label such as NP or VP
- Convention to use upper and lower-case to distinguish, or else “quotes” for terminals

- **Productions (rules)**

$$W \rightarrow X Y Z$$

- Exactly one **non-terminal** on left-hand side (LHS)
- An ordered list of symbols on right-hand side (RHS)
 - can be **Terminals or Non-terminals**

- **Start symbol:** S

3

Why “Context Free”

$$W \rightarrow X Y Z$$

- Production rule **depends only on the LHS** (and **not** on ancestors, neighbours)
 - **Analogous to Markov chain**
 - **Behaviour at each step depends only on current state**

4

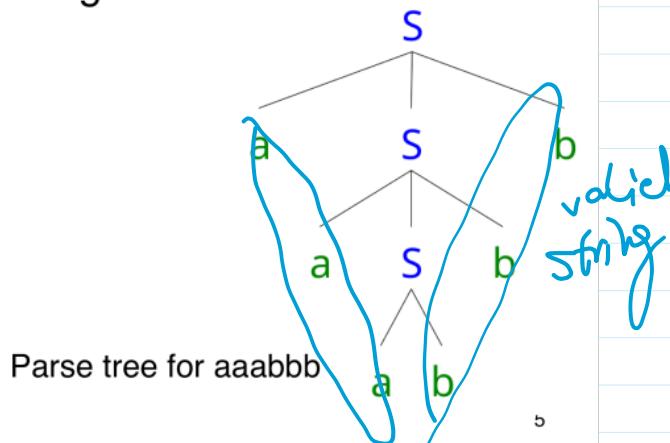
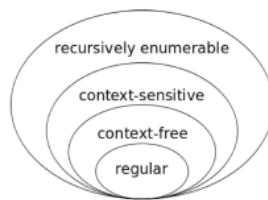
Context-Free vs. Regular

- Context-free languages more general than Regular languages

↳ Allows recursive nesting

- CFG for $a^n b^n$:

↳ $S \rightarrow a S b$
 ↳ $S \rightarrow a b$

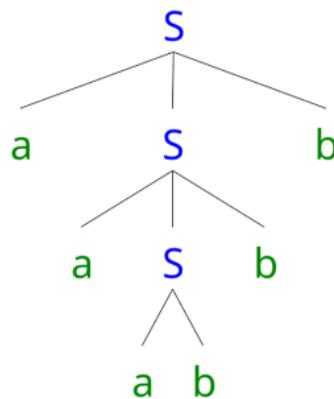


Parsing

- Given a string and production rules
- Produce a valid parse tree

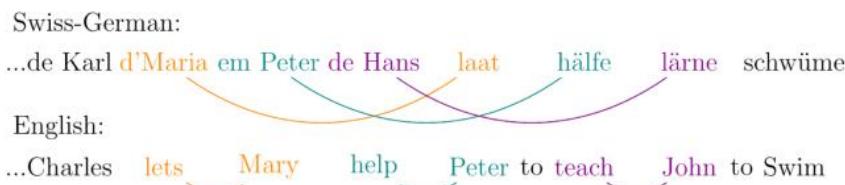
↳ $S \rightarrow a S b$
 ↳ $S \rightarrow a b$

aaabbb →



What This Means?

- If English can be represented with CFG:
 - ▶ we can build a “parser” to automatically judge whether a sentence is grammatical!
- But is natural language context-free?
- Not quite: cross-serial dependencies ($a^m b^n c^m d^n$)



7

But...

- Context-free representations strike a good balance:
 - ▶ CFG cover most syntactic patterns
 - ▶ CFG parsing is computational efficient
- We use CFG to describe a core fragment of English syntax

8

Syntactic Constituents

- Sentences are broken into **constituents**
 - ▶ word sequence that function as a **coherent unit** for linguistic analysis
- Constituents have certain **key properties**:
 - ▶ movement
 - ▶ substitution
 - ▶ coordination

9

Movement

- Constituents can be moved around sentences
 - ▶ Abigail gave [her brother] [a fish]
 - ▶ Abigail gave [a fish] to [her brother]
- Contrast: [gave her], [brother a]
cannot be swapped

10

Substitution

- Constituents can be substituted by other phrases of the same type
 - Max thanked [his older sister]
 - Max thanked [her]
- Contrast: Max thanked, thanked his
can't be substituted

11

Coordination

- Constituents can be conjoined with coordinators like *and* and *or*
 - [Abigail] and [her young brother] brought a fish
 - Abigail [brought a fish] and [gave it to Max]
 - Abigail [brought] and [greedily ate] a fish
- Contrast: [brother brought], [brought a]


12

Constituents and Phrases

- Once we identify constituents, **we use phrases to describe them**
- Phrases are **determined by their head word:**
 - noun phrase:** her younger **brother**
 - verb phrase:** greedily **ate it**
- We can use CFG to formalise these intuitions**

13

A Simple CFG for English

Terminal symbols: *rat, the, ate, cheese*

Non-terminal symbols: S, NP, VP, DT, VBD, NN

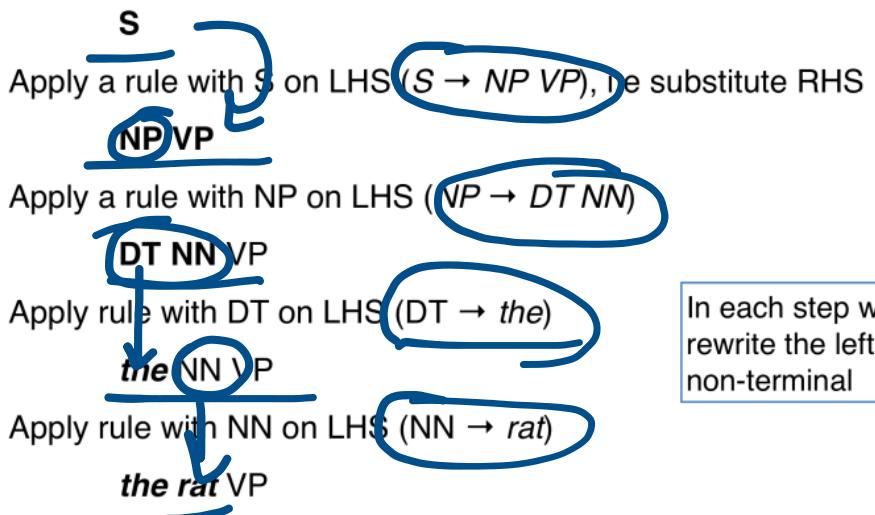
Productions:

$$\begin{aligned} S &\rightarrow NP\ VP \\ NP &\rightarrow DT\ NN \\ VP &\rightarrow VBD\ NP \\ DT &\rightarrow the \\ NN &\rightarrow rat \\ NN &\rightarrow cheese \\ VBD &\rightarrow ate \end{aligned}$$

14

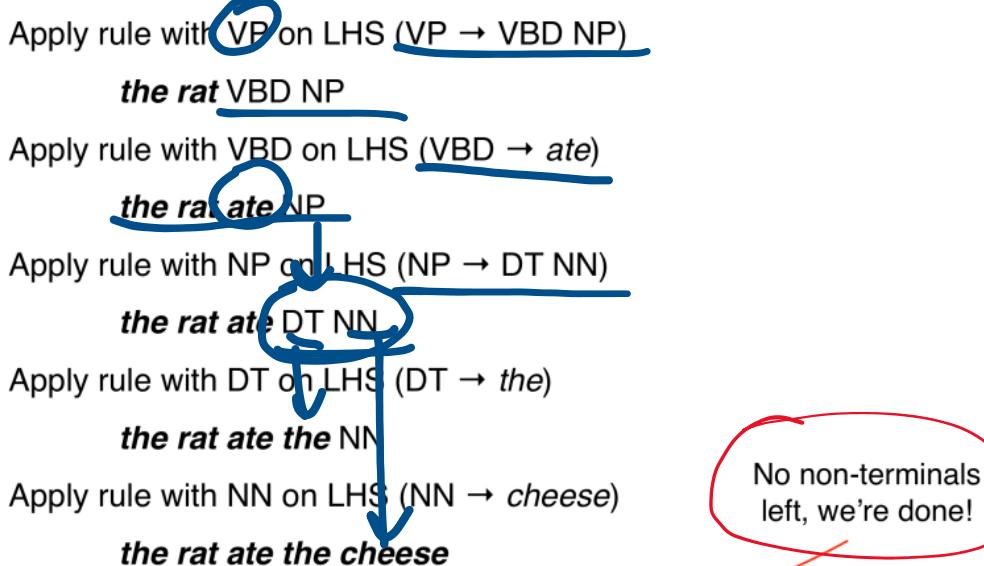
Generating Sentences with CFGs

Always start with S (the sentence/start symbol)



15

Generating Sentences with CFGs

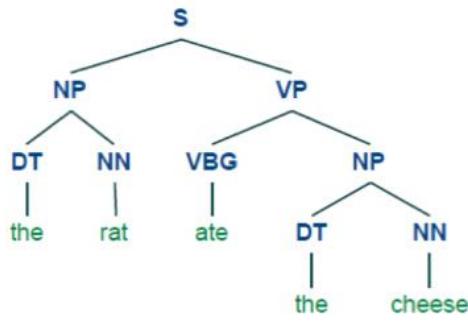


16

CFG Trees

- Generation corresponds to a syntactic tree
- Non-terminals are internal nodes
- Terminals are leaves

(S (NP (DT the)
 (NN rat))
 (VP (VBG ate)
 (NP (DT the)
 (NN cheese))))



- Parsing is the reverse process

sentence \Rightarrow tree

17

A CFG for Arithmetic Expressions

$$\begin{aligned}
 S &\rightarrow S \text{ OP } S \mid \text{NUM} \\
 \text{OP} &\rightarrow + \mid - \mid \times \mid \div \\
 \text{NUM} &\rightarrow \text{NUM DIGIT} \mid \text{DIGIT} \\
 \text{DIGIT} &\rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9
 \end{aligned}$$

- S = starting symbol
- '|' = operator OR
- Recursive, NUM and S can produce themselves

18

Parsing

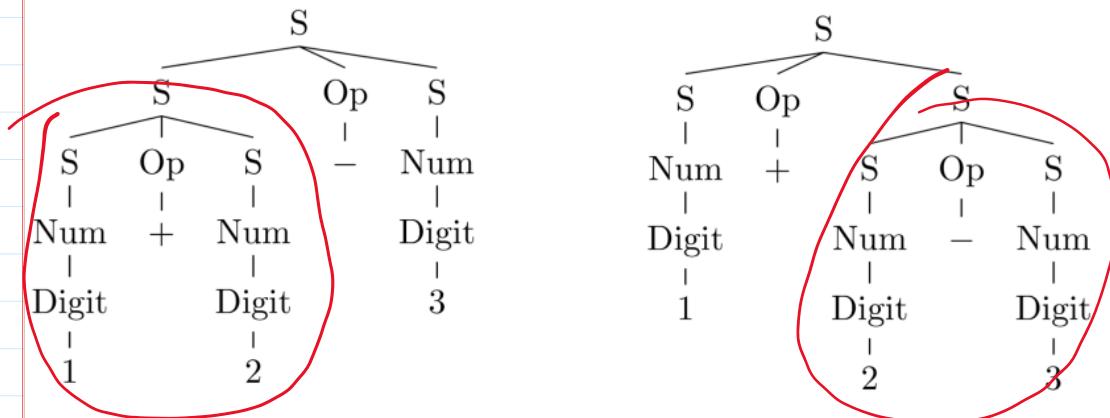
- Is '4' a valid string?

$$\begin{aligned} S &\rightarrow S \text{ OP } S \mid \text{NUM} \\ \text{OP} &\rightarrow + \mid - \mid \times \mid \div \\ \text{NUM} &\rightarrow \text{NUM DIGIT} \mid \text{DIGIT} \\ \text{DIGIT} &\rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9 \end{aligned}$$


19

Parsing

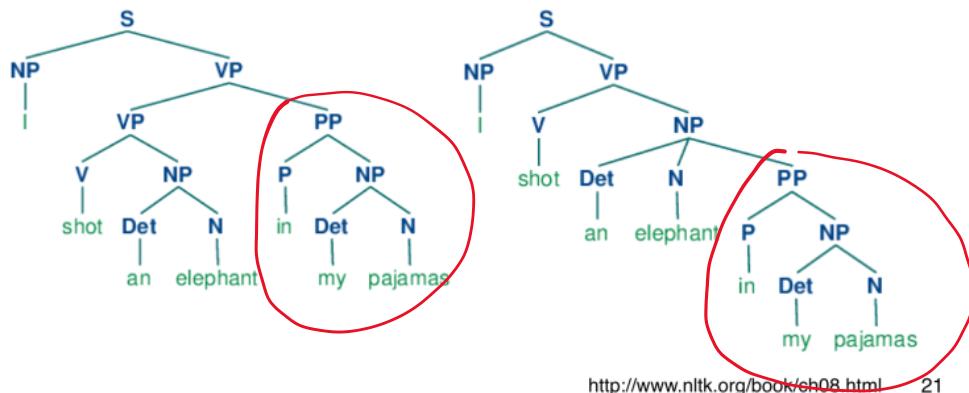
- Is '1+2-3' a valid string?

$$\begin{aligned} S &\rightarrow S \text{ OP } S \mid \text{NUM} \\ \text{OP} &\rightarrow + \mid - \mid \times \mid \div \\ \text{NUM} &\rightarrow \text{NUM DIGIT} \mid \text{DIGIT} \\ \text{DIGIT} &\rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9 \end{aligned}$$


20

Parse Ambiguity

- Often more than one tree can describe a string
- “While hunting in Africa, I shot an elephant in my pajamas. How he got into my pajamas, I don't know.”
Animal Crackers (1930)



Parsing CFG

CYK Algorithm

- Bottom-up approach to parsing in CFG
- Tests whether a string is valid given a CFG, without enumerating all possible parses
- Core idea: form small constituents first, and merge them into larger constituents
- Requirement: CFGs must be in Chomsky Normal Forms

23

Convert to Chomsky Normal Form

- Change grammar so all rules of form:
 - $A \rightarrow B C$ *non-terminal \rightarrow 2x non-terminal*
 - $A \rightarrow a$ *non-terminal \rightarrow terminal*
- Convert rules of form $A \rightarrow B c$ into:
 - $A \rightarrow B X$
 - $X \rightarrow c$

24

Convert to Chomsky Normal Form

- Convert rules $A \rightarrow B C D$ into:
 - $A \rightarrow B Y$
 - $Y \rightarrow C D$
 - E.g., $VP \rightarrow VP NP NP$
for ditransitive cases, “*sold [her] [the book]*”
- X, Y are new symbols we have introduced

25

CNF (cont)

- CNF disallows unary rules, $A \rightarrow B$.
- Imagine $NP \rightarrow S$; and $S \rightarrow NP \dots$ leads to infinitely many trees with same yield.
- Replace RHS non-terminal with its productions
 - E.g convert $A \rightarrow B$, $B \rightarrow 1$, $B \rightarrow 2$ into:
 - $A \rightarrow 1, A \rightarrow 2$

26

The CYK Parsing Algorithm

- Convert grammar to Chomsky Normal Form (CNF)
- Fill in a parse table
- Use table to derive parse
- S in top right corner of table = success!
- Convert result back to original grammar

27

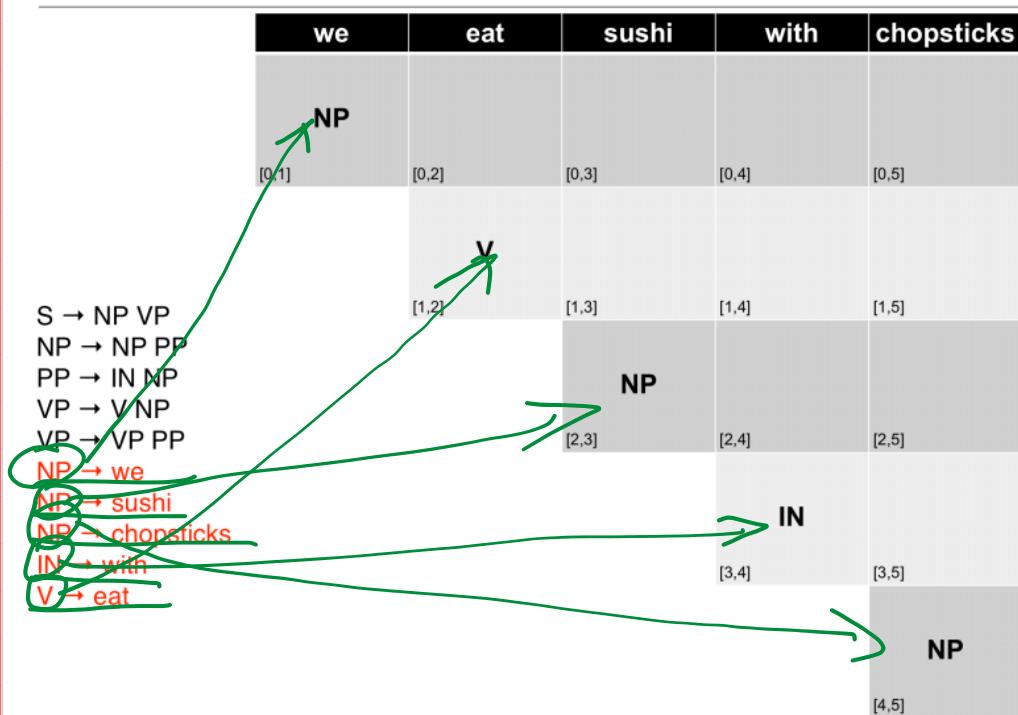
COMP90042 → *words* L14 → *column*

we	eat	sushi	with	chopsticks
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
[1,2]	[1,3]	[1,4]	[1,5]	
[2,3]	[2,4]	[2,5]		
[3,4]	[3,5]			[4,5]

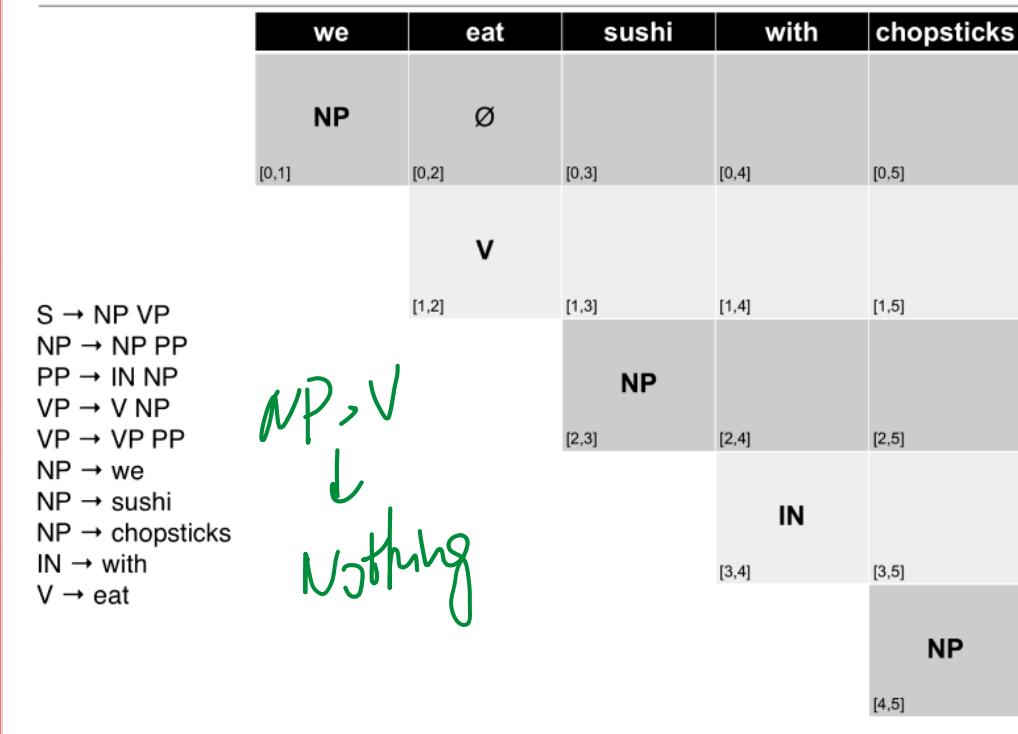
row id

$S \rightarrow NP\ VP$
 $NP \rightarrow NP\ PP$
 $PP \rightarrow IN\ NP$
 $VP \rightarrow V\ NP$
 $VP \rightarrow VP\ PP$
 $NP \rightarrow we$
 $NP \rightarrow sushi$
 $NP \rightarrow chopsticks$
 $IN \rightarrow with$
 $V \rightarrow eat$

28



29



30

we	eat	sushi	with	chopsticks
NP	\emptyset			
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		V ← VP Split=2		
		NP		
		[1,2]	[1,4]	[1,5]
			[2,4]	[2,5]
			IN	
			[3,4]	[3,5]
				NP
				[4,5]

31

we	eat	sushi	with	chopsticks
NP	\emptyset	S		
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		Split=1		
		V ← VP Split=2		
		NP		
		[1,2]	[1,4]	[1,5]
			[2,4]	[2,5]
			IN	
			[3,4]	[3,5]
				NP
				[4,5]

32

$S \rightarrow NP VP$

NP → NP PP

PP → IN NP

VP → V NP

VP → VP PP

NP → we

NP → sushi

NP → chopsticks

IN → with

V → eat

NP, VP

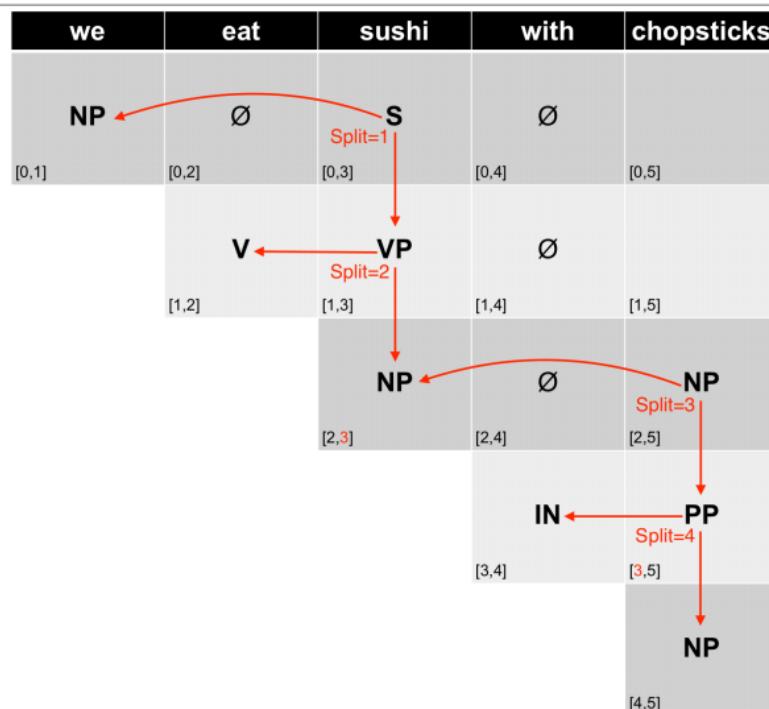
NP, NP \Rightarrow Nothing

we	eat	sushi	with	chopsticks
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
NP	Ø	S Split=1	Ø	
[1,2]	[1,3]	VP Split=2	Ø	[1,5]
V	NP		Ø	
[2,3]	[2,4]		[2,5]	
		IN		
		[3,4]	[3,5]	
				NP
			[4,5]	

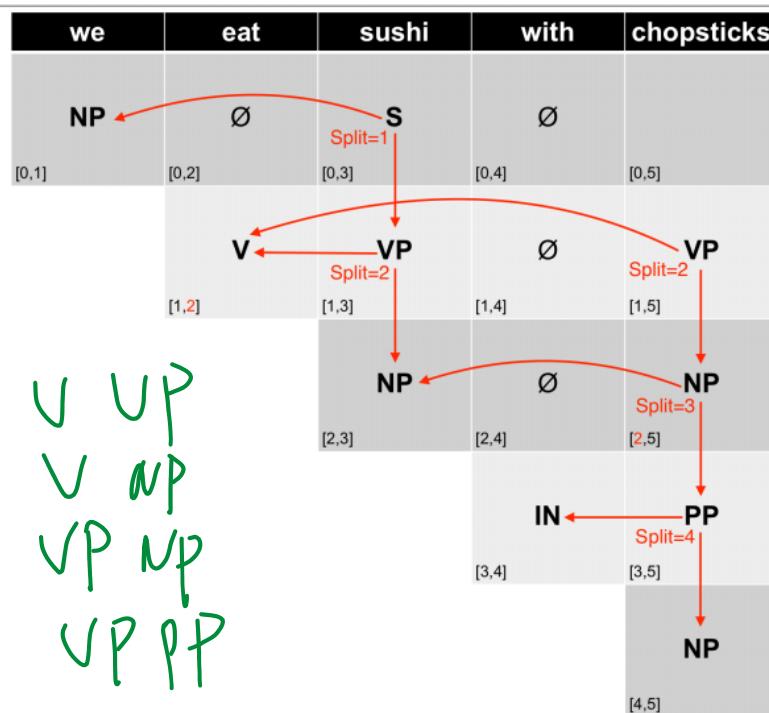
33

we	eat	sushi	with	chopsticks
[0,1]	[0,2]	[0,3] S Split=1	[0,4]	[0,5]
		VP Split=2		
[1,2]	[1,3]	[1,4]	[1,5]	
		NP		
	[2,3]	[2,4]	[2,5]	
		IN PP Split=4		
	[3,4]	[3,5]		
		NP		
			[4,5]	

34



35



36

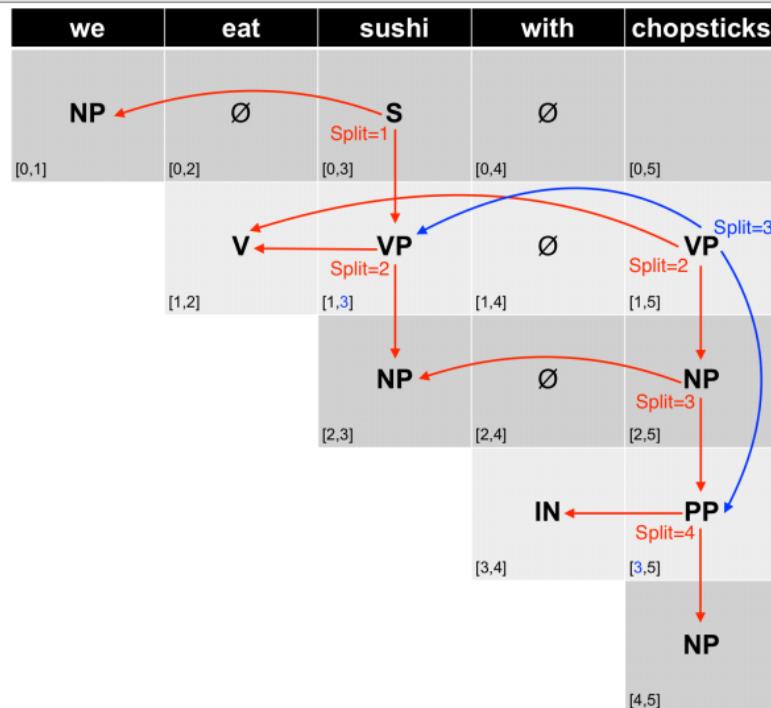
V UP

V NP

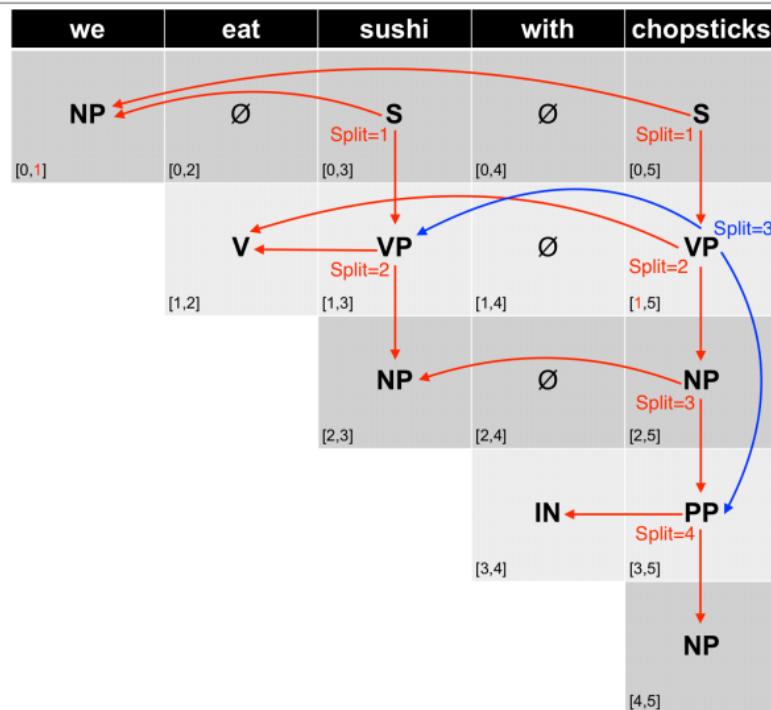
VP NP

VP PP

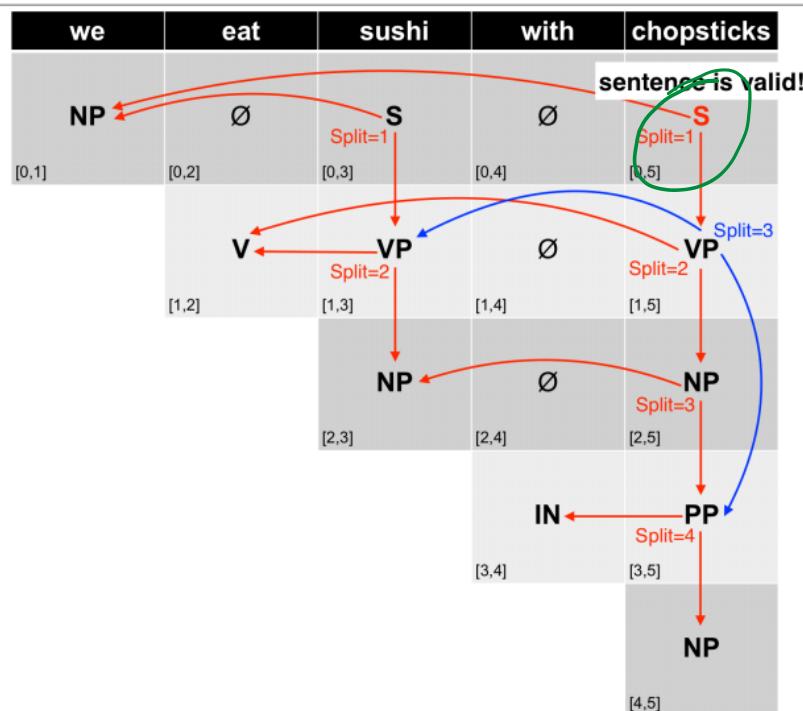
S → NP VP
 NP → NP PP
 PP → IN NP
VP → V NP
 VP → VP PP
 NP → we
 NP → sushi
 NP → chopsticks
 IN → with
 V → eat



37



38

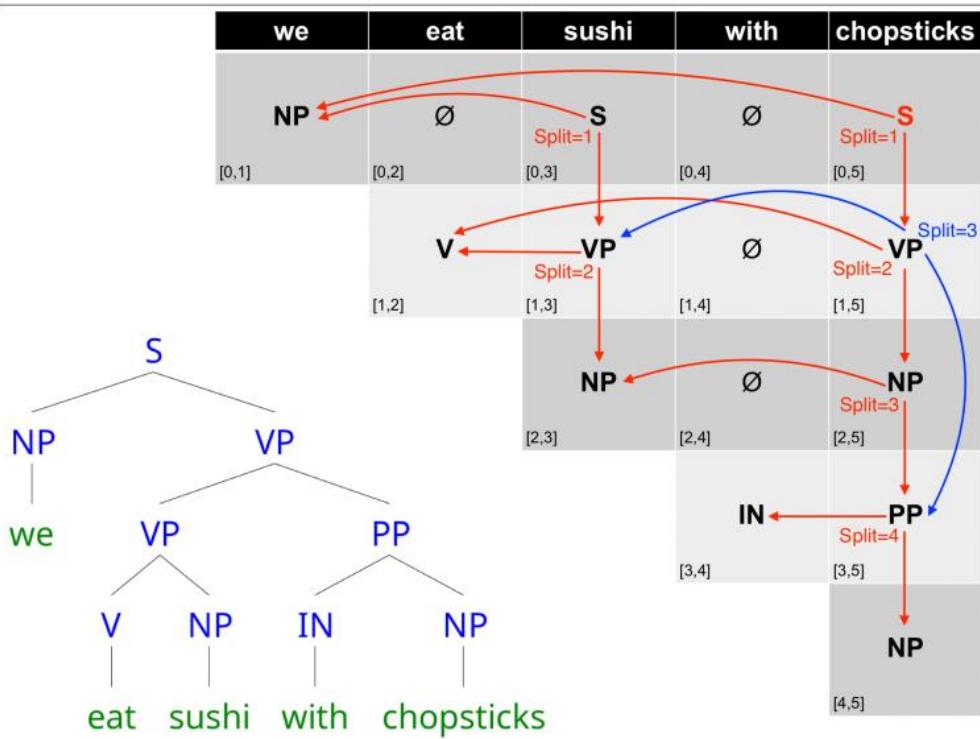


39

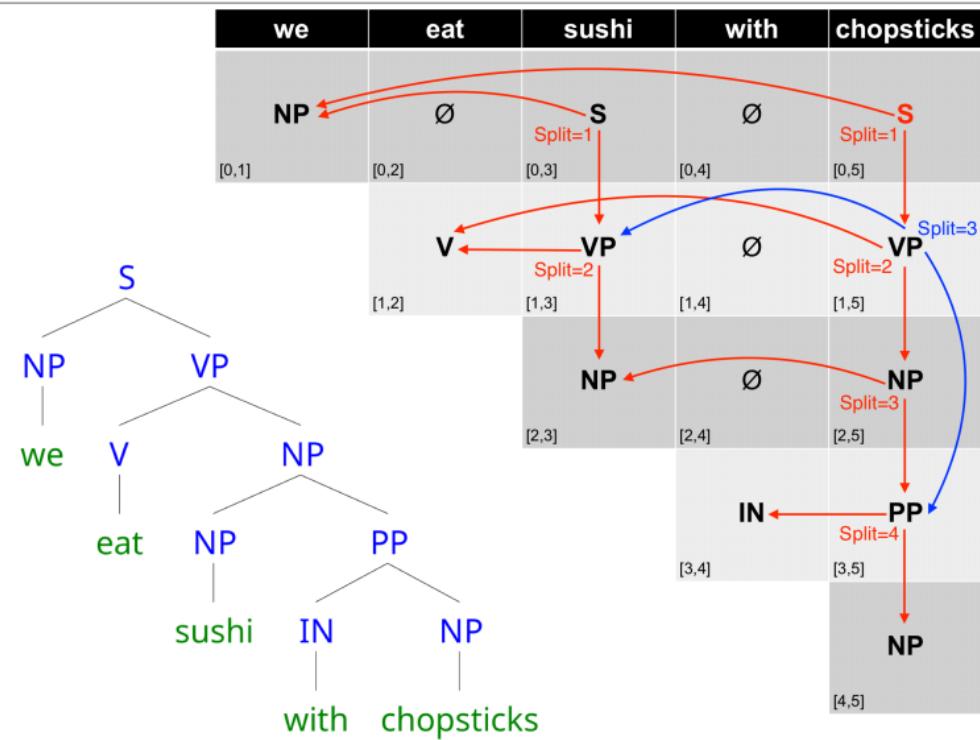
CYK: Retrieving the Parses

- S in the top-left corner of parse table indicates success
- To get parse(s), follow pointers back for each match

40



41



42

CYK Algorithm

```

function CKY-PARSE(words, grammar) returns table
    for j ← from 1 to LENGTH(words) do
        for all {A | A → words[j] ∈ grammar}
            table[j − 1, j] ← table[j − 1, j] ∪ A
        for i ← from j − 2 downto 0 do
            for k ← i + 1 to j − 1 do
                for all {A | A → BC ∈ grammar and B ∈ table[i, k] and C ∈ table[k, j]}
                    table[i, j] ← table[i, j] ∪ A
    
```

Figure 12.5 The CKY algorithm.

$\mathcal{O}(n^3)$

JM3, Ch 12 43

Representing English with CFGs

From Toy Grammars to Real Grammars

- Toy grammars with handful of productions good for demonstration or extremely limited domains
- For real texts, we need real grammars
- Many thousands of production rules

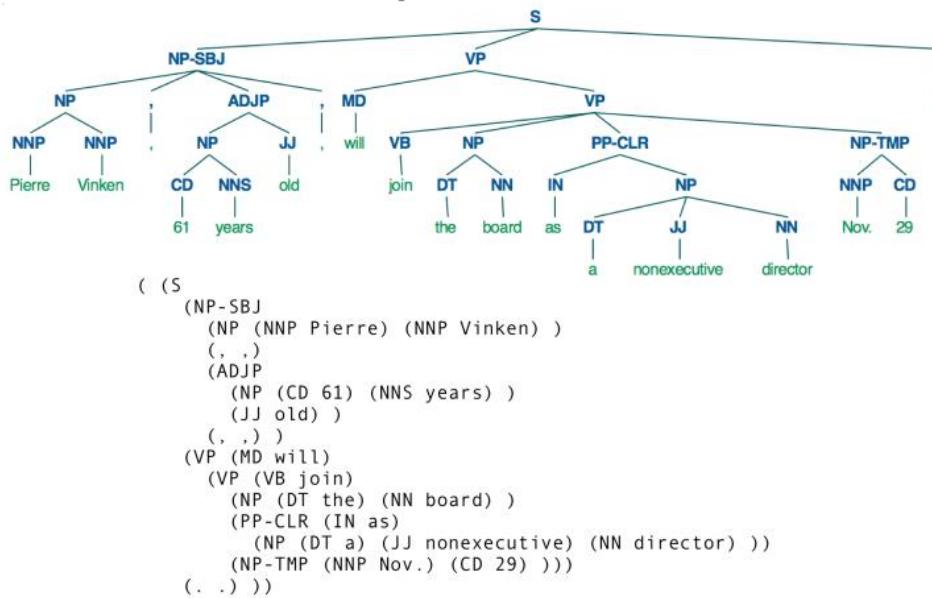
45

Key Constituents in Penn Treebank

- Sentence (S)
- Noun phrase (NP)
- Verb phrase (VP)
- Prepositional phrase (PP)
- Adjective phrase (AdjP)
- Adverbial phrase (AdvP)
- Subordinate clause (SBAR)

46

Example PTB/0001



47

Basic English Sentence Structures

- **Declarative sentences** ($S \rightarrow NP\ VP$)
 - *The rat ate the cheese*
- **Imperative sentences** ($S \rightarrow VP$)
 - *Eat the cheese!*
- **Yes/no questions** ($S \rightarrow \cancel{VB}\ NP\ VP$)
 - *Did the rat eat the cheese?*
- **Wh-subject-questions** ($S \rightarrow WH\ VP$)
 - *Who ate the cheese?*
- **Wh-object-questions** ($S \rightarrow WH\ \cancel{VB}\ NP\ VP$)
 - *What did the rat eat?*

48

English Noun Phrases

- **Pre-modifiers**
 - ▶ DT, CD, ADJP, NNP, NN
 - ▶ E.g. *the two very best Philly cheese steaks*

- **Post-modifiers**
 - ▶ PP, VP, SBAR
 - ▶ A delivery from Bob coming today that I don't want to miss

NP → DT? CD? ADJP? (NNINNP)+ PP* VP? SBAR?

NP → PRP

pre

post

Verb Phrases

- Auxiliaries
 - ▶ MD, AdvP, VB, TO
 - ▶ E.g *should really have tried to wait*

$\text{VP} \rightarrow (\text{MDIVBITO}) \text{AdvP? VP}$

- Arguments and adjuncts
 - ▶ NP, PP, SBAR, VP, AdvP
 - ▶ E.g. *told him yesterday that I was ready*
 - ▶ E.g. *gave John a gift for his birthday to make amends*

$\text{VP} \rightarrow \text{VB } \text{NP? } \text{NP? } \text{PP}^* \text{ AdvP}^* \text{ VP? } \text{SBAR?}$

Other Constituents

- **Prepositional phrase**
 - $PP \rightarrow IN\ NP$ *in the house*
- **Adjective phrase**
 - $AdjP \rightarrow (AdvP)\ JJ$ *really nice*
- **Adverb phrase**
 - $AdvP \rightarrow (AdvP)\ RB$ *not too well*
- **Subordinate clause**
 - $SBAR \rightarrow (IN)\ S$ *since I came here*
- **Coordination**
 - $NP \rightarrow NP\ CC\ NP; VP \rightarrow VP\ CC\ VP;$ etc. *Jack and Jill*
- **Complex sentences**
 - $S \rightarrow S\ SBAR; S \rightarrow SBAR\ S;$ etc. *if he goes, I'll go*

51

A Final Word

- Context-free grammars can represent linguistic structure
- There are relatively fast dynamic programming algorithms to retrieve this structure
- But what about ambiguity?
 - Extreme ambiguity will slow down parsing
 - If multiple possible parses, which is best?

52

Readings

- E18 Ch. 9.2, 10.1



Probabilistic Context-Free Grammar

COMP90042

Natural Language Processing

Lecture 15



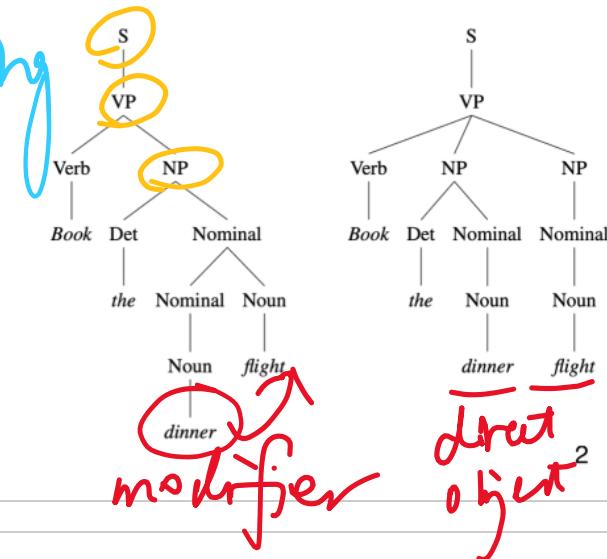
COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

Ambiguity In Parsing

CFG tree

- Context-free grammars assign hierarchical structure to language
 - ▶ Linguistic notion of a '*syntactic constituent*'
 - ▶ Formulated as generating all strings in the language; or
 - ▶ Predicting the structure(s) for a given string → *parse*
- Raises problem of ambiguity, e.g., which is better?



Outline

- Probabilistic context-free grammars (PCFGs)
- Parsing using dynamic programming
- Limitations of 'context-free' assumption and some solutions:
 - ▶ parent annotation
 - ▶ head lexicalisation

Basics of Probabilistic CFGs

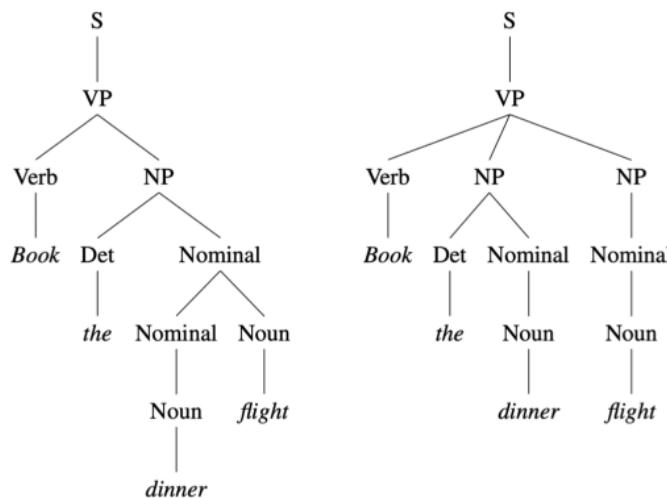
- As for CFGs, same symbol set:
 - Terminals: words such as *book*
 - Non-terminal: syntactic labels such as NP or NN
- Same productions (rules)
 - LHS non-terminal → ordered list of RHS symbols
- In addition, store a **probability** with each production
 - NP → DT NN [p = 0.45]
 - NN → cat [p = 0.02]
 - NN → leprechaun [p = 0.00001]
 - ...

4

Probabilistic CFGs

- Probability values denote **conditional**
 - Pr(LHS → RHS)
 - Pr(RHS | LHS)
- Consequently they:
 - must be positive values, between 0 and 1
 - must sum to one for given LHS
- E.g.,
 - NN → aadvark [p = 0.0003]
 - NN → cat [p = 0.02]
 - NN → leprechaun [p = 0.0001]
 - $\sum_x \Pr(\text{NN} \rightarrow x) = 1$

5



	Rules	P		Rules	P
S	\rightarrow VP	.05	S	\rightarrow VP	.05
VP	\rightarrow Verb NP	.20	VP	\rightarrow Verb NP NP	.10
NP	\rightarrow Det Nominal	.20	NP	\rightarrow Det Nominal	.20
Nominal	\rightarrow Nominal Noun	.20	NP	\rightarrow Nominal	.15
Nominal	\rightarrow Noun	.75	Nominal	\rightarrow Noun	.75
Verb	\rightarrow book	.30	Verb	\rightarrow book	.30
Det	\rightarrow the	.60	Det	\rightarrow the	.60
Noun	\rightarrow dinner	.10	Noun	\rightarrow dinner	.10
Noun	\rightarrow flight	.40	Noun	\rightarrow flight	.40

Stochastic Generation with PCFGs

Almost the same as for CFG, with one twist:

1. Start with S, the sentence symbol
 2. Choose a rule with S as the LHS
 - ▶ Randomly select a RHS according to $\text{Pr}(\text{RHS} \mid \text{LHS})$
 - e.g., $S \rightarrow VP$
 - ▶ Apply this rule, e.g., substitute VP for S
 3. Repeat step 2 for each non-terminal in the string (here, VP)
 4. Stop when no non-terminals remain
- Gives us a tree, as before, with a sentence as the yield

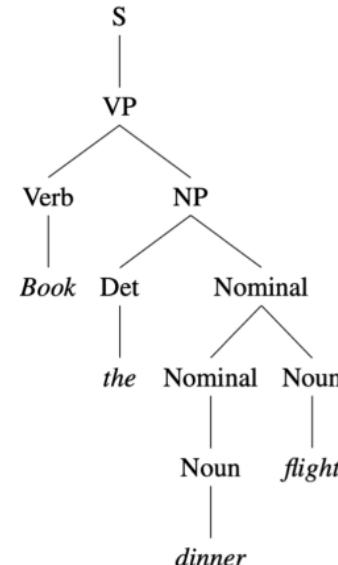
How Likely is a Tree?

- Given a tree, we can compute its probability
 - Decomposes into probability of each production

- E.g., for (left) tree,

► $P(\text{tree}) =$

$P(S \rightarrow VP) \times$
 $P(VP \rightarrow \text{Verb NP}) \times$
 $P(\text{Verb} \rightarrow Book) \times$
 $P(NP \rightarrow \text{Det Nominal}) \times$
 $P(\text{Det} \rightarrow the) \times$
 $P(\text{Nominal} \rightarrow \text{Nominal Noun}) \times$
 $P(\text{Nominal} \rightarrow \text{Noun}) \times$
 $P(\text{Noun} \rightarrow dinner) \times$
 $P(\text{Noun} \rightarrow flight)$



8

How Likely is a Tree?

$P(\text{tree})$

$= P(S \rightarrow VP) \times P(VP \rightarrow \text{Verb NP}) \times P(\text{Verb} \rightarrow Book) \times$
 $P(NP \rightarrow \text{Det Nominal}) \times P(\text{Det} \rightarrow the) \times P(\text{Nominal} \rightarrow \text{Nominal Noun}) \times$
 $P(\text{Nominal} \rightarrow \text{Noun}) \times P(\text{Noun} \rightarrow dinner) \times P(\text{Noun} \rightarrow flight)$

$$= 0.05 \times 0.20 \times 0.30 \times$$

$$0.20 \times 0.60 \times 0.20 \times$$

$$0.75 \times 0.10 \times 0.40$$

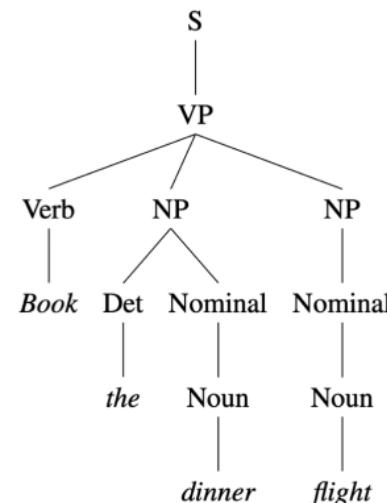
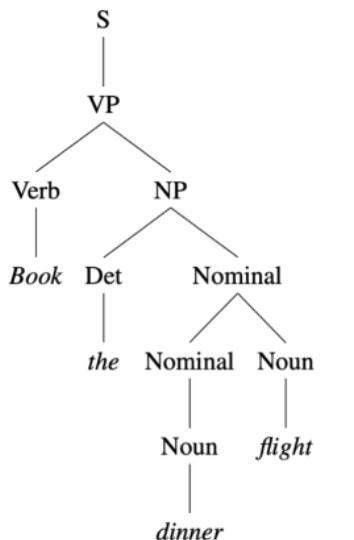
$$= 2.2 \times 10^{-6}$$

	Rules	P
S	$\rightarrow VP$.05
VP	$\rightarrow \text{Verb NP}$.20
NP	$\rightarrow \text{Det Nominal}$.20
Nominal	$\rightarrow \text{Nominal Noun}$.20
Nominal	$\rightarrow \text{Noun}$.75
Verb	$\rightarrow book$.30
Det	$\rightarrow the$.60
Noun	$\rightarrow dinner$.10
Noun	$\rightarrow flight$.40

9

Resolving Parse Ambiguity

- Can select between different trees based on $P(T)$
- $P(T_{\text{left}}) = 2.2 \times 10^{-6}$ $P(T_{\text{right}}) = 6.1 \times 10^{-7}$



10

Parsing PCFGs

- Instead of selecting between two trees, can we select a tree from the set of all possible trees?
- Before we looked at
 - CYK
 - for unweighted grammars (CFGs)
 - finds all possible trees
- But there are often 1000s, many completely nonsensical
- Can we solve for the most probable tree?

11

CYK for PCFGs

- CYK finds **all trees** for a sentence; we want **best** tree
- Prob. CYK follows similar process to standard CYK
- Convert grammar to Chomsky Normal Form (CNF)
 - ▶ E.g., $VP \rightarrow Verb \ NP \ NP$ [0.10]
 - ▶ becomes $VP \rightarrow Verb \ NP+NP$ [0.10]
 $NP+NP \rightarrow NP \ NP$ [1.0]
 - ▶ where $NP+NP$ is a new symbol.

12

PCFG Parsing Example

13

	we	eat	sushi	with	chopsticks
	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		[1,2]	[1,3]	[1,4]	[1,5]
S → NP VP	1				
NP → NP PP	$\frac{1}{2}$				
→ we	$\frac{1}{4}$				
→ sushi	$\frac{1}{8}$				
→ chopsticks	$\frac{1}{8}$				
PP → IN NP	1			[3,4]	
IN → with	1				[3,5]
VP → V NP	$\frac{1}{2}$				
→ VP PP	$\frac{1}{4}$				
→ MD V	$\frac{1}{4}$				[4,5]
V → eat	1				

14

	we	eat	sushi	with	chopsticks
	NP 1/4				
	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		[1,2]	[1,3]	[1,4]	[1,5]
S → NP VP	1				
NP → NP PP	$\frac{1}{2}$				
→ we	$\frac{1}{4}$				
→ sushi	$\frac{1}{8}$				
→ chopsticks	$\frac{1}{8}$				
PP → IN NP	1				
IN → with	1				
VP → V NP	$\frac{1}{2}$				
→ VP PP	$\frac{1}{4}$				
→ MD V	$\frac{1}{4}$				
V → eat	1				

15

we	eat	sushi	with	chopsticks
NP 1/4	Ø			
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	V 1			
	[1,2]	[1,3]	[1,4]	[1,5]
S → NP VP	1			
NP → NP PP	$\frac{1}{2}$			
→ we	$\frac{1}{4}$			
→ sushi	$\frac{1}{8}$			
→ chopsticks	$\frac{1}{8}$			
PP → IN NP	1			
IN → with	1			
VP → V NP	$\frac{1}{2}$			
→ VP PP	$\frac{1}{4}$			
→ MD V	$\frac{1}{4}$			
V → eat	1			

16

? → NP ✓
Nothing

we	eat	sushi	with	chopsticks
NP 1/4	Ø			
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	V 1 ← VP 1/16 ($\frac{1}{2} * \frac{1}{4} * \frac{1}{8}$)			
	[1,2]	[1,3]	[1,4]	[1,5]
S → NP VP	1			
NP → NP PP	$\frac{1}{2}$			
→ we	$\frac{1}{4}$			
→ sushi	$\frac{1}{8}$			
→ chopsticks	$\frac{1}{8}$			
PP → IN NP	1			
IN → with	1			
VP → V NP	$\frac{1}{2}$			
→ VP PP	$\frac{1}{4}$			
→ MD V	$\frac{1}{4}$			
V → eat	1			

17

? → ✓ NP

we	eat	sushi	with	chopsticks
NP 1/4 [0,1]	Ø [0,2]	S 1/64 (1 * 1/4 * 1/64) [0,3]		
		V 1 ← VP 1/16 [1,2]		
		[1,3]	[1,4]	[1,5]
S → NP VP NP → NP PP → we → sushi → chopsticks PP → IN NP IN → with VP → V NP → VP PP → MD V V → eat	1 ½ ¼ ⅛ ⅛ 1 1 ½ ¼ ¼ 1	NP 1/8 [2,3]	IN 1 [3,4]	NP 1/8 [4,5]

18

we	eat	sushi	with	chopsticks
NP 1/4 [0,1]	Ø [0,2]	S 1/64 [0,3]	Ø [0,4]	Ø [0,5]
		V 1 ← VP 1/16 [1,2]	Ø [1,4]	Ø [1,5]
		[1,3]		
S → NP VP NP → NP PP → we → sushi → chopsticks PP → IN NP IN → with VP → V NP → VP PP → MD V V → eat	1 ½ ¼ ⅛ ⅛ 1 1 ½ ¼ ¼ 1	NP 1/8 [2,3]	IN 1 [3,4]	NP 1/8 [4,5]

19

we	eat	sushi	with	chopsticks
NP 1/4	Ø	S 1/64	Ø	
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		VP 1/16	Ø	
	[1,2]	[1,3]	[1,4]	[1,5]
		NP 1/8	Ø	
		[2,3]	[2,4]	[2,5]
S → NP VP	1			
NP → NP PP	½			
→ we	¼			
→ sushi	⅛			
→ chopsticks	⅛			
PP → IN NP	1			
IN → with	1			
VP → V NP	½			
→ VP PP	¼			
→ MD V	¼			
V → eat	1			

20

? → IN NP

we	eat	sushi	with	chopsticks
NP 1/4	Ø	S 1/64	Ø	
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		VP 1/16	Ø	
	[1,2]	[1,3]	[1,4]	[1,5]
		NP 1/8	Ø	NP 1/128
		[2,3]	[2,4]	[2,5]
S → NP VP	1			
NP → NP PP	½			
→ we	¼			
→ sushi	⅛			
→ chopsticks	⅛			
PP → IN NP	1			
IN → with	1			
VP → V NP	½			
→ VP PP	¼			
→ MD V	¼			
V → eat	1			

21

? → NP PP

	we	eat	sushi	with	chopsticks
	NP 1/4	Ø	S 1/64	Ø	
	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		V 1 ← VP 1/16		Ø	VP 1/256 (1/2 * 1 * 1/128)
		[1,2]	[1,3]	[1,4]	[1,5]
			NP 1/8	Ø	NP 1/128
			[2,3]	[2,4]	[2,5]
			IN 1 ← PP 1/8		
			[3,4]		
					NP 1/8
					[4,5]
S → NP VP	1				
NP → NP PP	½				
→ we	¼				
→ sushi	⅛				
→ chopsticks	⅛				
PP → IN NP	1				
IN → with	1				
VP → V NP	½				
→ VP PP	¼				
→ MD V	¼				
V → eat	1				

22

	we	eat	sushi	with	chopsticks
	NP 1/4	Ø	S 1/64	Ø	
	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		V 1 ← VP 1/16	→ VP 1/16	Ø	VP 1/512 (1/4 * 1/16 * 1/8)
		[1,2]	[1,3]	[1,4]	[1,5]
			NP 1/8	Ø	NP 1/128
			[2,3]	[2,4]	[2,5]
			IN 1 ← PP 1/8		
			[3,4]		
					NP 1/8
					[4,5]
S → NP VP	1				
NP → NP PP	½				
→ we	¼				
→ sushi	⅛				
→ chopsticks	⅛				
PP → IN NP	1				
IN → with	1				
VP → V NP	½				
→ VP PP	¼				
→ MD V	¼				
V → eat	1				

23

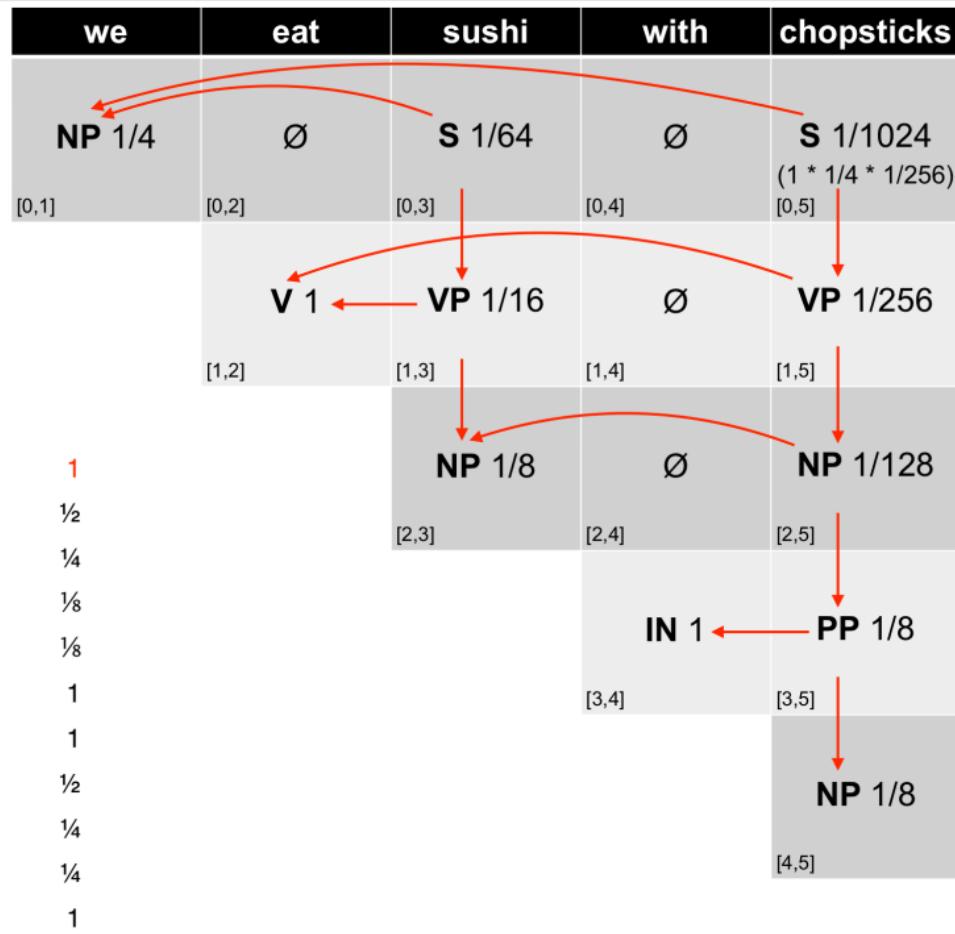
" " " "

parse tree

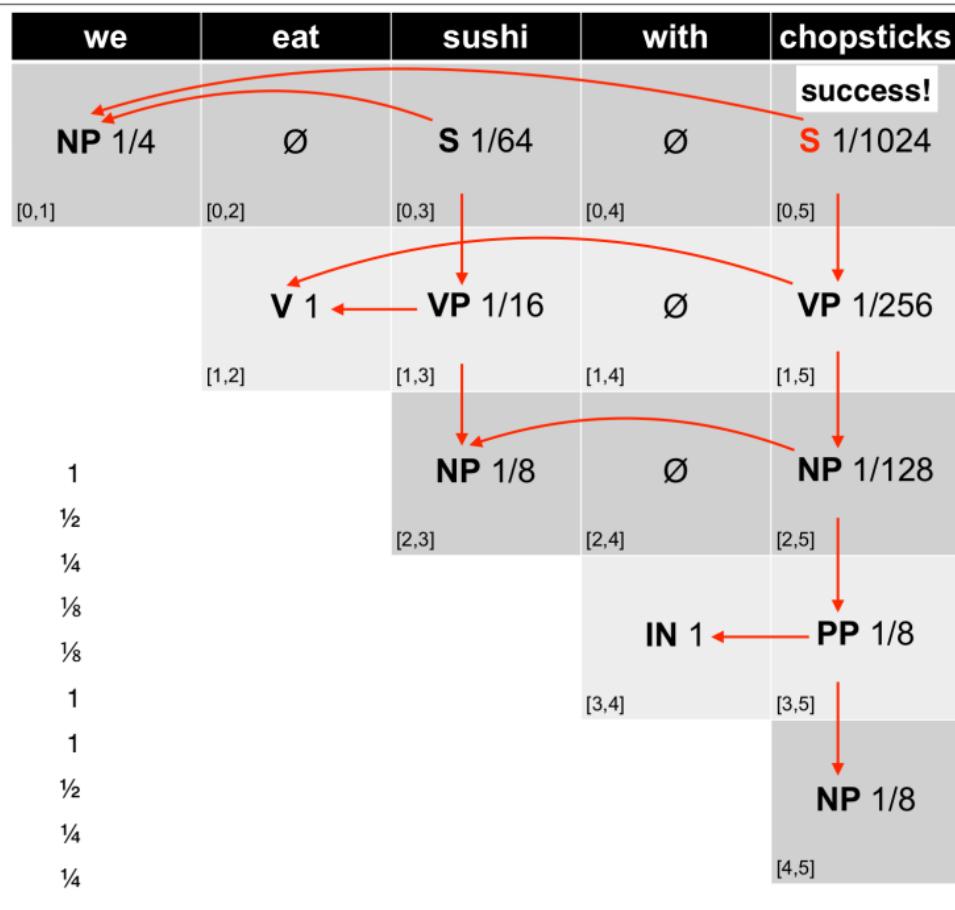
COMP90042

L15

we	eat	sushi	with	chopsticks
NP 1/4	\emptyset	S 1/64	\emptyset	
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	V 1	VP 1/16	\emptyset	VP 1/256
	[1,2]	[1,3]	[1,4]	[1,5]
		NP 1/8	\emptyset	NP 1/128
		[2,3]	[2,4]	[2,5]
			IN 1	PP 1/8
			[3,4]	[3,5]
				NP 1/8
				[4,5]
S → NP VP	1			
NP → NP PP	$\frac{1}{2}$			
→ we	$\frac{1}{4}$			
→ sushi	$\frac{1}{8}$			
→ chopsticks	$\frac{1}{8}$			
PP → IN NP	1			
IN → with	1			
VP → V NP	$\frac{1}{2}$			
→ VP PP	$\frac{1}{4}$			
→ MD V	$\frac{1}{4}$			
V → eat	1			



25

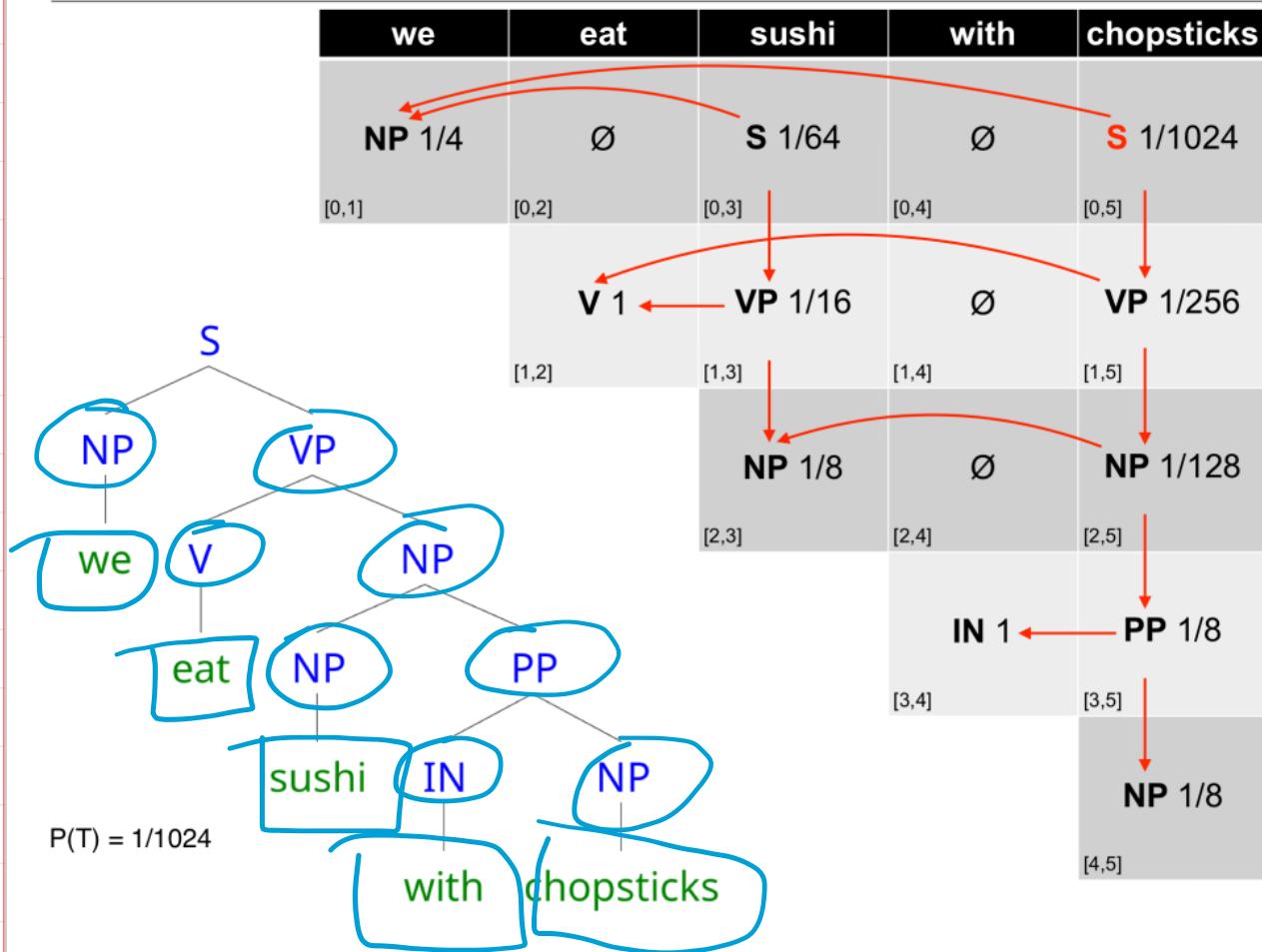


26

Prob CYK: Retrieving the Parses

- S in the top-right corner of parse table indicates success
- Retain back-pointer to best analysis
- To get parse(s), follow pointers back for each match
- Convert back from CNF by removing new non-terminals

27



28

Prob. CYK

```

function PROBABILISTIC-CYK(words, grammar) returns most probable parse
    and its probability
    for  $j \leftarrow$  from 1 to LENGTH(words) do
        for all {  $A \mid A \rightarrow words[j] \in grammar$  }
             $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
        for  $i \leftarrow$  from  $j-2$  downto 0 do
            for  $k \leftarrow i+1$  to  $j-1$  do
                for all {  $A \mid A \rightarrow BC \in grammar,$ 
                    and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
                    if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
                         $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
                         $back[i, j, A] \leftarrow \{k, B, C\}$ 
    return BUILD-TREE( $back[1, \text{LENGTH}(words), S]$ ),  $table[1, \text{LENGTH}(words), S]$ 

```

Source: JM3 Ch 14

29

CYK can be thought of as storing all events with probability = 1

function CKY-PARSE(*words, grammar*) **returns** *table*

```

    for  $j \leftarrow$  from 1 to LENGTH(words) do
        for all {  $A \mid A \rightarrow words[j] \in grammar$  }
             $table[j-1, j] \leftarrow table[j-1, j] \cup A$ 
    for  $i \leftarrow$  from  $j-2$  downto 0 do
        for  $k \leftarrow i+1$  to  $j-1$  do
            for all {  $A \mid A \rightarrow BC \in grammar$  and  $B \in table[i, k]$  and  $C \in table[k, j]$  }
                 $table[i, j] \leftarrow table[i, j] \cup A$ 

```

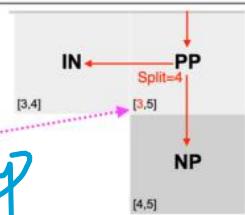


Figure 12.5 The CKY algorithm.

validity test now looks to see that the child chart cells have non-zero probability

function PROBABILISTIC-CYK(*words, grammar*) **returns** most probable parse
 and its probability

```

    for  $j \leftarrow$  from 1 to LENGTH(words) do
        for all {  $A \mid A \rightarrow words[j] \in grammar$  }
             $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
    for  $i \leftarrow$  from  $j-2$  downto 0 do
        for  $k \leftarrow i+1$  to  $j-1$  do
            for all {  $A \mid A \rightarrow BC \in grammar,$ 
                and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
                if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
                     $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
                     $back[i, j, A] \leftarrow \{k, B, C\}$ 
    return BUILD-TREE( $back[1, \text{LENGTH}(words), S]$ ),  $table[1, \text{LENGTH}(words), S]$ 

```

Instead of storing set of symbols, store the probability of best scoring tree fragment covering span $[i, j]$ with root symbol A

Overwrite lower scoring analysis if this one is better, and record the best production

chart now stores probabilities for each span and symbol

30

Complexity of CYK

- What's the space and time complexity of this algorithm?
 - in terms of n the length of the input sentence

space $O(1^y)$

time $O(n^3)$

31

Issues with PCFG

32

PCFG Problem 1: Poor Independence Assumptions

- Rewrite decisions made independently, whereas interdependence is often needed to capture global structure.

- $NP \rightarrow Det\ N$
- Probability of this rule independent of rest of tree

↙ or ↘

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

NP statistics in the Switchboard corpus

- No way to represent this contextual differences in PCFG probabilities

33

Poor Independence Assumptions

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

NP statistics in the Switchboard corpus

$$\begin{aligned} NP \rightarrow DT\ NN & .28 \\ NP \rightarrow PRP & .25 \end{aligned}$$

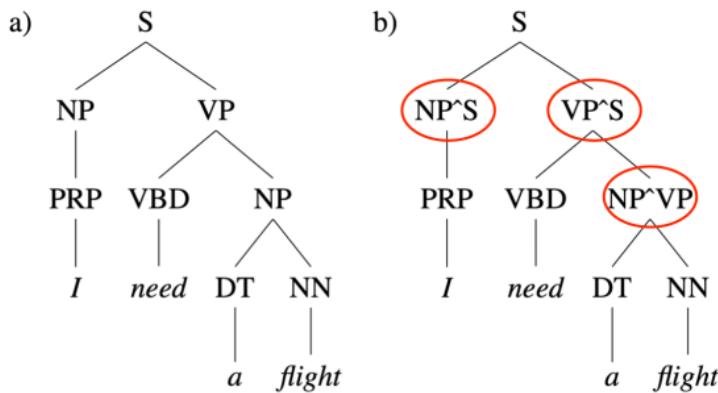
PCFG probabilities based on Switchboard corpus

- No way to capture the fact that in subject position, $NP \rightarrow PRP$ should go up to 0.91
- While in object position $NP \rightarrow DT\ NN$ should go up to 0.66
- Solution: add a condition to denote whether NP is a subject or object

34

Solution: Parent Conditioning

- Make non-terminals more explicit by incorporating parent symbol into each symbol

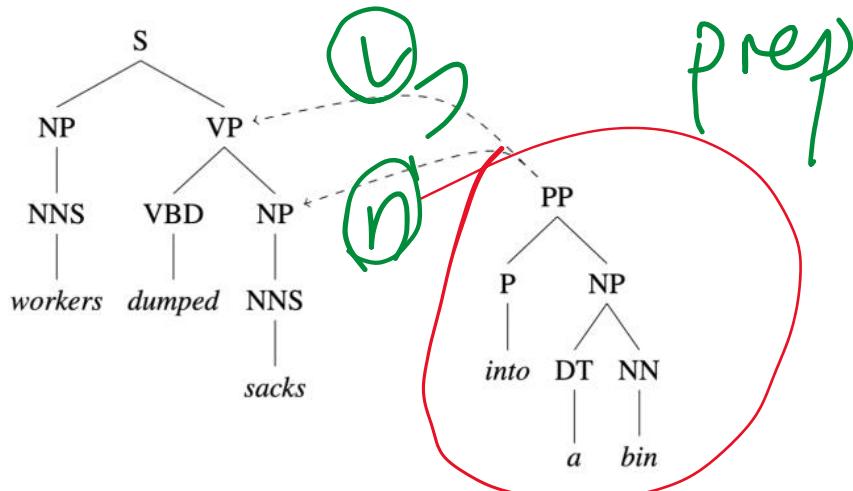


- NP^AS represents subject position (left)
- NP^AVP denotes object position (right)

35

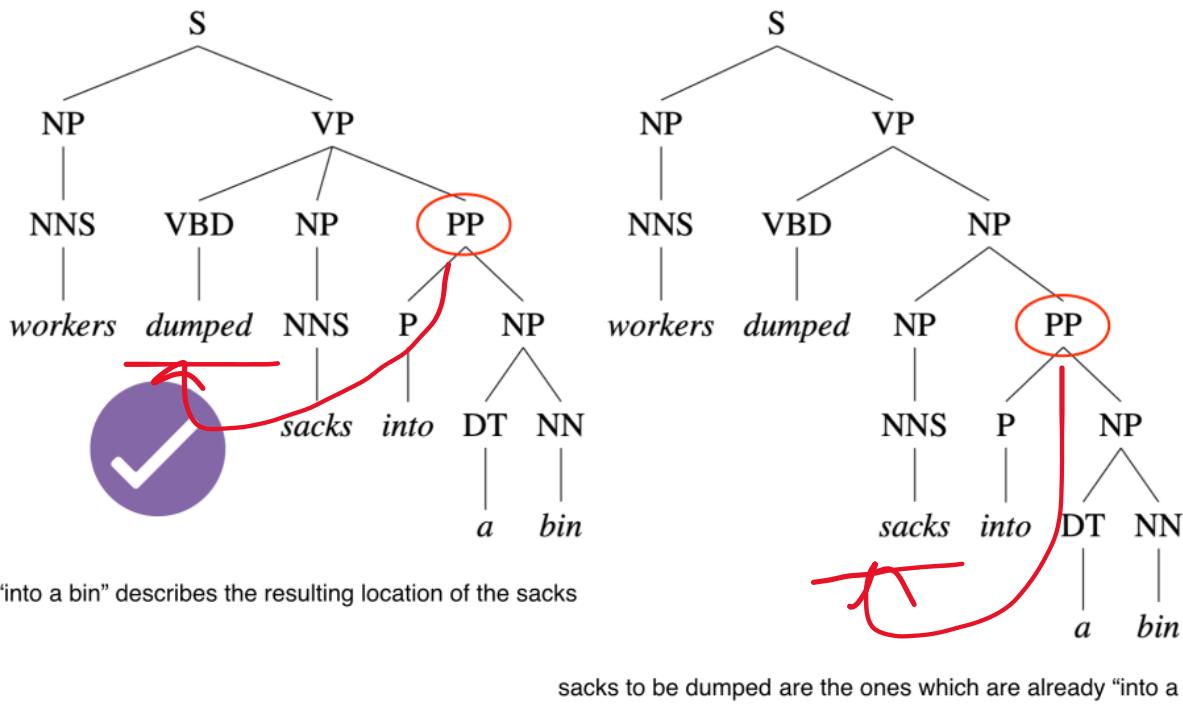
PCFG Problem 2: Lack of Lexical Conditioning

- Lack of sensitivity to words in tree
- Prepositional phrase (PP) attachment ambiguity
 - ▶ *Worker dumped sacks into a bin*



36

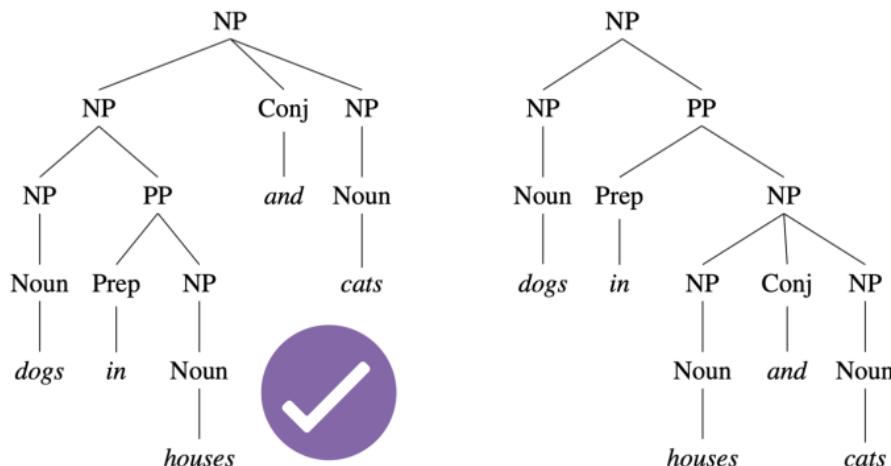
PP Attachment Ambiguity



37

Coordination Ambiguity

- *dogs in houses and cats*



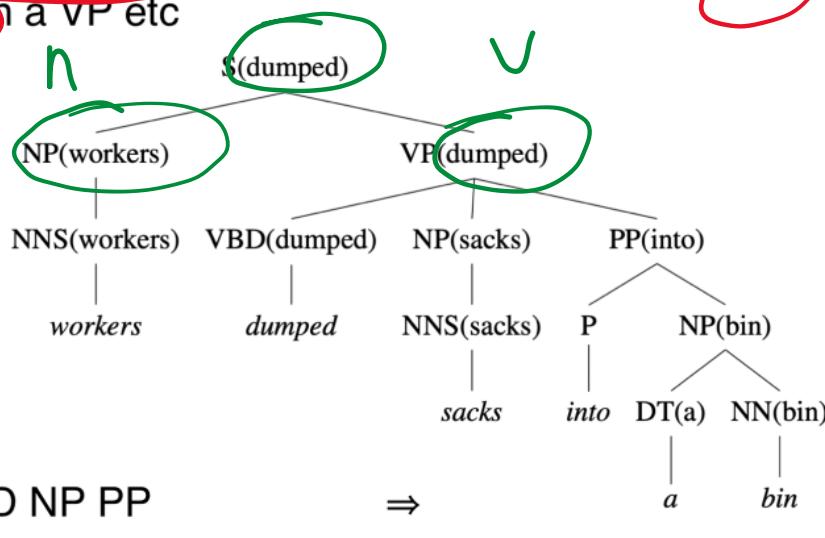
- *dogs* is semantically a better conjunct for *cats* than *houses* (*dogs* can't fit into *houses*!)

38

Solution: Head Lexicalisation

- Record head word with parent symbols

- the most salient child of a constituent, usually the noun in a NP, verb in a VP etc



- $VP \rightarrow VBD\ NP\ PP$

\Rightarrow

$VP(\text{dumped}) \rightarrow VBD(\text{dumped})\ NP(\text{sacks})\ PP(\text{into})$

39

Head Lexicalisation

- Incorporate head words into productions, such that the most important links between words is captured
 - rule captures correlations between head tokens of phrases
 - $VP(\text{dumped}) / NP(\text{sacks})$ for $PP(\text{into})$
- Grammar symbol inventory expands massively!
 - Many of the productions much too specific, seen very rarely
 - Learning more involved to avoid sparsity problems (e.g., zero probabilities)

40

A Final Word

- PCFGs widely used, and there are efficient parsers available.
 - ▶ Collins parser, Berkeley parser, Stanford parser
 - ▶ all use some form of lexicalisation
- But there are other grammar formalisms
 - ▶ Lexical function grammar
 - ▶ Head-driven phrase structure grammar
 - ▶ Next lecture: dependency grammar

41

Required Reading

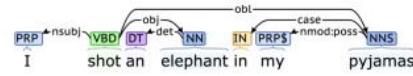
- J&M3 Ch. 14 – 14.6 (skip 14.6.1)

42



Dependency Grammar

COMP90042
Natural Language Processing
Lecture 16



1

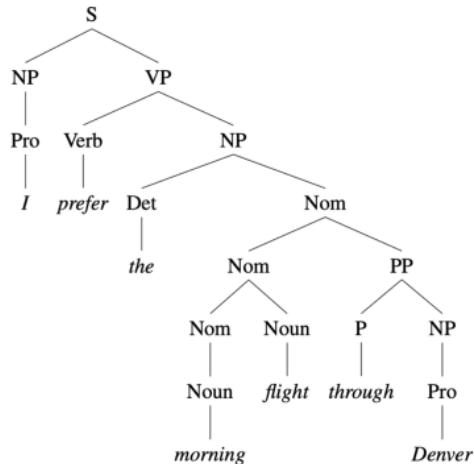
COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

Context-Free Grammars (Recap)

- CFGs assume a constituency tree which identifies the **phrases** in a sentence

- based on idea that **these phrases are interchangeable**

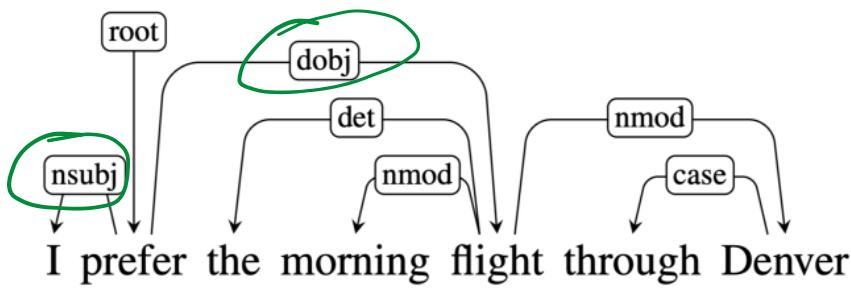
(e.g., swap an NP for another NP) and maintain grammaticality



3

Dependency Grammars

- Dependency grammar offers a simpler approach
 - describe relations between pairs of words
 - namely, between **heads** and **dependents**
 - e.g. (*prefer*, *dobj*, *flight*)



4

Why?

- Deal better with languages that are morphologically rich and have a relatively free word order
 - CFG need a separate rule for each possible place a phrase can occur in
- Head-dependent relations similar to semantic relations between words
 - More useful for applications: coreference resolution, information extraction, etc

5

Dependency Relations

- Captures the grammatical relation between a head and a dependent
- Head = central word
- Dependent = supporting word
- Grammatical relation = subject, direct object, etc
- Many dependency theories and taxonomies proposed for different languages
- Universal Dependency: a framework to create a set of dependency relations that are computationally useful and cross-lingual

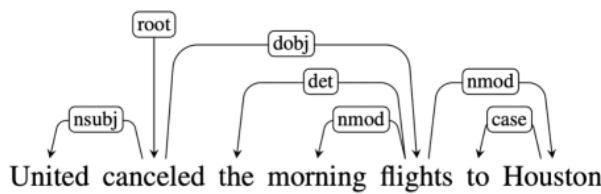
6

Universal Dependency

Clausal Argument Relations		Description
NSUBJ		Nominal subject
DOBJ		Direct object
IOBJ		Indirect object
CCOMP		Clausal complement
XCOMP		Open clausal complement

Nominal Modifier Relations		Description
NMOD		Nominal modifier
AMOD		Adjectival modifier
NUMMOD		Numeric modifier
APPOS		Appositional modifier
DET		Determiner
CASE		Prepositions, postpositions and other case markers

Other Notable Relations		Description
CONJ		Conjunct
CC		Coordinating conjunction



7

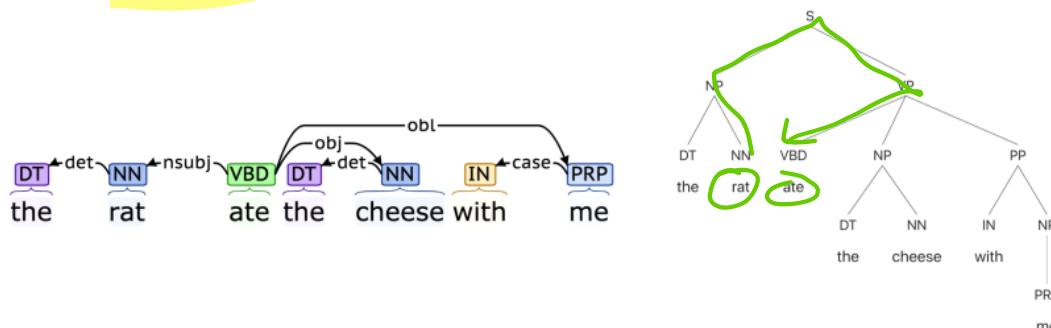
More Examples

Relation	Examples with head and dependent
NSUBJ	United canceled the flight.
DOBJ	United diverted the flight to Reno. We booked her the first flight to Miami.
IOBJ	We booked her the flight to Miami.
NMOD	We took the morning flight.
AMOD	Book the cheapest flight.
NUMMOD	Before the storm JetBlue canceled 1000 flights.
APPOS	United, a unit of UAL, matched the fares.
DET	The flight was canceled. Which flight was delayed?
CONJ	We flew to Denver and drove to Steamboat.
CC	We flew to Denver and drove to Steamboat.
CASE	Book the flight through Houston.

8

Question Answering

- Dependency tree more directly represents the core of the sentence: *who did what to whom?*
 - captured by the links incident on verb nodes

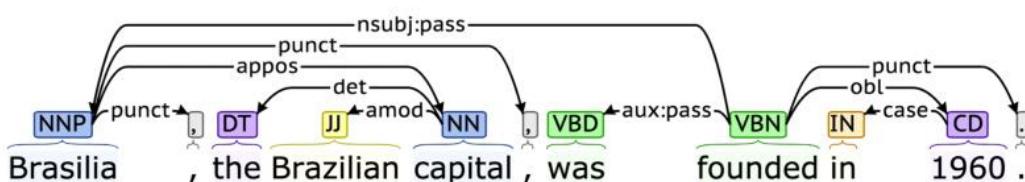


- more minor details are buried deeper in the tree (e.g., adjectives, determiners etc)

9

Information Extraction

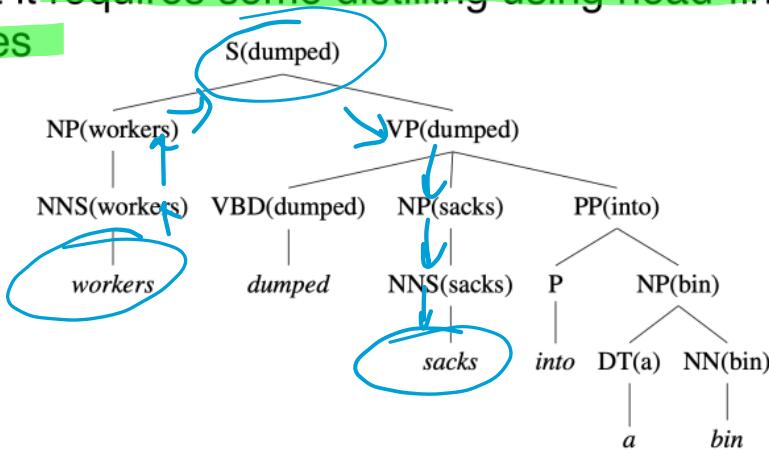
- “Brasilia, the Brazilian capital, was founded in 1960.”
 - capital(Brazil, Brasilia)
 - founded(Brasilia, 1960)
- Dependency tree captures relations succinctly



10

What about CFGs

- Constituency trees can also provide similar information
- But it requires some distilling using head-finding rules



11

Dependency vs Constituency

- Dependency tree**
 - each node is a word token
 - one node is chosen as the root
 - directed edges link heads and their dependents
- Constituency tree**
 - forms a hierarchical tree
 - word tokens are the leaves
 - internal nodes are 'constituent phrases' e.g., NP, VP etc
- Both use part-of-speech

12

Properties of a Dependency Tree

- Each word has a single head (parent)
- There is a single root node
- There is a unique path to each word from the root
- All arcs are **projective**
 - ▶ Transition-based parsers can only produce projective trees

*many outgoing links
but only one incoming link*

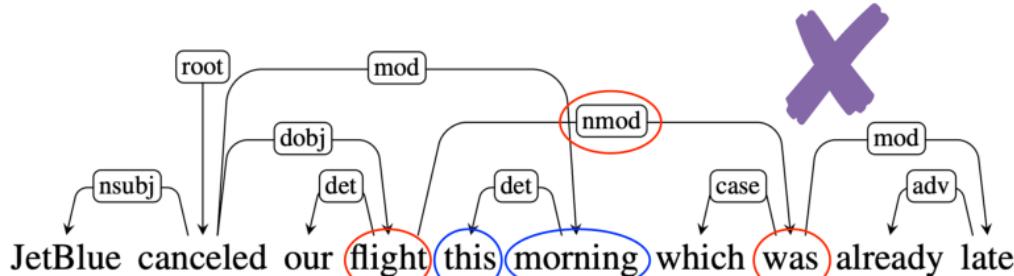
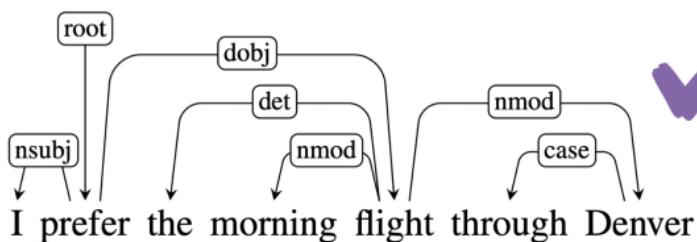
13

Projectivity

- An arc is projective if there is a path from head to every word that lies between the head and the dependent
- A dependency tree is projective if all arcs are projective
- That is, a dependency tree is projective if it can be drawn with no crossing edges
- Most sentences are projective, however exceptions exist
 - Common in languages with flexible word order

14

Projectivity



15

Dependency Treebanks

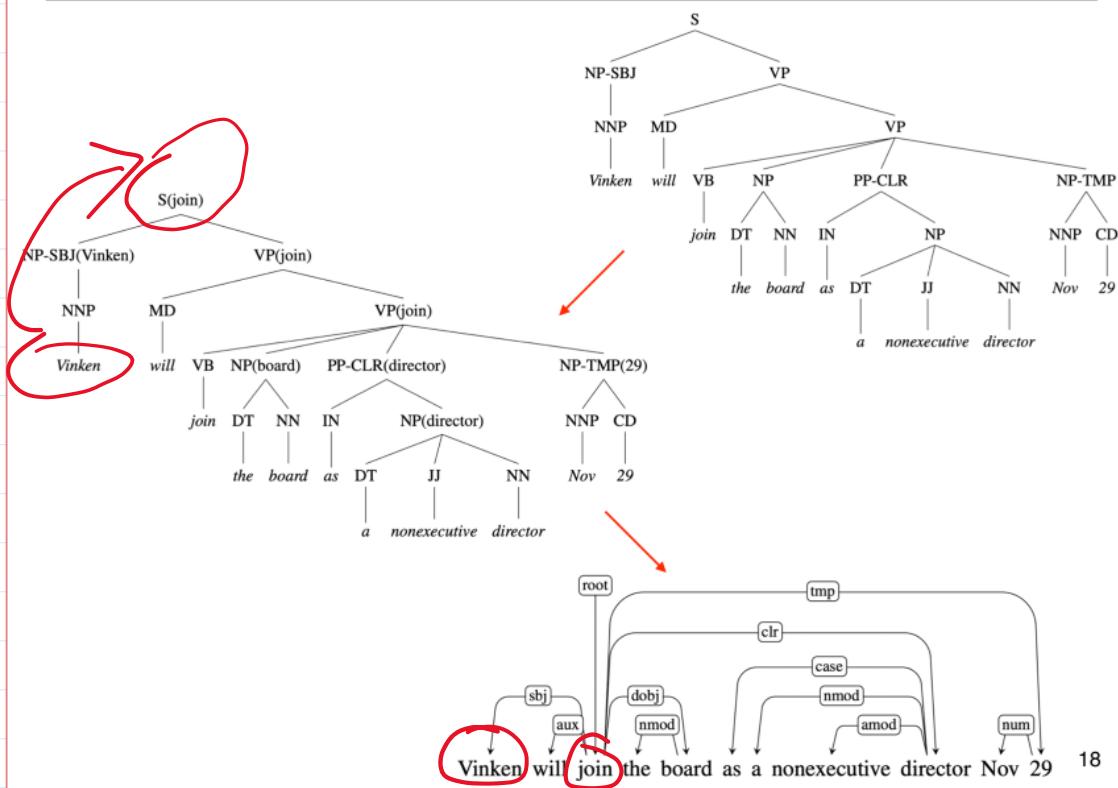
- A few dependency treebanks
 - Czech, Arabic, Danish, Dutch, Greek, Turkish ...
- More recently, **Universal Dependency Treebank**
 - collates >100 treebanks, >60 languages
 - unified part-of-speech, morphology labels, relation types
 - consistent handling of conjunctions and other tricky cases
- <http://universaldependencies.org/>

16

Treebank Conversion

- Many constituency treebanks; some can be automatically converted into dependencies
- Dependency trees generated from constituency trees are always projective**
- Main idea: identify head-dependent relations in constituency structure and the corresponding dependency relations
 - Use various heuristics, e.g., head-finding rules
 - often with manual correction

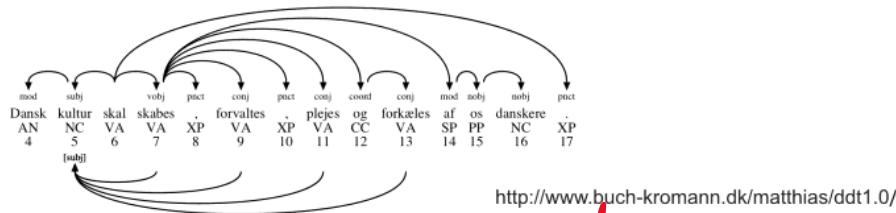
17



18

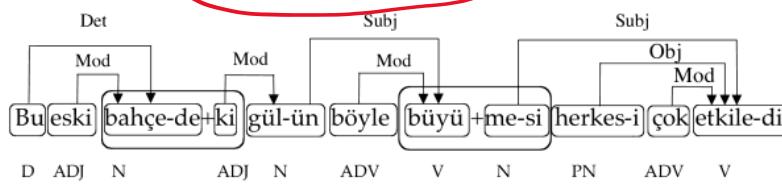
Examples From Treebanks

- Danish DDT includes additional 'subject' link for verbs



- METU-Sabancı Turkish treebank *Subwords*

- edges between morphological units, not just words



Oflazer, Kemal, et al. "Building a Turkish treebank." *Treebanks*. Springer, 2003. 261-277.

19

Parsing

20

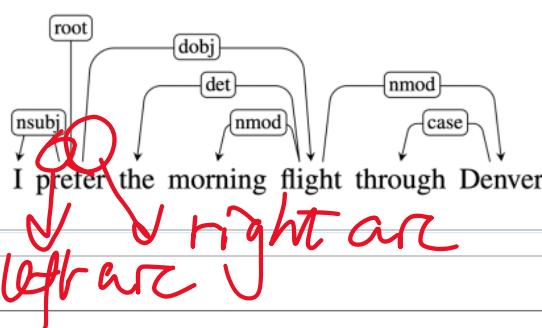
Dependency Parsing

- Parsing: task of finding the best structure for a given input sentence
 - $\operatorname{argmax}_t \operatorname{score}(t | w)$
- Two main approaches:
 - transition-based:** treats problem as incremental sequence of decisions over next action in a state machine
 - graph-based:** encodes problem using nodes/edges and use graph theory methods to find optimal solutions

21

Transition-Based Parsing: Intuition

- Examine the words in a single pass from left to right
- At each step, perform one of the following actions:
 - Assign current word as head of some previously seen words **[left arc]**
 - Assign some previously seen words as head of current word **[right arc]**
 - Don't do anything, store it and move to next step **[shift]**



22

Transition-Based Parsing

- Transition-based parsers can only produce **projective trees**

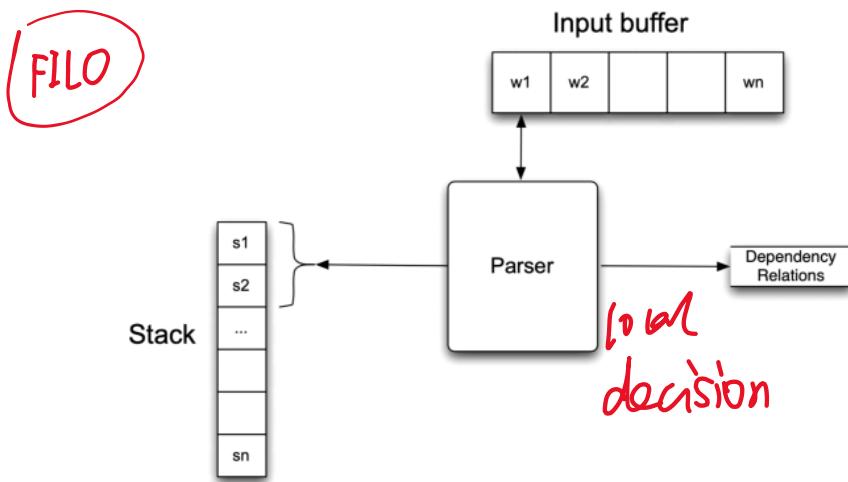
leftarc U

Transition-Based Parsing

- Transition-based parsers can only produce projective trees
- Frames parsing as sequence of transitions
 - ▶ maintain two data structures
 - *buffer*: input words yet to be processed
 - *stack*: head words currently being processed
 - ▶ two types of transitions
 - *shift*: move word from buffer to top of stack
 - *arc*: add arc (left/right) between top two items on stack, and remove dependent from stack

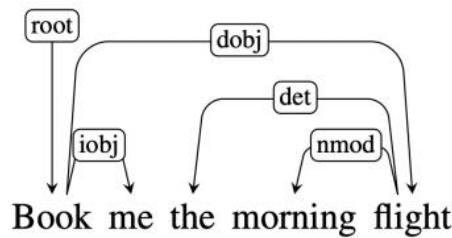
23

Transition-Based Parsing



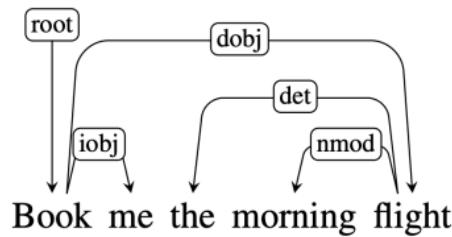
J&M Fig 15.5

24



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	

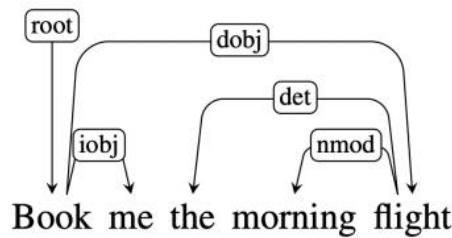
25



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

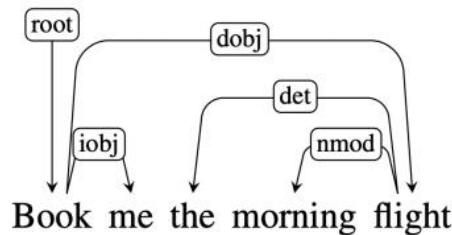
↑
bottom ↑P

26



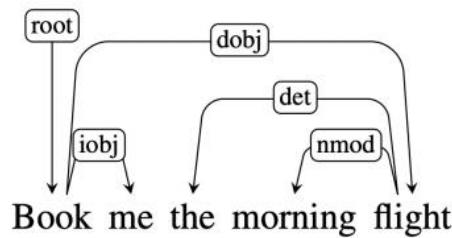
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)

27



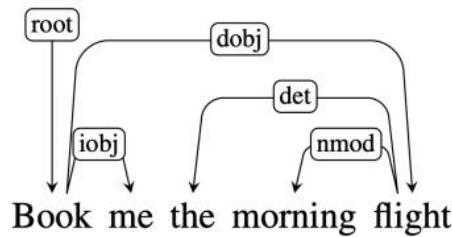
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	

28



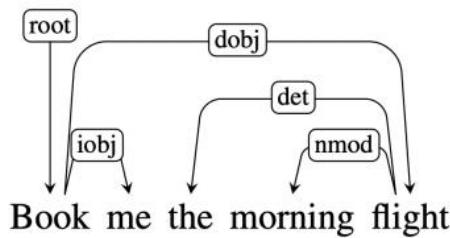
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)

29



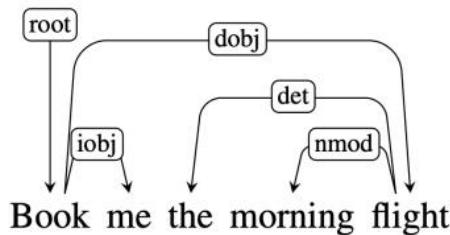
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)

30



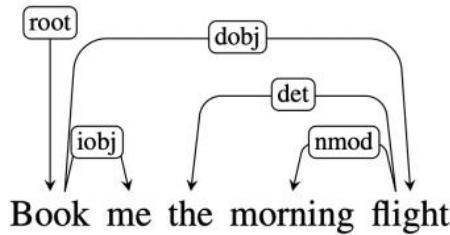
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)

31



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)

32



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

33

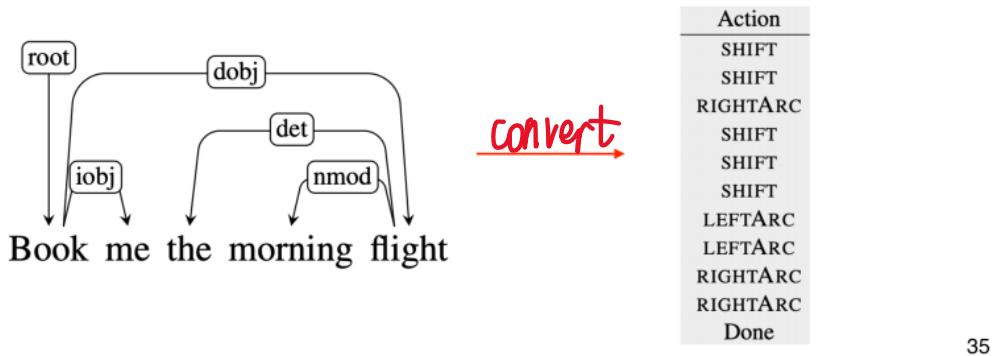
Dependency Labels

- For simplicity, we omit labels on the dependency relations
- In practice, we parameterise the left-arc and right-arc actions with dependency labels:
 - E.g. left-arc-nsubj or right-arc-dobj
- This expands the list of actions to more than just 3 types

34

The Right Action?

- We assume an **oracle** that tells us the right action at every step
- Given a dependency tree, the role of oracle is to generate a sequence of ground truth actions



35

Parsing Model

- We then train a supervised model to **mimic the actions of the oracle**
 - ▶ To learn at every step the correct action to take (given by the oracle)
 - ▶ At test time, the trained model can be used to **parse a sentence to create the dependency tree**

36

Parsing As Classification

- Input:
 - ▶ Stack (top-2 elements: s_1 and s_2)
 - ▶ Buffer (first element: b_1)
- Output
 - ▶ 3 classes: *shift*, *left-arc*, or, *right-arc*
- Features
 - ▶ word (w), part-of-speech (t)

37

Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled → United) (flights → morning) (flights → the)

- Input features:

- ▶ $s_1.w = \underline{\text{flights}}$
- ▶ $s_2.w = \underline{\text{cancelled}}$
- ▶ $s_1.t = \text{NNS}$
- ▶ $s_2.t = \text{VBD}$
- ▶ $b_1.w = \underline{\text{to}}$
- ▶ $b_1.t = \text{TO}$
- ▶ $s_1.t \circ s_2.t = \text{NNS_VBD}$

Source	Feature templates	
One word		
$s_1.w$	$s_1.t$	$s_1.wt$
$s_2.w$	$s_2.t$	$s_2.wt$
$b_1.w$	$b_1.w$	$b_0.wt$
Two word		
$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

- Output label: shift

38

Classifiers

- Traditionally SVM works best
- Nowadays, deep learning models are the state-of-the-art
- Weakness: local classifier based on greedy search
- Solutions:
 - Beam search: keep track of top-N best actions
(next lecture!)
 - Dynamic oracle: during training, use predicted actions occasionally

39

Graph-Based Parsing

- Given an input sentence, construct a fully-connected, weighted, directed graph
- Vertices: all words
- Edges: head-dependent arcs
- Weight: score based on training data (relation that is frequently observed receive a higher score)
- Objective: find the maximum spanning tree

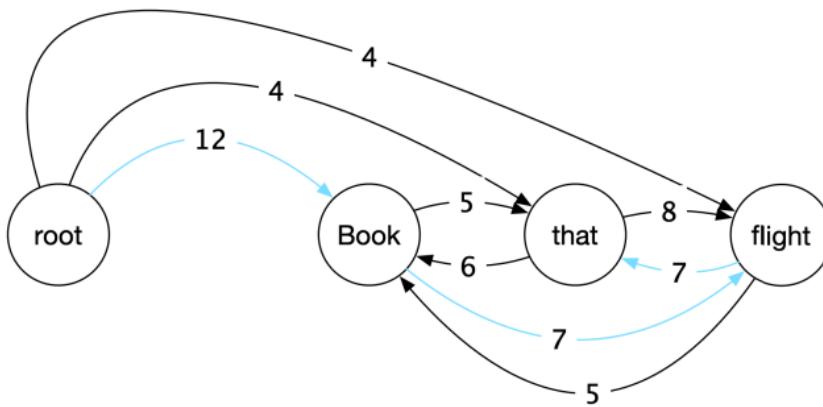
40

Advantage

- Can produce non-projective trees
 - ▶ Not a big deal for English
 - ▶ But a problem for many other languages
- Score entire trees
 - ▶ Avoid making greedy local decisions like transition-based parsers
 - ▶ Captures long dependencies better

41

Example



- Caveat: tree may contain cycles
- Solution: need to do cleanup to remove cycles
(Chu-Liu-Edmonds algorithm)

42

A Final Word

- Dependency parsing a compelling, alternative, formulation to constituency parsing
 - ▶ structures based on words as internal nodes
 - ▶ edges encode word-word syntactic and semantic relations
 - ▶ often this is the information we need for other tasks!
- Transition-based parsing
 - ▶ a sequence of shift and left/right-arc actions
- Graph-based parsing
 - ▶ frames it as a maximum spanning tree problem

43

Required Reading

- J&M3 Ch. 15

44



Machine Translation

COMP90042

Natural Language Processing

Lecture 17



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L17

Introduction

- **Machine translation (MT)** is the task of translating text from one **source language** to another **target language**

虽然北风呼啸，但天空依然十分清澈



However, the sky remained clear under the strong north wind

2

Why?

- Removes language barrier
- Makes information in any languages accessible to anyone
- But translation is a classic “AI-hard” challenge
 - ▶ Difficult to preserve the meaning and the fluency of the text after translation

3

MT is Difficult

- Not just simple word for word translation
- Structural changes, e.g., syntax and semantics
- Multiple word translations, idioms
- Inflections for gender, case etc
- Missing information (e.g., determiners)

虽然 北 风 呼啸 , 但 天空 依然 十分 清澈
although north wind howls , but sky still extremely limpid

4

Outline

- Statistical Machine Translation
 - ▶ Word-based approach
- Neural Machine Translation
 - ▶ Encoder-decoder architecture
- Attention Mechanism
 - ▶ Improvement to Neural MT
- Machine Translation Evaluation

5

Statistical MT

6

Early Machine Translation

- Started in early 1950s
- Motivated by the Cold War to translate Russian to English
- Rule-based system
 - ▶ Use bilingual dictionary to map Russian words to English words
- Goal: translate 1-2 million words an hour within 5 years

7

Statistical MT

- Given French sentence f , aim is to find the best English sentence e
 - ▶ $\operatorname{argmax}_e P(e | f)$
- Use Baye's rule to decompose into two components
 - ▶ $\operatorname{argmax}_e P(f | e)P(e)$

$$P(f | e)$$

ignore

8

Language vs Translation Model

- $\operatorname{argmax}_e P(f|e)P(e)$
- $P(e)$: language model
 - ▶ learns how to write fluent English text
- $P(f|e)$: translation model
 - ▶ learns how to translate words and phrases from English to French

9

How to Learn LM and TM?

- Language model:
 - ▶ Based on text frequency in large monolingual corpora
- Translation model:
 - ▶ Based on word co-occurrences in parallel corpora
 - ▶ i.e. English-French sentence pairs

10

Parallel Corpora

- One text in multiple languages
- Produced by human translation
 - ▶ Bible, news articles, legal transcripts, literature, subtitles
 - ▶ Open parallel corpus:
<http://opus.nlpl.eu/>



The Rosetta Stone

11

Models of Translation

- How to learn $P(f|e)$ from parallel text?
- We only have sentence pairs; words are not aligned in the parallel text
- Not word to word translation:
 - ▶ rearrangement of words

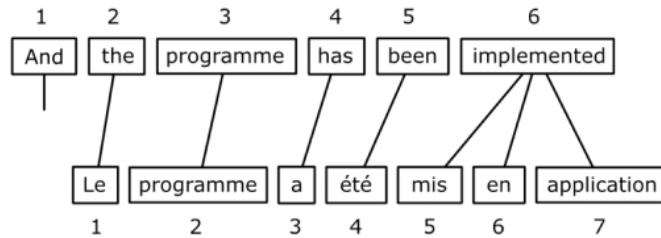
他 是 该 加 税 计 划 的 少 数 支 持 者 之 一
 He is one of the few supporters of this tax increase plan

Zhang, Liu, Li, Zhou, Zong (2016) 12

Alignment

- Idea: introduce word alignment as a latent variable into the model

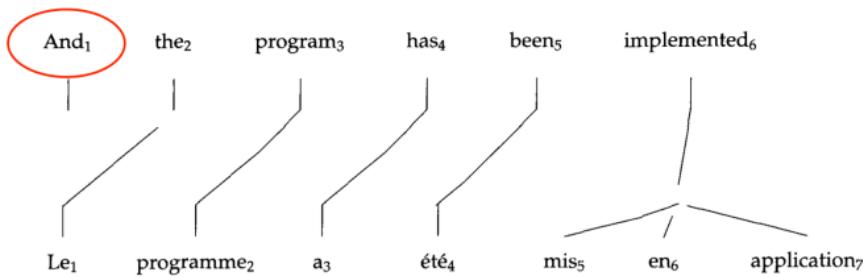
$$\triangleright P(f, a | e)$$



https://en.wikipedia.org/wiki/Bitext_word_alignment#/media/File:Word_alignment.svg 13

Complexity of Alignment

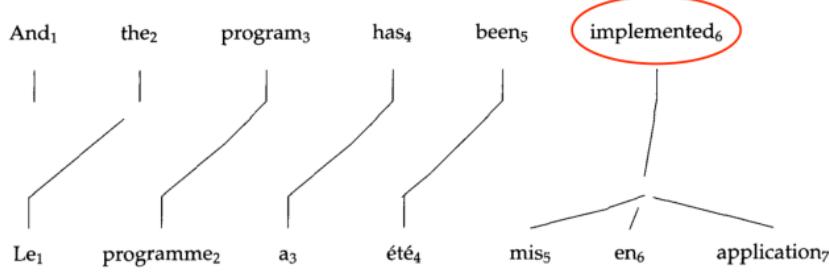
- Some words are dropped and have no alignment



Brown, Della Pietra, Della Pietra, Mercer, 1993 14

Complexity of Alignment

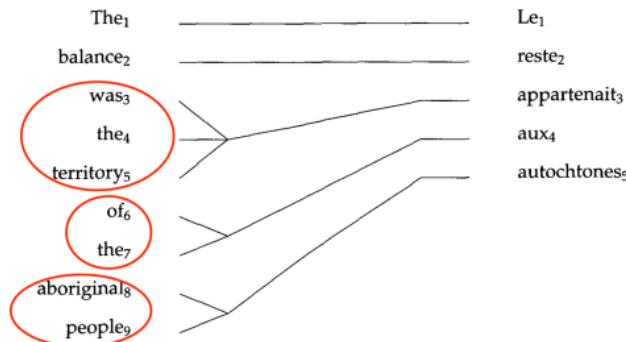
- One-to-many alignment



Brown, Della Pietra, Della Pietra, Mercer, 1993 15

Complexity of Alignment

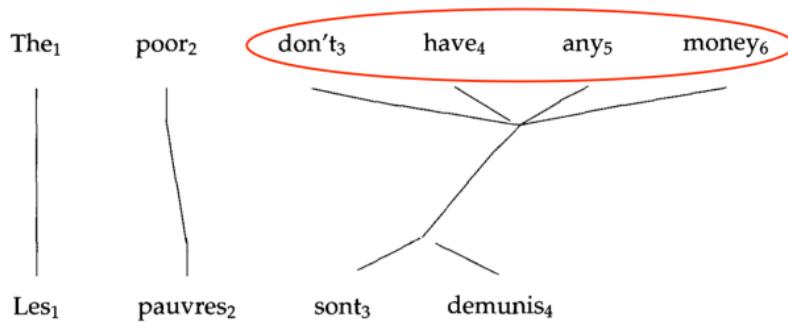
- Many-to-one alignment



Brown, Della Pietra, Della Pietra, Mercer, 1993 16

Complexity of Alignment

- Many-to-many alignment



Brown, Della Pietra, Della Pietra, Mercer, 1993 17

Learning the Alignment

- Word alignments are rarely provided in parallel corpora
 - ▶ As such alignment is introduced as a latent variable
- Use algorithms such as expectation maximisation (EM) to learn

Statistical MT: Summary

- A very popular field of research in NLP prior to 2010s
- Lots of feature engineering
- State-of-the-art systems are very complex
 - ▶ Difficult to maintain
 - ▶ Significant effort needed for new language pairs

19

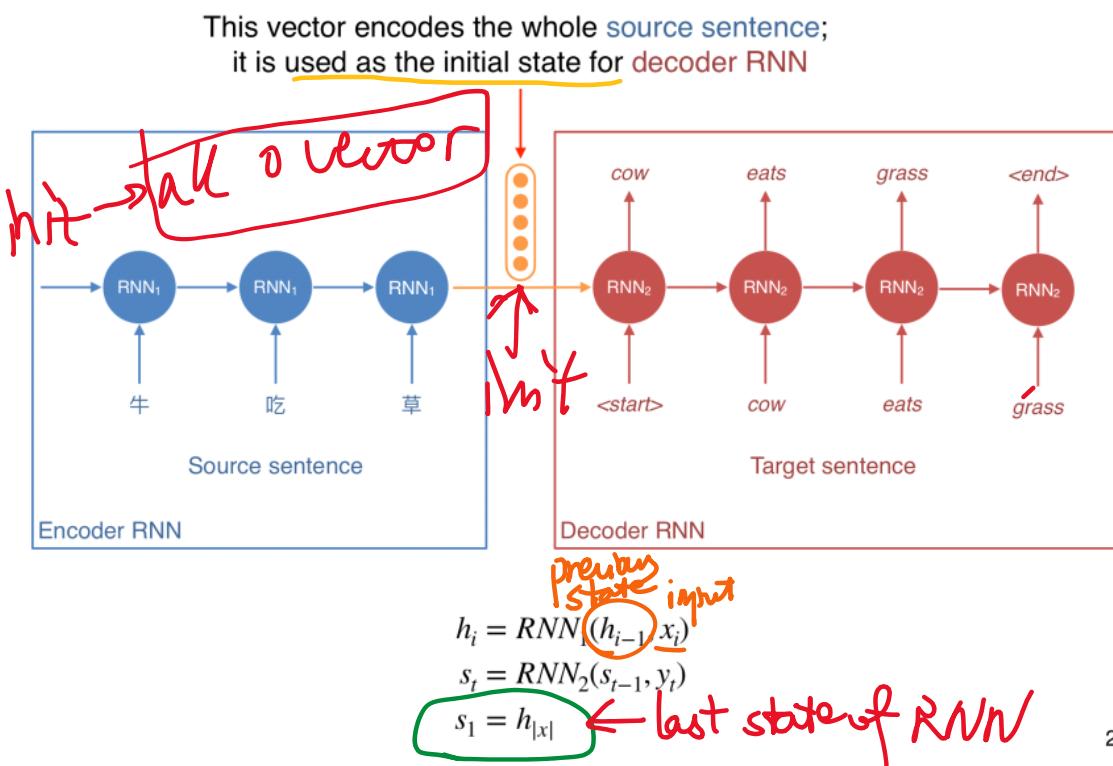
Neural Machine Translation

20

Introduction

- Neural machine translation is a new approach to do machine translation
- Use a single neural model to directly translate from source to target
- Architecture: encoder-decoder model
 - ▶ 1st RNN to encode the source sentence
 - ▶ 2nd RNN to decode the target sentence

21



22

Neural MT

- The decoder RNN can be interpreted as a **conditional language model**
 - ▶ Language model: predicts the next word given previous words in target sentence y
 - ▶ Conditional: prediction is also conditioned on the source sentence x
- $P(y|x) = P(y_1|x)P(y_2|y_1, x)\dots P(y_t|y_1, \dots, y_{t-1})$

$\underbrace{P(y_1|x)P(y_2|y_1, x)\dots P(y_t|y_1, \dots, y_{t-1})}_{\text{previous}}$
source sentence

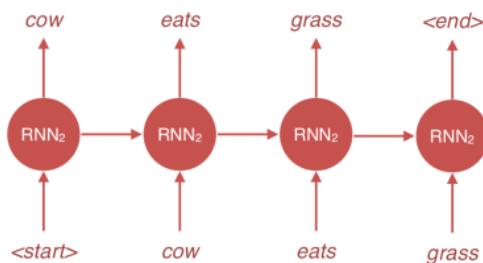
Training Neural MT

- Requires parallel corpus just like statistical MT
- Trains with next word prediction, just like a language model!

Language Model Training Loss

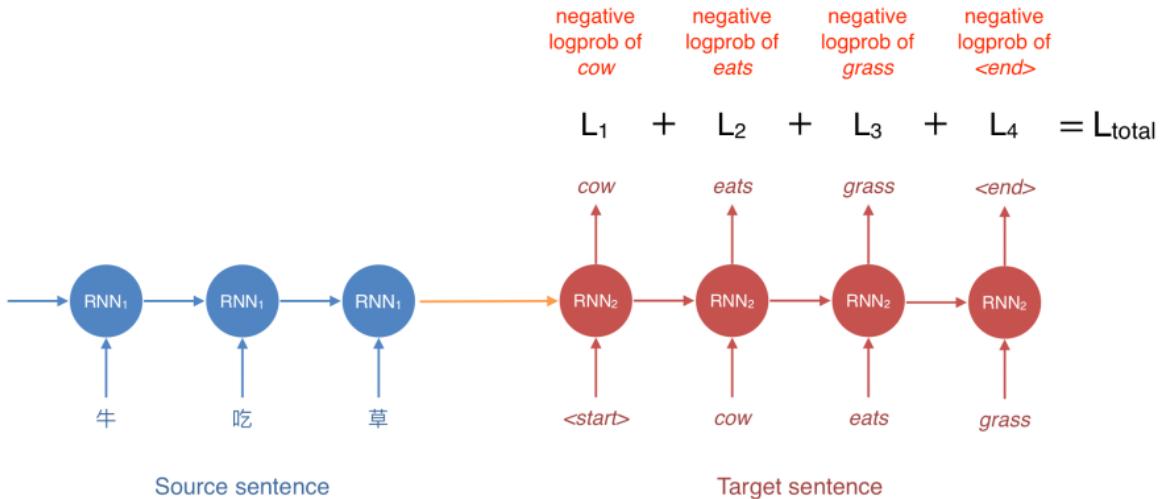
cross entropy loss

$$\text{negative logprob of } \text{cow} + \text{negative logprob of } \text{eats} + \text{negative logprob of } \text{grass} + \text{negative logprob of } \langle \text{end} \rangle = L_{\text{total}}$$



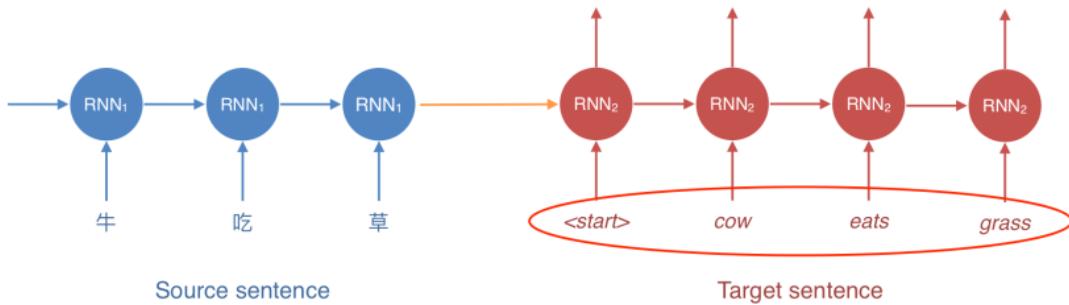
25

Neural MT Training Loss



26

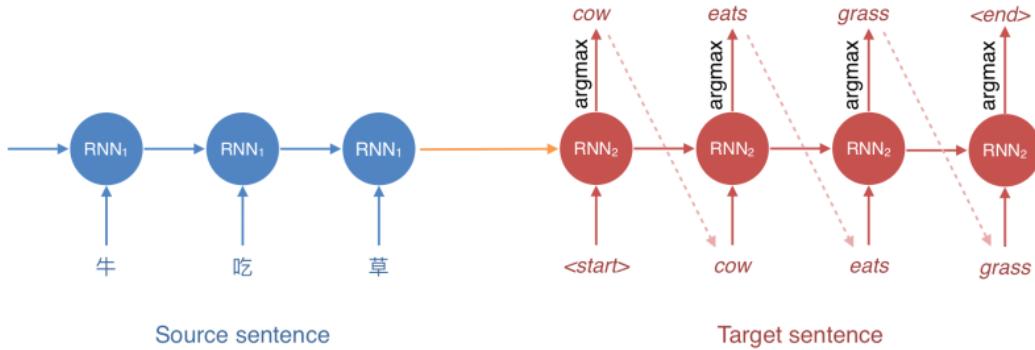
Training



- During training, we have the target sentence
- We can therefore feed the right word from target sentence, one step at a time

27

Decoding at Test Time



- But at test time, we don't have the target sentence (that's what we're trying to predict!)
- argmax: take the word with the highest probability at every step

28

exposure bias problem!
→ generate wrong word, never recover
Greedy Decoding

- argmax decoding is also called greedy decoding

- argmax decoding is also called **greedy decoding**
- Issue: **does not guarantee optimal probability**
 $P(y|x)$

29

COMP90042

L17

Exhaustive Search Decoding

- To find optimal $P(y|x)$, we need to **consider every word at every step to compute the probability of all possible sequences**
- $O(V^n)$; V = vocab size; n = sentence length
- **Far too expensive to be feasible**

30

Beam Search Decoding

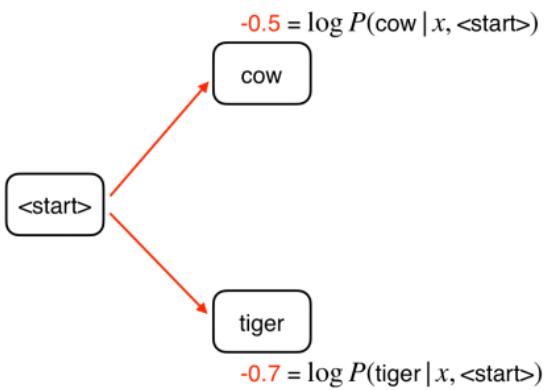
- Instead of considering all possible words at every step, **consider k best words**
- That is, **we keep track of the top-k words that produce the best partial translations (hypotheses)** thus far
- **k = beam width (typically 5 to 10)**
- **k = 1 = greedy decoding**
- **k = V = exhaustive search decoding**

31

<start>

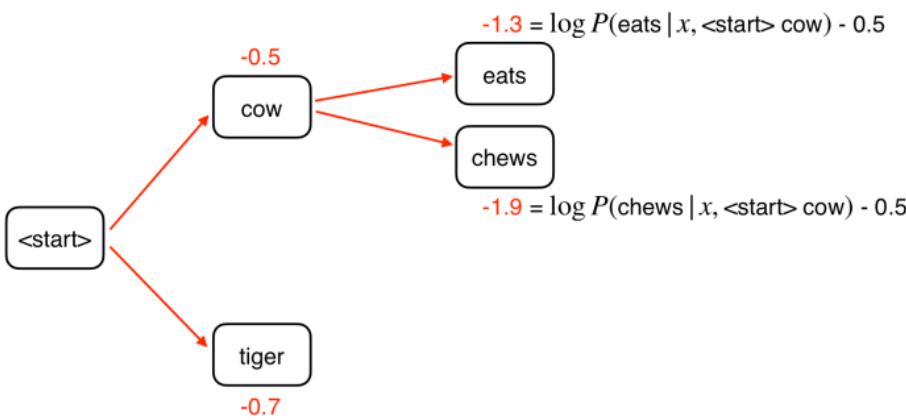
Beam search decoding with k=2

32



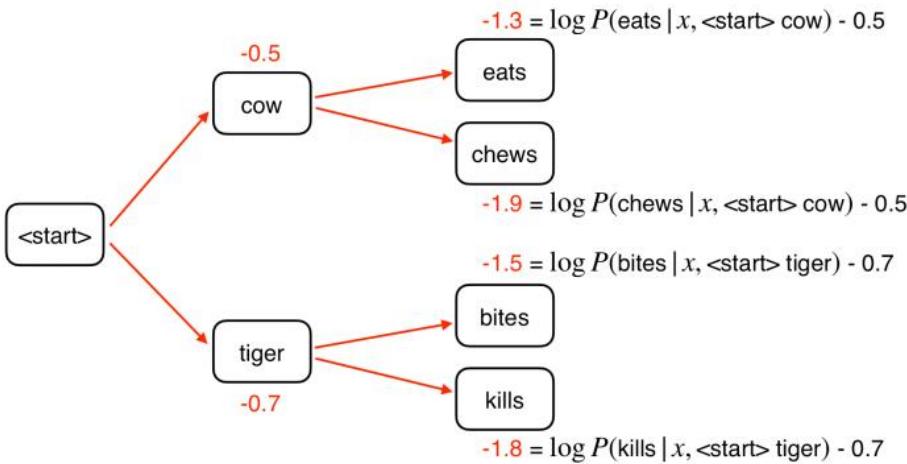
Beam search decoding with k=2

33

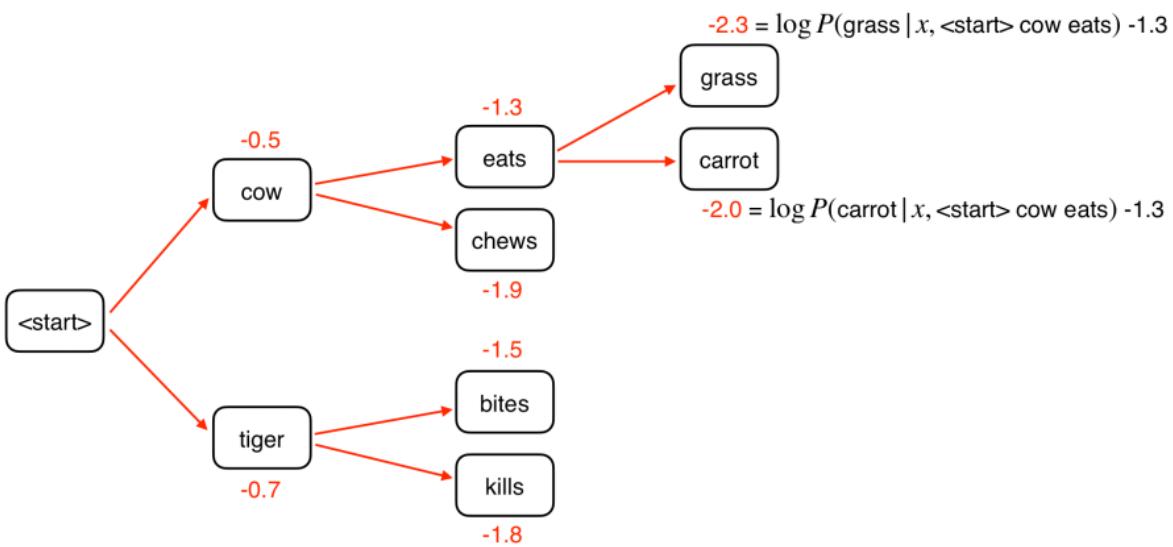


Beam search decoding with k=2

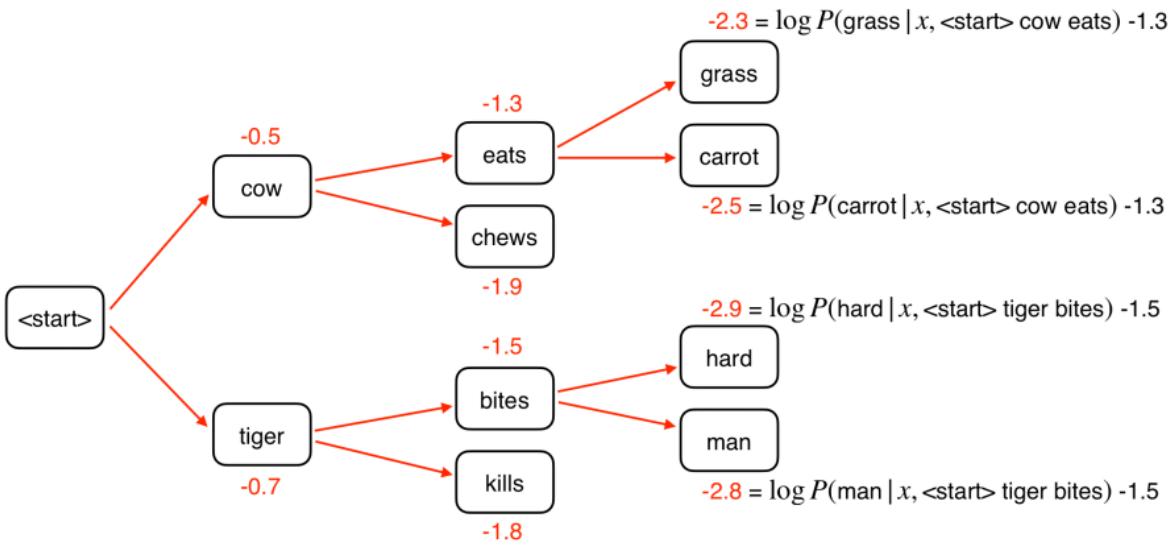
34

Beam search decoding with $k=2$

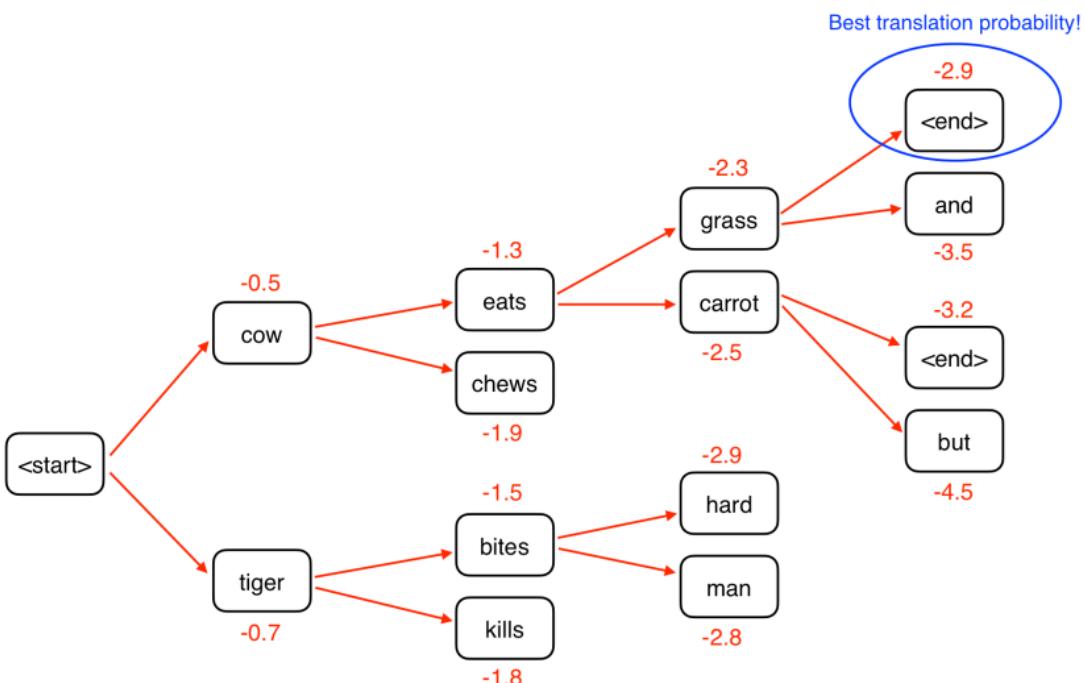
35

Beam search decoding with $k=2$

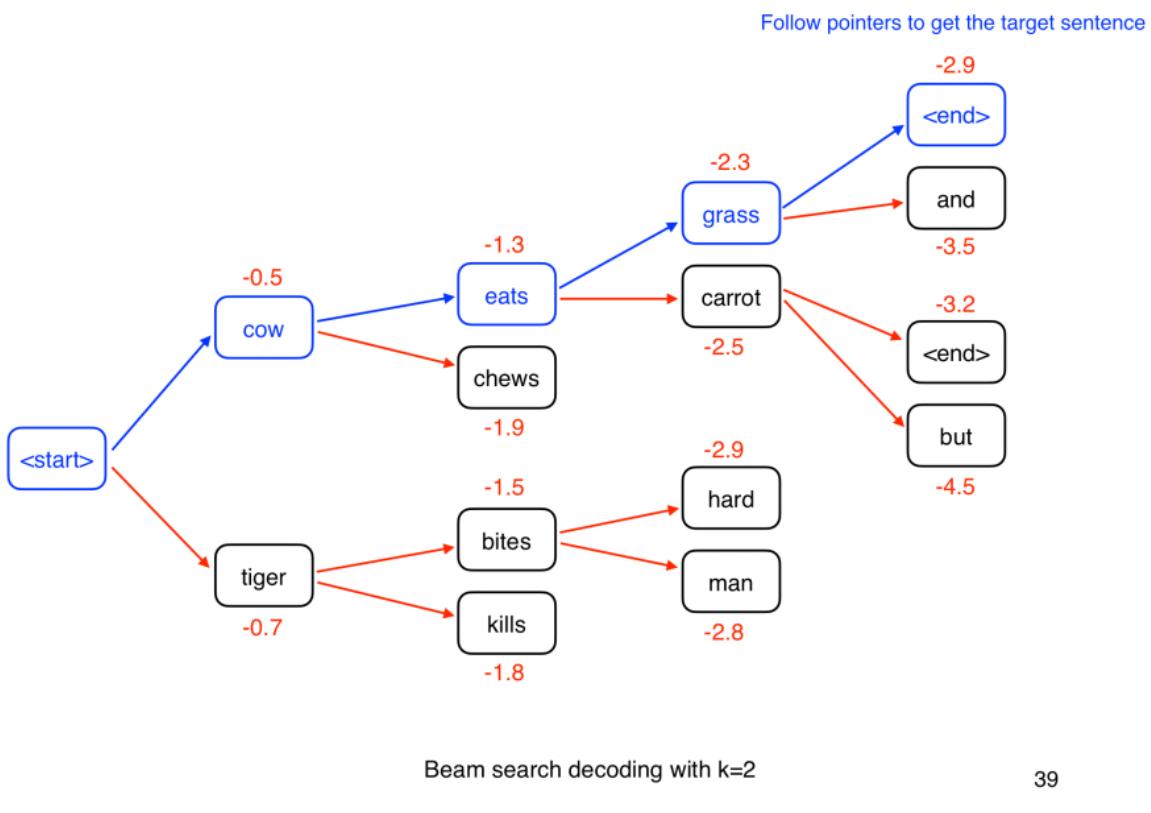
36

Beam search decoding with $k=2$

37

Beam search decoding with $k=2$

38



When to Stop?

- When decoding, we stop when we generate $\langle \text{end} \rangle$ token
- But multiple hypotheses may terminate their sentence at different time steps
- We store hypotheses that have terminated, and continue explore those that haven't
- Typically we also set a maximum sentence length that can be generated (e.g. 50 words)

Neural MT: Summary

- Single end-to-end model
 - Statistical MT systems have multiple sub-components
- Less feature engineering
- Generated translation is more fluent
- But can produce new details that are not in the source sentence (hallucination)

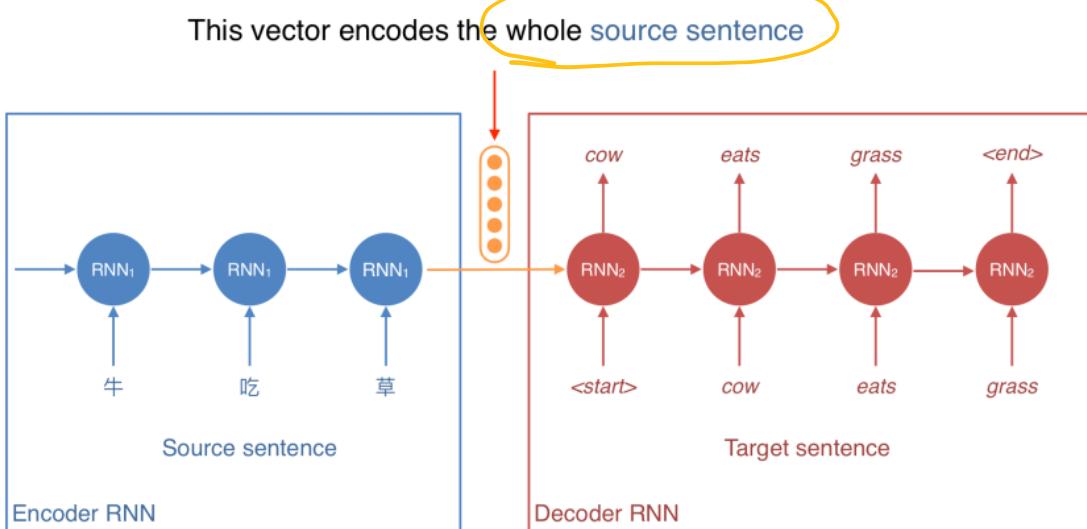
Source: Caldiero sprach mit E! Nachrichten nach dem hart erkämpften Sieg, noch immer unter dem Schock über den Gewinn des Großen Preises von 1 Million \$.

Reference: Caldiero spoke with E! News after the hard-fought victory, still in shock about winning the \$1 million grand prize.

NMT Translation: Caldiero spoke with E, after the hard won victory, still under the shock of the winning of the Grand Prix of 1 million \$.

Lee, Firat, Agarwal, Fannjiang, Sussillo, 2018 41

Attention (!)



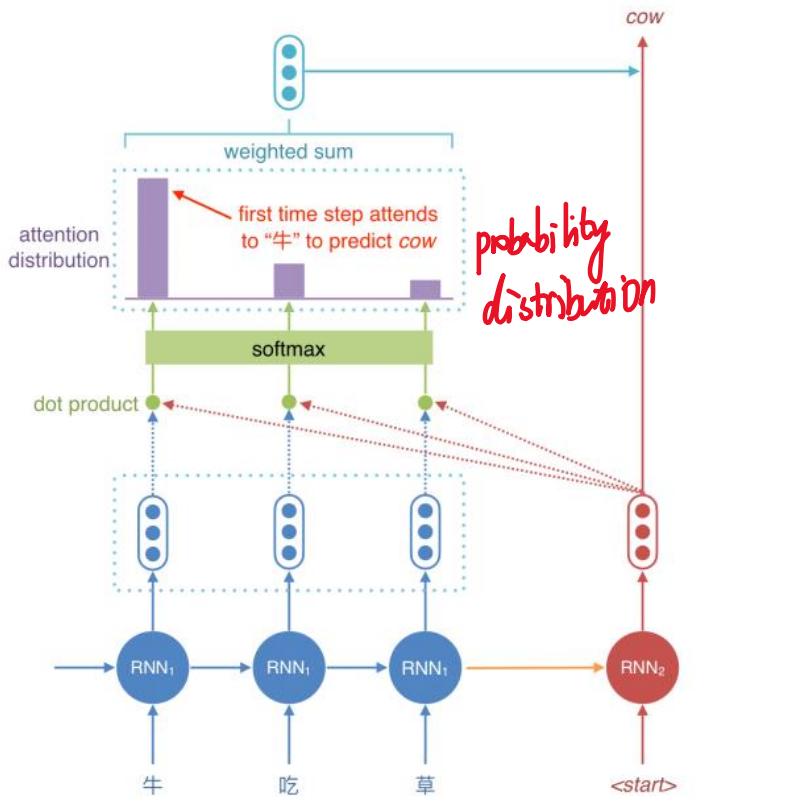
- With a long source sentence, the encoded vector is unlikely to capture all the information in the sentence
- This creates an **information bottleneck**

43

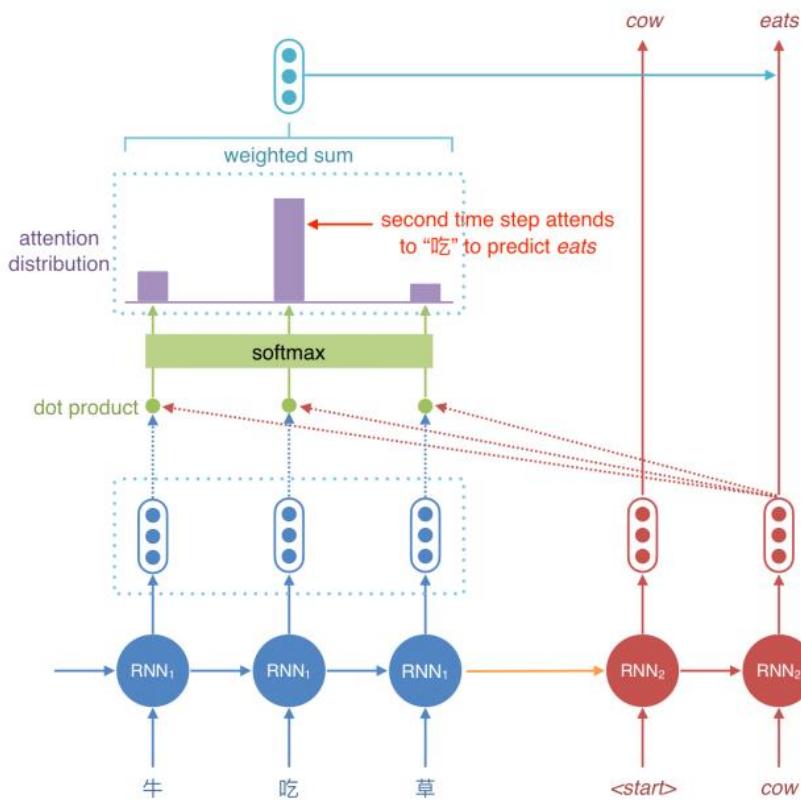
Attention

- For the decoder, at every time step allow it to ‘attend’ to words in the source sentence

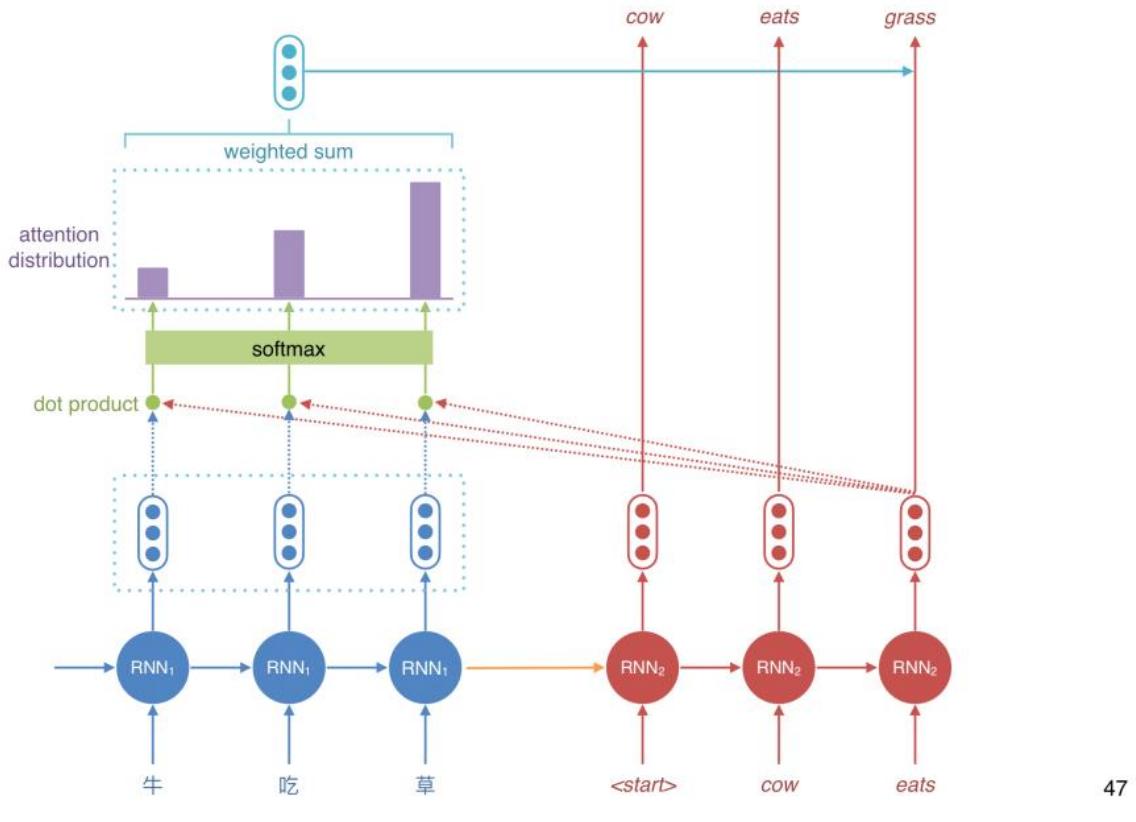
44



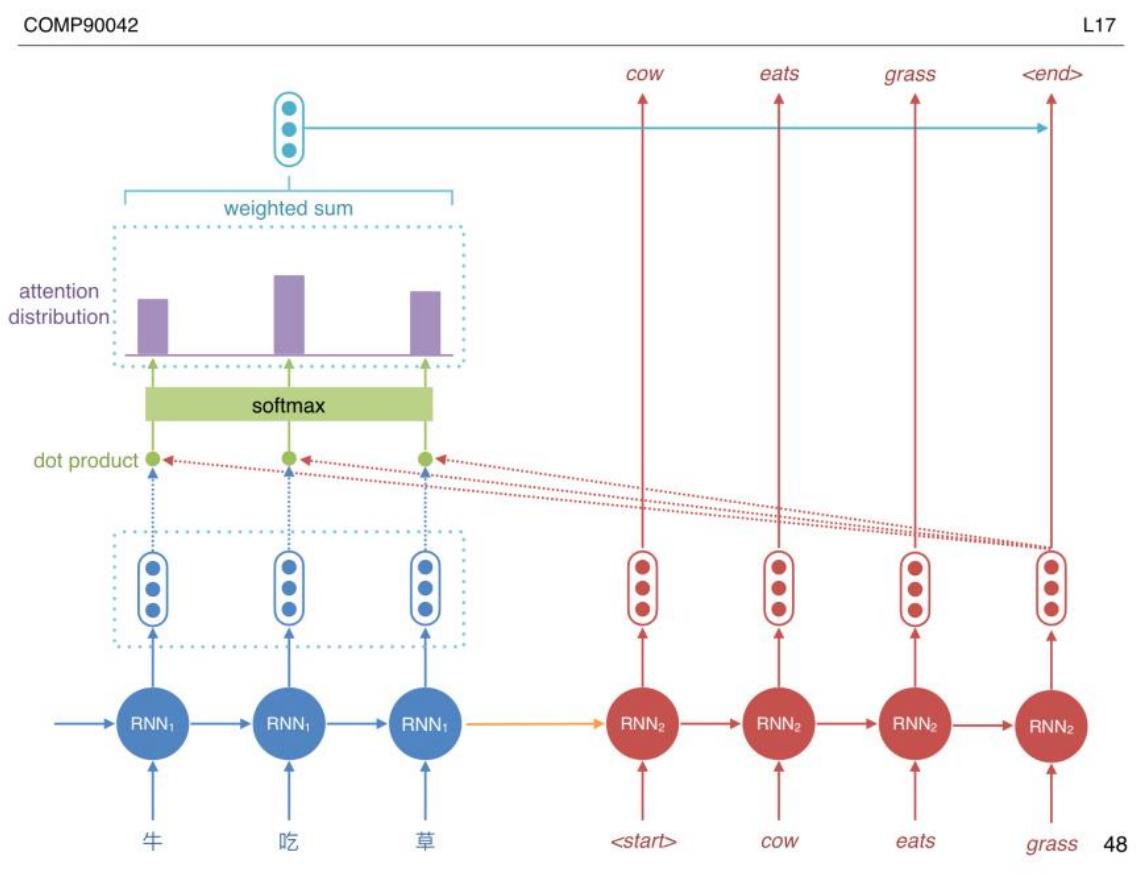
45



46



47

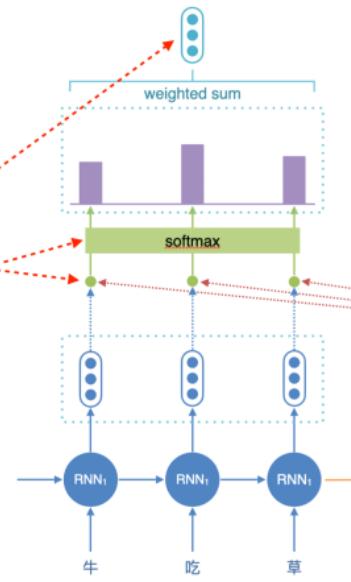


48

Encoder-Decoder with Attention

- Encoder hidden states: $h_i = RNN_1(h_{i-1}, x_i)$
- Decoder hidden states: $s_t = RNN_2(s_{t-1}, y_t)$
- For each time step in the decoder, attend to each of the hidden states to produce the attention weights:
 - $e_t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_{|x|}]$
- Apply softmax to the attention weights to get a valid probability distribution:
 - $\alpha_t = \text{softmax}(e_t)$
- Compute weighted sum of the encoder hidden states:
 - $c_t = \sum_{i=1}^{|x|} \alpha_t^i h_i$
- Concatenate c_t and s_t to predict the next word

context vector



49

Variants

- Attention:
 - Dot product: $s_t^T h_i$
 - Bilinear: $s_t^T W h_i$
 - Additive: $v^T \tanh(W_s s_t + W_h h_i)$
- c_t can be injected to the current state (s_t), or to the input word (y_t)

50

Attention: Summary

- Solves the information bottleneck issue by allowing decoder to have access to the source sentence words directly
- Provides some form of interpretability
 - ▶ Attention weights can be seen as word alignments
- Most state-of-the-art MT systems are based on this architecture
 - ▶ Google Translate (<https://slator.com/technology/google-facebook-amazon-neural-machine-translation-just-had-its-busiest-month-ever/>)

51

Evaluation

52

MT Evaluation

- BLEU: compute n-gram overlap between "reference" translation and generated translation
- Typically computed for 1 to 4-gram

$$\text{BLEU} = \text{BP} \times \exp \left(\frac{1}{N} \sum_n^N \log p_n \right)$$

"Brevity Penalty" to penalise short outputs

$$p_n = \frac{\# \text{ correct n-grams}}{\# \text{ predicted n-grams}}$$

$$\text{BP} = \min \left(1, \frac{\text{output length}}{\text{reference length}} \right)$$

Precision-based

53

A Final Word

- Statistical MT
- Neural MT
- Encoder-decoder with attention architecture is a general architecture that can be used for other tasks
 - ▶ Summarisation (lecture 21)
 - ▶ Dialogue generation

54

Reading

- https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf (Section 1-5)



I18-information-ext...

Information Extraction

COMP90042

Natural Language Processing

Lecture 18



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

Information Extraction

- Given this:
 - ▶ “*Brasilia, the Brazilian capital, was founded in 1960.*”
- Obtain this:
 - ▶ capital(Brazil, Brasilia)
 - ▶ founded(Brasilia, 1960)
- Main goal: turn **text** into **structured data** such as databases, etc.
- Help **decision makers** in applications.

2

Examples

- Stock analysis
 - ▶ Gather information from news and social media → summarise into a **structured format** → decide whether to buy/sell at current stock price *query database*
- Medical and biological research
 - ▶ Obtain information from **articles** about diseases and treatments → decide which treatment to apply for a new patient
- Rumour detection
 - ▶ Detect events in social media → decide where, when and how to act

3

How?

- Given this:
 - ▶ “Brasilia, the Brazilian capital, was founded in 1960.”
- Obtain this:
 - ▶ capital(Brazil, Brasilia)
 - ▶ founded(Brasilia, 1960)
- Two steps:
 - ▶ **Named Entity Recognition (NER)**: find out entities such as “Brasilia” and “1960”
 - ▶ **Relation Extraction**: use context to find the relation between “Brasilia” and “1960” (“founded”)

4

Machine learning in IE

- **Named Entity Recognition (NER)**: sequence models such as RNNs, HMMs or CRFs. *(label)*
- **Relation Extraction**: mostly classifiers, either binary or multi-class. *type of relationship*
relation/non-relation
- This lecture: how to frame these two tasks in order to apply classifiers and sequence labellers.
- Choice of machine learning methods is up to the user (yes, deep learning methods can be applied).

5

Named Entity Recognition

6

Named Entity Recognition

Citing high fuel prices, United Airlines said Friday it has increased fares by \$6 per round trip on flights to some cities also served by lower-cost carriers.

American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

Named Entity Recognition

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

8

Typical Entity Tags

- **PER**: people, characters
- **ORG**: companies, sports teams
- **LOC**: regions, mountains, seas
- **GPE**: countries, states, provinces
(sometimes conflated with **LOC**)
- **FAC**: bridges, buildings, airports
- **VEH**: planes, trains, cars
- **Tag-set is application-dependent**: some domains deal with **specific entities** e.g. proteins, genes or works of art.

9

NER as Sequence Labelling

- NE tags can be ambiguous:
 - ▶ “Washington” can be either a person, a location or a political entity.
- We faced a similar problem when doing POS tagging.
 - ▶ Solution: incorporate context by treating NER as sequence labelling.
- Can we use an out-of-the-box sequence tagger for this (e.g., HMM)? word level
 - ▶ Not really: entities can span multiple tokens:
 - ▶ Solution: adapt the tag set.

10

IO tagging

- [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.
- ‘I-ORG’ represents a token that is inside an entity (ORG in this case).
- All tokens which are not entities get the ‘O’ token (for outside).
- Can not differentiate between a single entity with multiple tokens or multiple entities with single tokens.

one/two?

Words	IO Label
American	I-ORG
Airlines	I-ORG
,	O
a	O
unit	O
of	O
AMR	I-ORG
Corp.	I-ORG
,	O
immediately	O
matched	O
the	O
move	O
,	O
spokesman	O
Tim	I-PER
Wagner	I-PER
said	O
.	O

11

IOB tagging

- [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.
- B-ORG represents the beginning of an ORG entity.
following words: I-ORG
- If the entity has more than one token, subsequent tags are represented as I-ORG.

Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

12

NER as Sequence Labelling

- Given a tagging scheme and an annotated corpus, one can train any sequence labelling model
- In theory, HMMs can be used but discriminative models such as MEMMs and CRFs are preferred
 - ▶ Character-level features (is the first letter uppercase?)
 - ▶ Extra resources, e.g., lists of names
 - ▶ POS tags

add features more easily

13

NER: Features

- Character and word shape features (ex: “L’Occitane”)
- Prefix/suffix:
 - ▶ L / L' / L'O / L'Oc / ...
 - ▶ e / ne / ane / tane / ...
- Word shape:
 - ▶ X'XXXXXXXX / X'Xx
 - ▶ XXXX-XX-XX (date!)
- POS tags / syntactic chunks: many entities are nouns or noun phrases
- Presence in a gazeteer: lists of entities, such as place names, people's names and surnames, etc.

14

features

Word	POS	Chunk	Short shape	Label
American	NNP	B-NP	Xx	B-ORG
Airlines	NNPS	I-NP	Xx	I-ORG
,	,	O	,	O
a	DT	B-NP	x	O
unit	NN	I-NP	x	O
of	IN	B-PP	x	O
AMR	NNP	B-NP	X	B-ORG
Corp.	NNP	I-NP	Xx.	I-ORG
,	,	O	,	O
immediately	RB	B-ADVP	x	O
matched	VBD	B-VP	x	O
the	DT	B-NP	x	O
move	NN	I-NP	x	O
,	,	O	,	O
spokesman	NN	B-NP	x	O
Tim	NNP	I-NP	Xx	B-PER
Wagner	NNP	I-NP	Xx	I-PER
said	VBD	B-VP	x	O
.	,	O	.	O

Figure 18.6 Word-by-word feature encoding for NER.

15

NER: Classifier

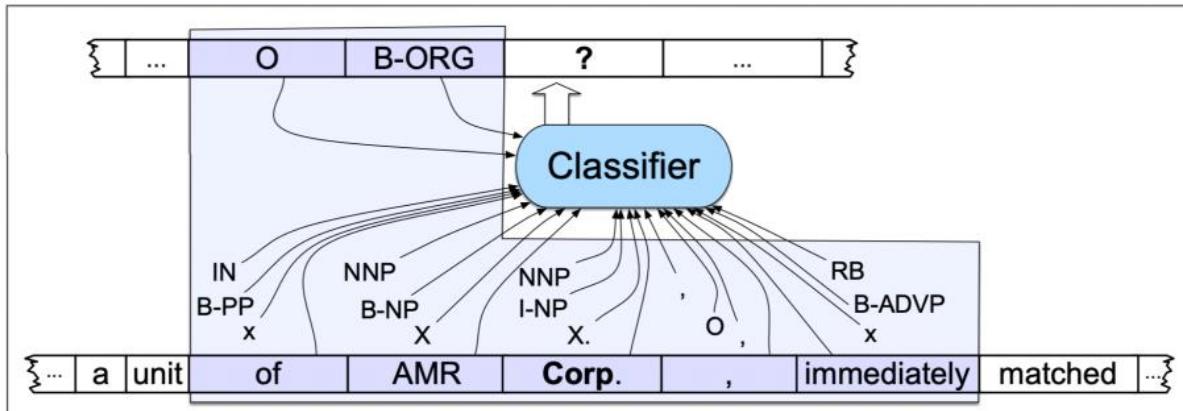
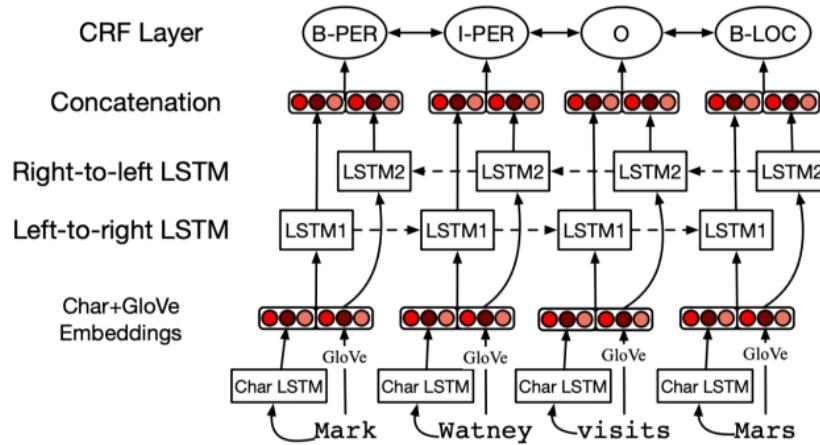


Figure 18.7 Named entity recognition as sequence labeling. The features available to the classifier during training and classification are those in the boxed area.

16

Deep Learning for NER

- *State of the art approach uses LSTMs with character and word embeddings (Lample et al. 2016)*



17

Relation Extraction

18

Relation Extraction

- [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.
- Traditionally framed as triple extraction:
 - ▶ unit(American Airlines, AMR Corp.)
 - ▶ spokesman(Tim Wagner, American Airlines)
- Key question: do we have access to a set of possible relations?
 - ▶ Answer depends on the application

19

Relation Extraction

- ▶ unit(American Airlines, AMR Corp.) → subsidiary
- ▶ spokesman(Tim Wagner, American Airlines) → employment

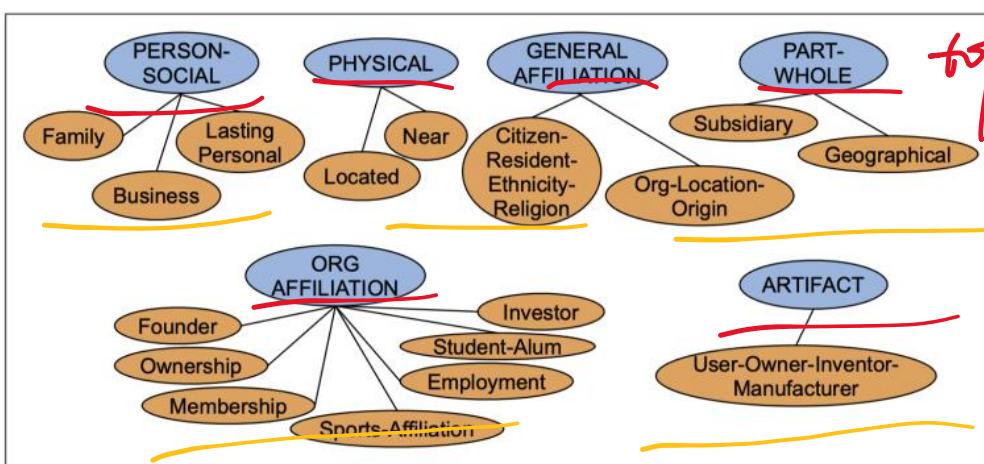


Figure 18.9 The 17 relations used in the ACE relation extraction task.

20

Methods

- If we have access to a **fixed relation database**:
 - ▶ Rule-based
 - ▶ Supervised
 - ▶ Semi-supervised
 - ▶ Distant supervision
- If **no restrictions on relations**:
 - ▶ Unsupervised
 - ▶ Sometimes referred as “OpenIE”

Closet relation

21

Rule-Based Relation Extraction

- “Agar is a substance prepared from a mixture of red algae such as Gelidium, for laboratory or industrial use.”
- [NP red algae] such as [NP Gelidium]
- NP_0 such as $NP_1 \rightarrow \text{hyponym}(NP_1, NP_0)$
- hyponym(Gelidium, red algae)
- Lexico-syntactic patterns: high precision, low recall, manual effort required

extract word pattern

22

More Rules

$NP \{, NP\}^* \{, \}$ (and or) other NP_H	temples, treasuries, and other important civic buildings
NP_H such as {NP,}* {(or and)} NP	red algae such as Gelidium
such NP_H as {NP,}* {(or and)} NP	such authors as Herrick, Goldsmith, and Shakespeare
$NP_H \{, \}$ including {NP,}* {(or and)} NP	common-law countries , including Canada and England
$NP_H \{, \}$ especially {NP,}* {(or and)} NP	European countries , especially France, England, and Spain

Figure 18.12 Hand-built lexico-syntactic patterns for finding hypernyms, using {} to mark optionality (Hearst 1992a, Hearst 1998).

23

Supervised Relation Extraction

- Assume a corpus with annotated relations
- Two steps. First, find if an entity pair is related or not (binary classification)
 - ▶ For each sentence, gather all possible entity pairs
 - ▶ Annotated pairs are considered positive examples
 - ▶ Non-annotated pairs are taken as negative examples
- Second, for pairs predicted as positive, use a multi-class classifier (e.g. SVM) to obtain the relation

24

Supervised Relation Extraction

- [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.
- First:
 - ▶ (American Airlines, AMR Corp.) → positive
 - ▶ (American Airlines, Tim Wagner) → positive
 - ▶ (AMR Corp., Tim Wagner) → negative
- Second:
 - ▶ (American Airlines, AMR Corp.) → subsidiary
 - ▶ (American Airlines, Tim Wagner) → employment

25

Features

- [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said.
- (American Airlines, Tim Wagner) → employment

M1 headword	<i>airlines</i> (as a word token or an embedding)
M2 headword	<i>Wagner</i>
Word(s) before M1	NONE
Word(s) after M2	said
Bag of words between	{ <i>a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman</i> }
M1 type	ORG
M2 type	PERS
Concatenated types	ORG-PERS
Constituent path	$NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
Base phrase path	$NP \rightarrow NP \rightarrow PP \rightarrow NP \rightarrow VP \rightarrow NP \rightarrow NP$
Typed-dependency path	$Airlines \leftarrow_{subj} matched \leftarrow_{comp} said \rightarrow_{subj} Wagner$

26

Semi-supervised Relation Extraction

- Annotated corpora is very expensive to create.
- Use seed tuples to bootstrap a classifier
 - Given a set of seed tuples
 - Find sentences containing these seed tuples
 - Extract general patterns from these sentences
 - Use these patterns to find new tuples
 - Repeat from step 2

27

Semi-supervised Relation Extraction

1. Given seed tuple: hub(Ryanair, Charleroi)
2. Find sentences containing terms in seed tuples
 - Budget airline **Ryanair**, which uses **Charleroi** as a hub, scrapped all weekend flights out of the airport.
3. Extract general patterns
 - [ORG], which uses [LOC] as a hub
4. Find new tuples with these patterns
 - hub(Jetstar, Avalon)
5. Add these new tuples to existing tuples and repeat step 2

28

Issue: Semantic Drift

- Pattern: [NP] has a {NP}* hub at [LOC]
 - *Sydney has a ferry hub at Circular Quay*
 - ▶ hub(Sydney, Circular Quay) *error accumulate!*
 - More erroneous patterns extracted from this tuple...
 - Should only accept patterns with high confidences
- being correct

29

Distant Supervision

- Semi-supervised methods assume the existence of seed tuples to mine new tuples
- Can we mine new tuples directly?
- Distant supervision obtain new tuples from a range of sources:
 - ▶ DBpedia
 - ▶ Freebase
- Generate massive training sets, enabling the use of richer features, and no risk of semantic drift
- Still rely on a fixed set of relations

30

ReVERB: Unsupervised Relation Extraction

- If there is no relation database or the goal is to find new relations, unsupervised approaches must be used.
- Relations become substrings, usually containing a verb
- “United has a hub in Chicago, which is the headquarters of United Continental Holdings.”
 - ▶ “has a hub in”(United, Chicago)
 - ▶ “is the headquarters of”(Chicago, United Continental Holdings)
- Main problem: mapping the substring relations into canonical forms

rule-based approaches

31

Evaluation

- NER: F1-measure at the entity level.
- Relation Extraction with known relation set: F1-measure
- Relation Extraction with unknown relations: much harder to evaluate
 - ▶ Usually need some human evaluation
 - ▶ Massive datasets used in these settings are impractical to evaluate manually: use a small sample
 - ▶ Can only obtain (approximate) precision, not recall.

32

Other IE Tasks

33

Temporal Expression Extraction

“[TIME July 2, 2007]: A fare increase initiated [TIME last week] by UAL Corp’s United Airlines was matched by competitors over [TIME the weekend], marking the second successful fare increase in [TIME two weeks].”

- **Anchoring:** when is “last week”?

▶ “last week” → 2007-W26

extract time

- **Normalisation:** mapping expressions to canonical forms.

▶ July 2, 2007 → 2007-07-02

- **Mostly rule-based approaches**

34

Event Extraction

- “American Airlines, a unit of AMR Corp., immediately [EVENT matched] [EVENT the move], spokesman Tim Wagner [EVENT said].”
- **Very similar to NER, including annotation and learning methods.**
- **Event ordering:** detect how a set of events happened in a timeline.
 - ▶ Involves both event extraction and temporal expression extraction.

35

A Final Word

- Information Extraction is a vast field with many different tasks and applications
 - Named Entity Recognition + Relation Extraction
 - Events can be tracked by combining event and temporal expression extraction
- Machine learning methods involve classifiers and sequence labelling models.

36

Reading

- JM3 Ch. 18 – 18.2
- References:
 - Lample et al, Neural Architectures for Named Entity Recognition, NAACL 2016
<https://github.com/glample/tagger>

37

Question Answering

COMP90042
Natural Language Processing
Lecture 19



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L19

Introduction

- Definition:** question answering ("QA") is the task of automatically determining the answer for a natural language question
- Main focus on "factoid" QA
 - Who is the prime minister of the United Kingdom in 2020?
→ Boris Johnson

based on facts
short, precise answer

2

COMP90042

L19

Examples

Factoid questions, have short precise answers:

- What war involved the battle of Chapultepec?
- Who was Confucius?
- What is the date of Boxing Day?
- What are some fragrant white climbing roses?
- What are tannins?

General non-factoid questions require a longer answer, critical analysis, summary, calculation, and more:

- Why is the date of Australia Day contentious?
- What is the angle 60 degrees in radians?

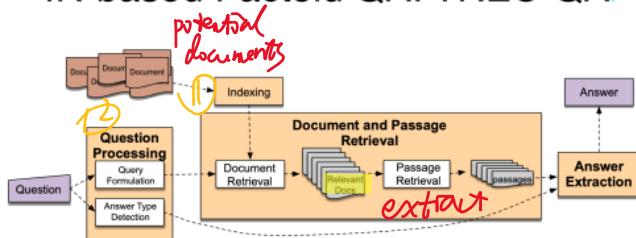
3

2 Key Approaches

- Information retrieval-based QA
 - Given a query, search relevant documents
 - Find answers within these relevant documents
 - Knowledge-based QA
 - Builds semantic representation of the query
 - Query database of facts to find answers
- Sources are different
- structure raw documents

IR-Based QA

IR-based Factoid QA: TREC-QA



1. Use question to make query for IR engine
2. Find document, and passage within document
3. Extract short answer string

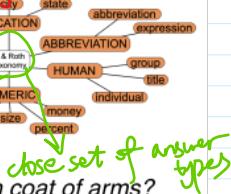
Question Processing

- Find key parts of question that will help retrieval
 - discard structural parts (wh-word, ?, etc) **function words**
 - formulate as tf-idf query, using unigrams or bigrams
 - identify entities and prioritise match
- May reformulate question using templates
- E.g., "Where is Federation Square located?"
 - query = "Federation Square located"
 - query = "Federation Square is located [in/at]"
- Predict expected answer type (here = LOCATION)

7

Answer Types

- Knowing the type of answer can help in:
 - finding the right passage containing the answer
 - finding the answer string → check if it is valid
- Treat as classification
 - given question, predict answer type
 - key feature is question headword
 - What are the **animals** on the Australian coat of arms?
 - Generally not a difficult task



8

Tag	Example
ABBREVIATION	
alt	What's the abbreviation for United Nations?
exp	What does the "z" stand for in the equation E=mc²?
DESCRIPTION	
definition	What are lemons?
descriptive	What is the name of the Canadian National anthem?
entity	How can you get rid of ants in your clothing?
reason	What caused the sinking of the Titanic?
ENTITY	
animal	What are the names of Odie's owners?
body	What part of your body contains the corpus callosum?
color	What color is the sky at sunset?
creature	In what book can I find the story of Aladdin?
currency	What is the name of the US dollar?
disease/condition	What does smallpox prevent?
food	What was the name of the food in the shape of a ship?
tool	What tool are we using in math?
instrument	What instrument does Max Roach play?
ling	What language is spoken in Australia?
letter	What letter appears on the cold-war tap in Spain?
object	What is the name of the first mobile phone?
plant	What are fragrant white climbing roses?
product	What is the function of a product?
reference	What are the names of the members?
sport	What was the name of the ball game played by the Mayans?
symbol	What is the chemical symbol for strontium?
synonym	What is another word for 'opposite'?
technique	How do you say "Grandma" in Italian?
topic	What is the name of Captain Haddock's ship?
whole	What is the singular of 'sheep'?
word	
HUMAN	
description	Who was Confucius?
group	What are the major companies that are part of Dow Jones?
ind	Who was the first Russian astronaut to do a spacewalk?
site	What was Queen Victoria's full reigning title?
LOCATION	
city	What's the oldest capital city in the Americas?
country	What is the name of the country where the Great Wall is?
mountain	What is the highest peak in Africa?
ocean	What is the name of the ocean?
state	What states do not have state income tax?
NUMERIC	
code	What is the telephone number for the University of Oklahoma?
count	About how many soldiers died in World War II?
date	What is the date of the American Revolution?
distance	How long was Mao's 1979 Long March?
money	How much did a McDonald's franchise cost in 1985?
number	What is the population of Mexico?
other	What is the population of Mexico?
percent	What fraction of a person's life is spent swimming?
ratio	How many times faster does a cheetah run than a puma?
percent	How fast must a spacecraft travel to escape Earth's gravity?
ratio	What is the size of Argentina?
spend	How many pounds are there in a stone?
size	
weight	

9

Retrieval

- Find top n documents matching query (standard IR)
- Next find passages (paragraphs or sentences) in these documents
- Should contain:
 - many instances of the question keywords
 - several named entities of the answer type
 - close proximity of these terms in the passage
 - high ranking by IR engine; etc
- Re-rank IR outputs to find best passage (e.g., using supervised learning)

10

Answer Extraction

- Find a concise answer to the question, as a span in the text
 - "Who is the federal MP for Melbourne?"
 - The Division of Melbourne is an Australian Electoral Division in Victoria, represented since the 2010 election by Adam Bandt, a member of the Greens.*
 - "How many Australian PMs have there been since 2013?"
 - Australia has had five prime ministers in five years. No wonder Merkel needed a cheat sheet at the G-20.*

11

Feature-Based Answer Extraction

- Frame it as a classification problem
- Classify whether a candidate answer (typically a short span) contains an answer
- Various features based on match to question, expected entity type match, specific answer patterns

"Who is the federal MP for Melbourne?"

The Division of Melbourne is an Australian Electoral Division in Victoria, represented since the 2010 election by Adam Bandt, a member of the Greens.

12

Neural Answer Extraction

- Use a neural network to extract answer
- AKA reading comprehension task
- But deep learning models requires lots of data
- Do we have enough data to train comprehension models?

13

MCTest (Richardson et al. 2016)

- Crowdworkers write fictional stories, questions and answers
- 500 stories, 2000 questions
- Multiple choice questions

James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.

One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.

After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

- 1) What is the name of the trouble making turtle?
 A) Fries
 B) Pudding
 C) James
 D) Jane

14

SQuAD (Rajpurkar et al. 2016)

- Use Wikipedia passages
- First set of crowdworkers create questions (given passage)
- Second set of crowdworkers label the answer make sure questions are answerable
- 150K questions (!)
- Second version includes unanswerable questions

Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, <i>Dangerously in Love</i> (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Q: "In what city and state did Beyoncé grow up?"
A: "Houston, Texas"
Q: "What areas did Beyoncé compete in when she was growing up?"
A: "singing and dancing"
Q: "When did Beyoncé release <i>Dangerously in Love</i> ?"
A: "2003"

15

train the reading comprehension system
some unanswerable questions

Reading Comprehension

- **Answer span** starts/ends at which token in passage?
- Compute:

- $p_{start}(i)$: prob. of token i is the starting token
- $p_{end}(i)$: prob. of token i is the ending token

Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".

Q: "In what city and state did Beyoncé grow up?"
A: "Houston, Texas"

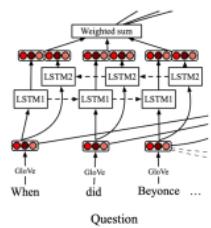
find highest
{ p_{start}
 p_{end}

16

LSTM-Based (Chen et al. 2017)

- Feed question tokens to a bidirectional LSTM
- Aggregate LSTM outputs via weighted sum to produce q , the final question embedding

Glove embeddings

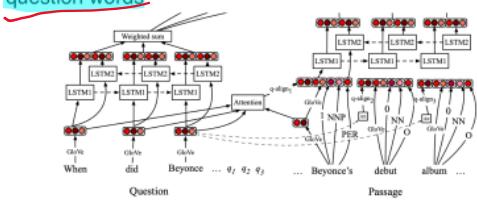


17

LSTM-Based (Chen et al. 2017)

- Process passage in a similar way, using another bidirectional LSTM
- More than just word embeddings as input
 - A feature to denote whether the word matches a question word
 - POS feature
 - Weighted question embedding: produced by attending to each question words

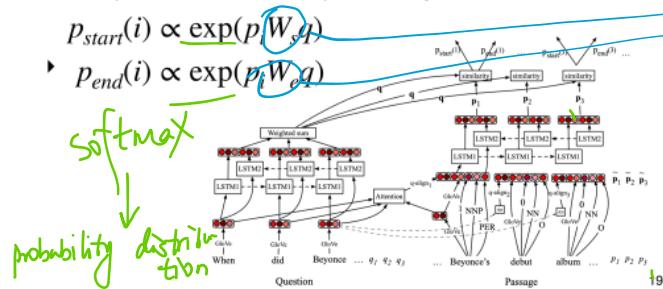
1/0



18

LSTM-Based (Chen et al. 2017)

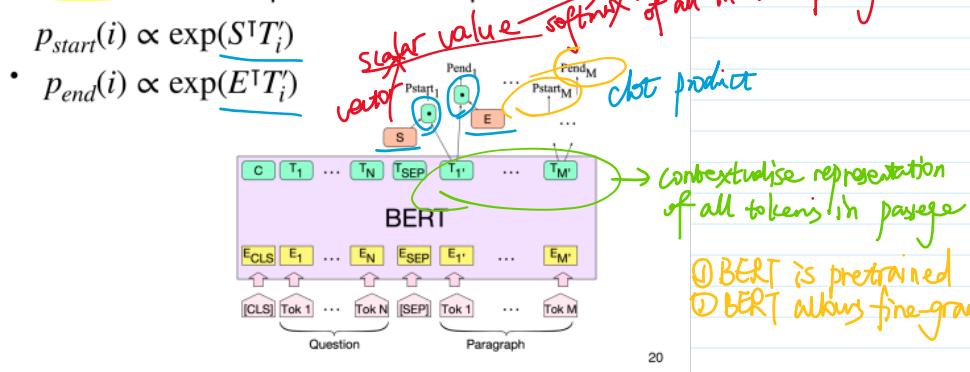
- $\{p_1, \dots, p_m\}$: one vector for each passage token from bidirectional LSTM
only one vector represents *whole question*
- To compute start and end probability for each token:



parameters in model

Bert-Based

- Fine-tune BERT to predict answer span



contextualise representation of all tokens in passage

- BERT is pretrained (not train from scratch)
- BERT allows fine-gradient attention between question and paragraph

Knowledge-Based QA

QA over structured KB

- Many large knowledge bases
 - Sports statistics, Moon rock data, ...
 - Freebase, DBpedia, Yago, ...
- Each with own query language, SQL, SPARQL etc.
- Can we support natural language queries?
 - E.g., "When was Ada Lovelace born?" → birth-year (Ada Lovelace, ?x)
"What is the capital of England?" → capital-city(?x, England)
 - And use it to query KB to find triple (*Ada Lovelace*, *birth-year*, *1815*) and provide answer *1815*.

22

Semantic Parsing

- Based on aligned questions and their logical form, e.g., GeoQuery (Zelle & Mooney 1996)
 - What is the capital of the state with the largest population?
 - answer(C, (capital(S,C), largest(P, (state(S), population(S,P)))))
- Can model using parsing (Zettlemoyer & Collins 2005) to build compositional logical form

$$\begin{array}{c}
 \text{What} \qquad \text{states} \qquad \text{border} \qquad \text{Texas} \\
 \hline
 \frac{(S/(S\setminus NP))/N}{\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)} \qquad \frac{N}{\lambda x.state(x)} \qquad \frac{(S\setminus NP)/NP}{\lambda x.\lambda y.borders(y,x)} \qquad \frac{NP}{\lambda x.borders(x,texas)} \\
 \hline
 \frac{S/(S\setminus NP)}{\lambda g.\lambda x.state(x) \wedge g(x)} \qquad \frac{(S\setminus NP)}{\lambda y.borders(y,texas)} \\
 \hline
 \frac{}{\lambda x.state(x) \wedge borders(x,texas)}
 \end{array}$$

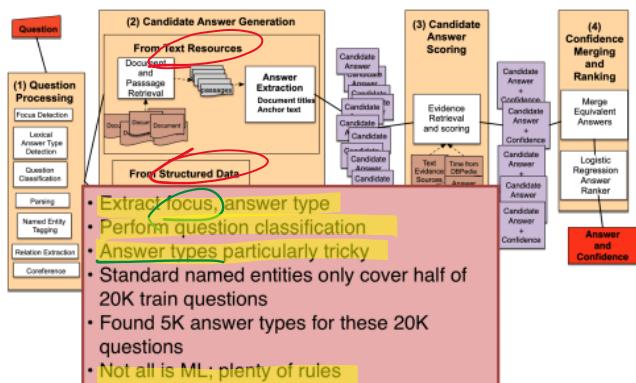
23

Hybrid Methods

- Why not use both text-based and knowledge-based resources for QA?
- IBM Watson (Ferrucci et al., 2010) which won the game show *Jeopardy!* uses a wide variety of resources to answer questions
 - William Wilkinson's "An Account of The Principalities of Wallachia and Moldavia" inspired this author's most famous novel.
 - Bram Stoker, Dracula

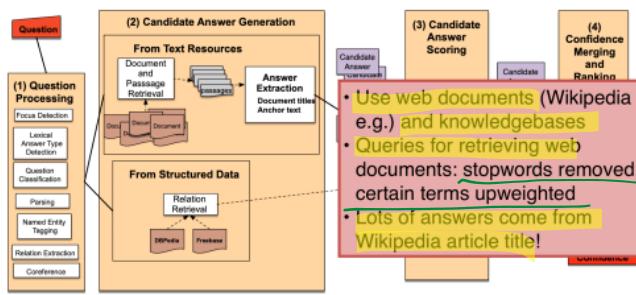
24

IBM's WATSON



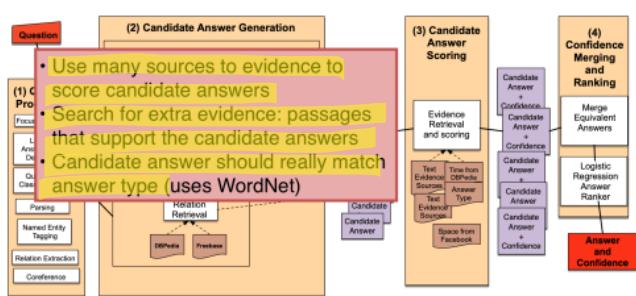
25

IBM's WATSON



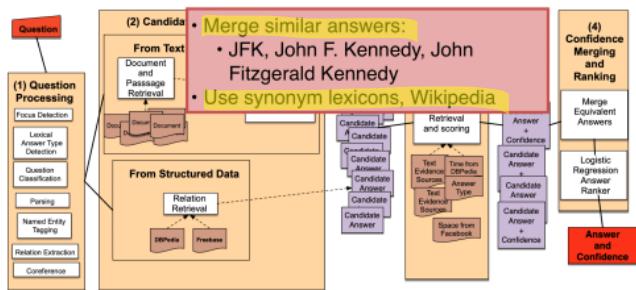
26

IBM's WATSON



27

IBM's WATSON



28

QA Evaluation

- TREC-QA: Mean Reciprocal Rank for systems returning matching passages or answer strings
 - ▶ E.g. system returns 4 passage for a query, first correct passage is the 3rd passage
 - ▶ $MRR = \frac{1}{3}$
- SQuAD:
 - ▶ Exact match of string against gold answer
 - ▶ F1 score over bag of selected tokens
- MCQ reading comprehension: Accuracy

29

A Final Word

- IR-based QA: search textual resources to answer questions
 - ▶ Reading comprehension: assumes question+passage
- Knowledge-based QA: search structured resources to answer questions
- Hot area: many new approaches & evaluation datasets being created all the time (narratives, QA, commonsense reasoning, etc)

30

Reading

- JM3 Ch. 25

31

References

- Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. In *ACL 2017*.
- Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. (2010). Building Watson: An overview of the DeepQA project. *AI magazine* 31(3), 59–79.
- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*.
- Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.
- Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *AAAI*.

32



Topic Modelling

COMP90042
Natural Language Processing
Lecture 20



Making Sense of Text

- English Wikipedia: 6M articles
- Twitter: 500M tweets per day
- New York Times: 15M articles
- arXiv: 1M articles
- What can we do if we want to learn something about these document collections?

2

Questions

- What are the less popular topics on Wikipedia?
- What are the big trends on Twitter in the past month?
- How do the themes/topics evolve over time in New York Times from 1900s to 2000s?
- What are some influential research areas?

3

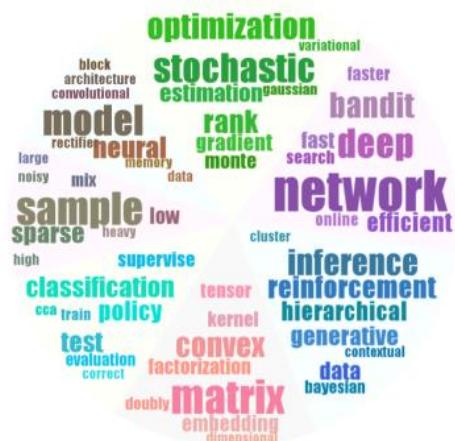
Topic Models To The Rescue

- Topic models learn common, overlapping themes in a document collection
 - Unsupervised model
 - ▶ No labels; input is just the documents!

4

What Is A Topic?

- A set of words
 - Collectively describes a concept or subject
 - Words of a topic typically appear in the same set of documents in the corpus



Li, Chua, 2017 ("Document Visualization using Topic Clouds")

5

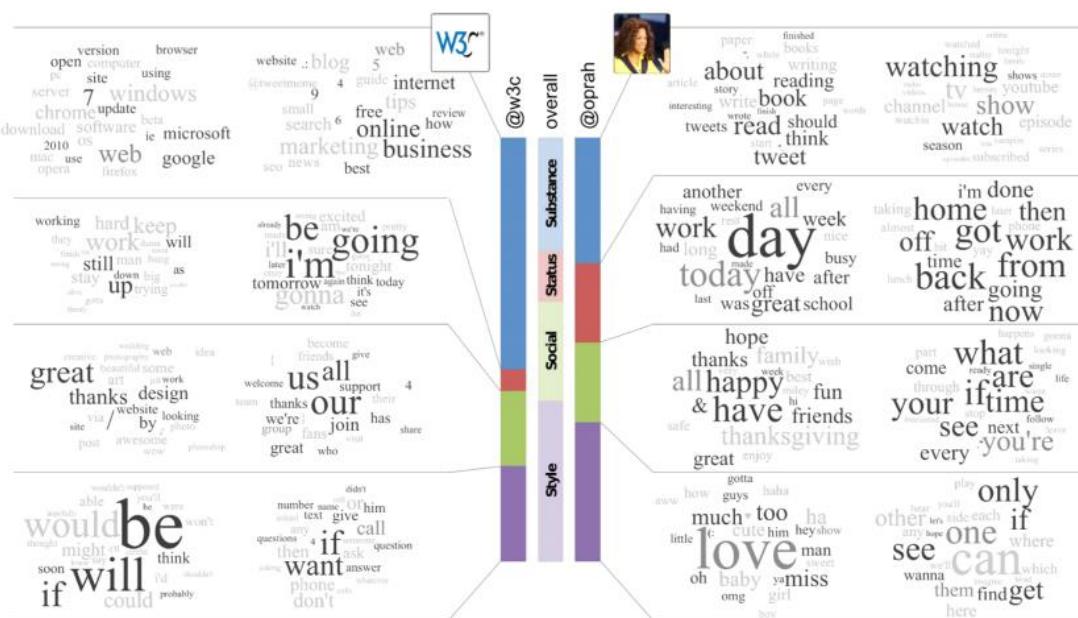
Wikipedia Topics



<https://appliedmachinelearning.blog/2017/09/28/topic-modelling-part-2-discovering-topics-from-articles-with-latent-dirichlet-allocation/>

6

Twitter Topics



Ramage, Dumais, Liebling, 2010

7

New York Times Topics

music band songs rock album jazz pop song singer night	book life novel story books man stories love children family	art museum show exhibition artist artists paintings painting century works	game Knicks nets points team season play games night coach	show film television movie series says life man character know
theater play production show stage street broadway director musical directed	clinton bush campaign gore political republican dole presidential senator house	stock market percent fund investors funds companies stocks investment trading	restaurant sauce menu food dishes street dining dinner chicken served	budget tax governor county mayor billion taxes plan legislature fiscal

<http://journalofdigitalhumanities.org/2-1/topic-modeling-and-digital-humanities-by-david-m-blei/>

8

A Brief History of Topic Models

9

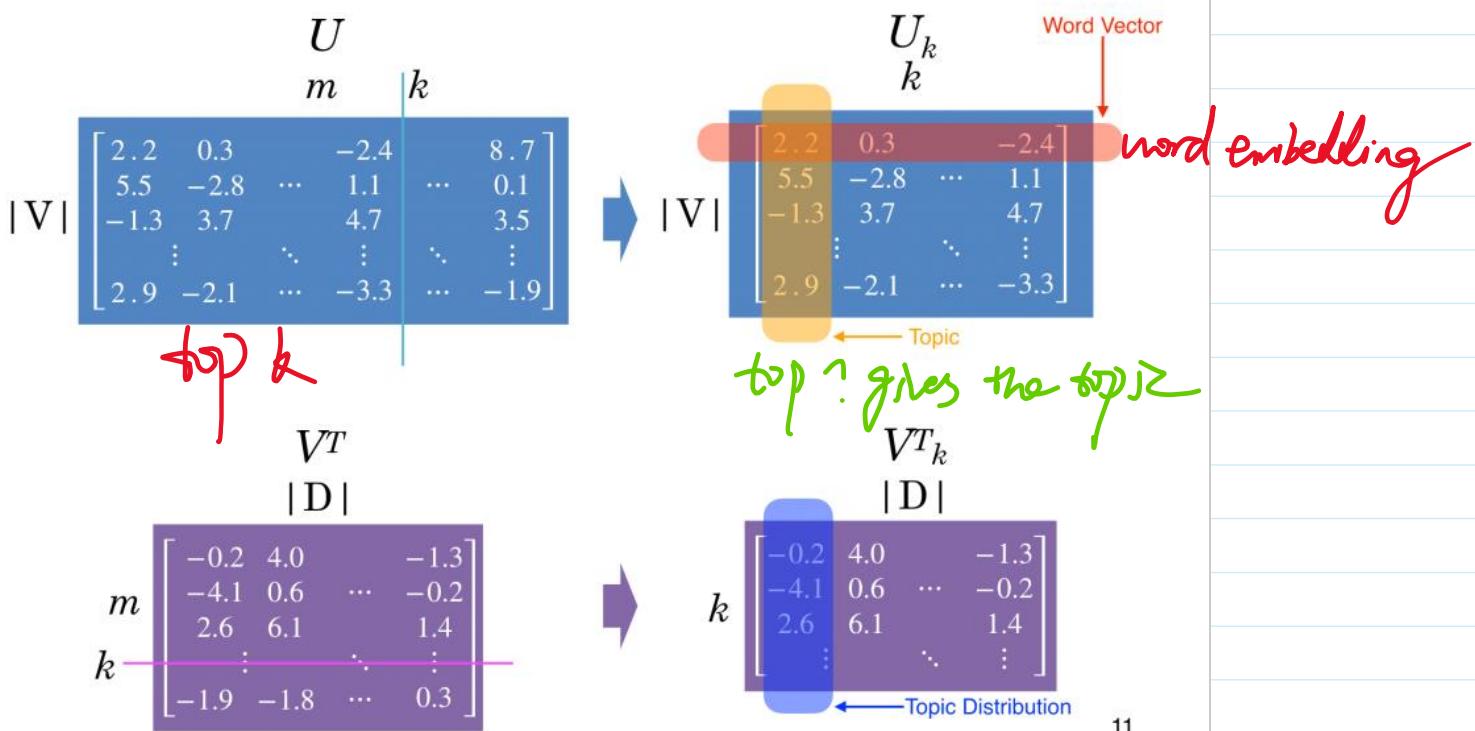
What Is A Topic Model?

- Latent Semantic Analysis (L10): SVD+Truncate

$$A = U \Sigma V^T$$

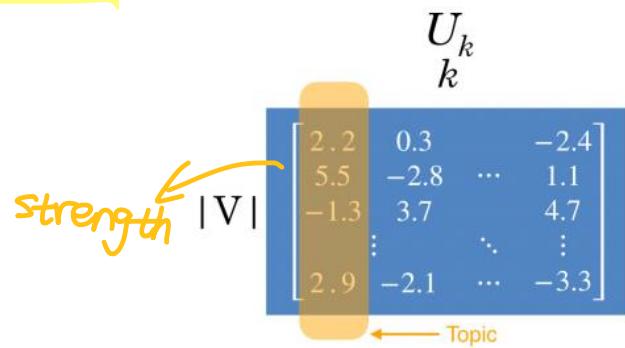
A
 $|D|$
 $|V|$ $\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$
 \downarrow
 U
 m
 $|V|$ $\begin{bmatrix} 2.2 & 0.3 & \dots & 8.7 \\ 5.5 & -2.8 & \dots & 0.1 \\ -1.3 & 3.7 & \dots & 3.5 \\ \vdots & \ddots & \ddots & \vdots \\ 2.9 & -2.1 & \dots & -1.9 \end{bmatrix}$
 Σ
 m
 m $\begin{bmatrix} 9.1 & 0 & 0 & \dots & 0 \\ 0 & 4.4 & 0 & \dots & 0 \\ 0 & 0 & 2.3 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0.1 \end{bmatrix}$
 V^T
 $|D|$
 m $\begin{bmatrix} -0.2 & 4.0 & \dots & -1.3 \\ -4.1 & 0.6 & \dots & -0.2 \\ 2.6 & 6.1 & \dots & 1.4 \\ \vdots & \ddots & \ddots & \vdots \\ -1.9 & -1.8 & \dots & 0.3 \end{bmatrix}$
10

$$A = U \Sigma V^T \quad \text{LSA: Truncate}$$



Issues

- Positive and negative values in the U and V^T
- Difficult to interpret



12

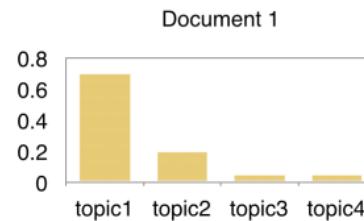
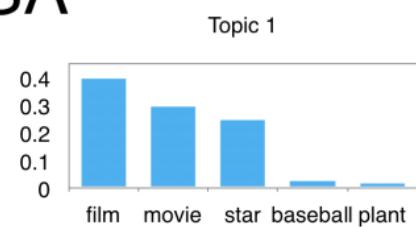
Probabilistic LSA

- Based on a probabilistic model

$$\begin{aligned}
 P(w, d) &= P(w | d)P(d) \\
 &= P(d) \sum_T P(w | t)P(t | d)
 \end{aligned}$$

Joint probability of a word and a document
 Number of topics
 Topic distribution for a document

Word distribution for a topic



13

Issues

- No more negative values!
- PLSA can learn topics and topic distribution for documents in the train corpus
- But it is unable to infer topic distribution on new documents
- PLSA needs to be re-trained for new documents

14

Latent Dirichlet Allocation

- Introduces a prior to the document-topic and topic-word distribution
- Fully generative; trained LDA model can infer topics on unseen documents!
- LDA is a Bayesian version of PLSA

15

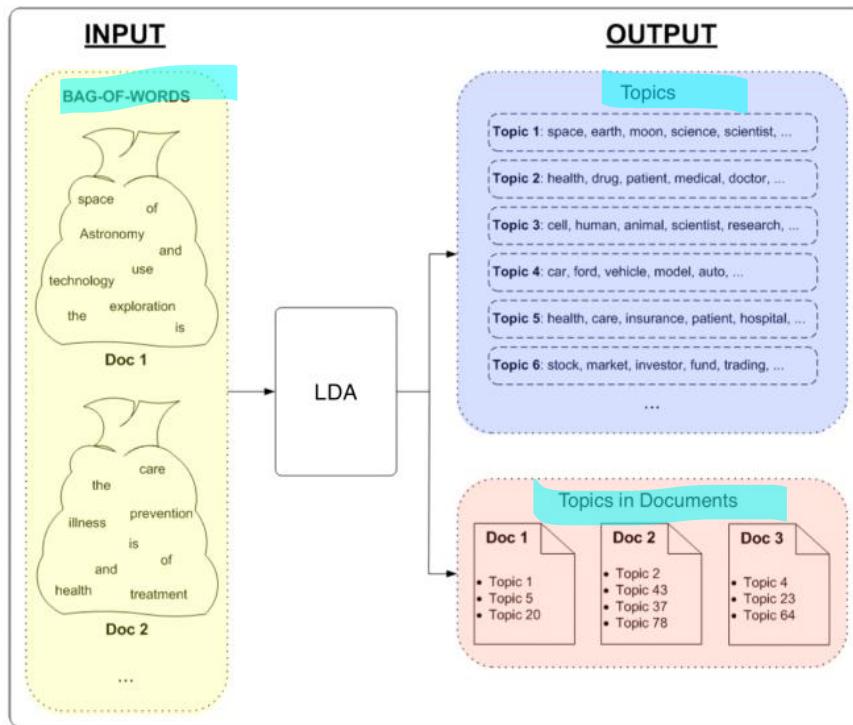
Latent Dirichlet Allocation

16

LDA

- Core idea: assume each document contains a mix of topics
- But the topic structure is hidden (latent)
- LDA infers the topic structure given the observed words and documents
- LDA produces soft clusters of documents (based on topic overlap), rather than hard clusters

17



18

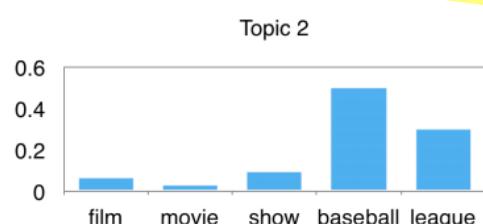
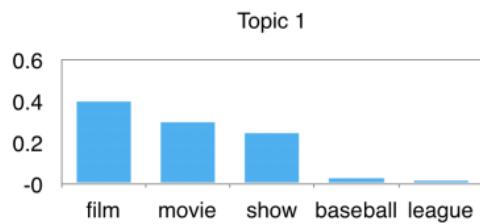
Input

- A collection of documents
- Bag-of-words
- Good preprocessing practice: appear a lot in topics
 - ▶ Remove stopwords → occur frequently
 - ▶ Remove low and high frequency word types
 - ▶ Lemmatisation

19

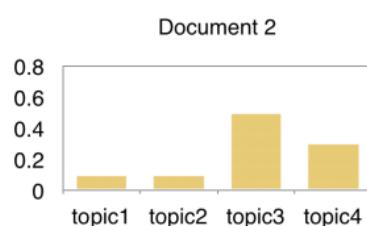
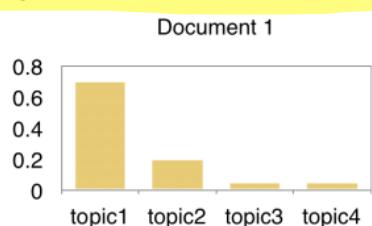
Output

- **Topics:** multinomial distribution over words in each topic



top k

- **Topics in documents:** multinomial distribution over topics in each document



20

Learning

- How do we learn the latent topics?
- Two main family of algorithms:
 - ▶ **Variational methods**
 - ✖ **Sampling-based methods**

21

Sampling Method (Gibbs)

- Randomly assign topics to all tokens in documents

doc ₁	mouse: t ₁	cat: t ₃	rat: t ₂	chase: t ₁	mouse: t ₃
doc ₂	scroll: t ₁	mouse: t ₃	scroll: t ₃	scroll: t ₂	click: t ₂
doc ₃	tonight: t ₂	baseball: t ₁	tv: t ₂	exciting: t ₁	

- Collect topic-word and document-topic co-occurrence statistics based on the assignments

	mouse	cat	scroll	tv	...
t ₁	1	0	1	0	
t ₂	0	0	1	1	
t ₃	2	1	1	0	

	t ₁	t ₂	t ₃
d ₁	2	1	2
d ₂	1	2	2
...			

✓5

- Go through every word token in corpus and sample a new topic

$$\triangleright P(t_i) \propto P(t_i | w)P(t_i | d)$$



Need to de-allocate the current topic assignment and update the co-occurrence matrices before sampling

- Repeat until convergence

22

When Do We Stop?

- Train until convergence
- Convergence = model probability of training set becomes stable
- How to compute model probability?

$$\log P(w_1, w_2, \dots, w_m) = \log \sum_{j=0}^T P(w_1 | t_j)P(t_j | d_{w_1}) + \dots + \log \sum_{j=0}^T P(w_m | t_j)P(t_j | d_{w_m})$$

$$\triangleright m = \# \text{word tokens}$$

Based on the topic-word co-occurrence matrix

Based on the document-topic co-occurrence matrix

23

Infer Topics For New Documents

- Randomly assign topics to all tokens in new/test documents

testdoc ₁	tiger: t ₂	cow: t ₁	cat: t ₃	tiger: t ₃	
testdoc ₂	football: t ₂	live: t ₂	men: t ₂	fun: t ₃	soccer: t ₁
testdoc ₃	news: t ₁	cnn: t ₃	tonight: t ₁		

- Update document-topic matrix based on the assignments; but use the trained topic-word matrix

from trained topic model

	mouse	cat	scroll	tv	...
t ₁	1	0	1	0	
t ₂	0	0	1	1	
t ₃	2	1	1	0	

new matrix

	t ₁	t ₂	t ₃
td ₁	1	1	2
td ₂	1	3	1
...			

- Go through every word in the test documents and sample topics:

$$\rightarrow P(t_i) \propto P(t_i | w)P(t_i | d)$$

- Repeat

24

Hyper-Parameters

- T: number of topic

attorney
charges
judge
authorities

economic
economy
company
market
sales

prison
manning
attorney
classified
bradley

media
oprah
winfrey
franken
network

Low T (<10): broad topics

High T (100+): fine-grained, specific topics

25

Hyper-Parameters

- β : prior on the topic-word distribution
- α : prior on the document-topic distribution
- Analogous to k in add- k smoothing in N -gram LM
- Pseudo counts when computing:
 - ▶ $p(w | t) : \beta$
 - ▶ $p(t | d) : \alpha$

$p(t_1 | d_1) = \frac{2 + \alpha}{5 + T\alpha}$

	mouse	cat	scroll	tv	...
t ₁	1	0	1	0	
t ₂	0	0	1	1	
t ₃	2	1	1	0	

	t ₁	t ₂	t ₃
d ₁	2	1	2
d ₂	1	2	2
...			

26

Hyper-Parameters

- High prior values → flatter distribution
- a very very large value would lead to a uniform distribution
- Low prior values → peaky distribution
- β : generally small (< 0.01)
 - ▶ Large vocabulary, but we want each topic to focus on specific themes
- α : generally larger (> 0.1)
 - ▶ Multiple topics within a document

Evaluation

28

How To Evaluate Topic Models?

- **Unsupervised learning → no labels**
- Intrinsic evaluation:
 - ▶ **model logprob / perplexity** on test documents

$$\triangleright L = \prod_w \sum_t P(w | t)P(t | d_w)$$

$$\triangleright \text{ppl} = \exp^{\frac{-\log L}{m}}$$

total number of word tokens in test documents

↓
smaller is better

29

Issues with Perplexity

- More topics = better (lower) perplexity
- Smaller vocabulary = better perplexity
 - ▶ Perplexity not comparable for different corpora, or different tokenisation/preprocessing methods
- Does not correlate with human perception of topic quality
- Extrinsic evaluation the way to go:
 - ▶ Evaluate topic models based on downstream task

30

Topic Coherence

- A better intrinsic evaluation method
- Measure how **coherent** the generated topics

food,
 farmers,
rice,
farm,
 agriculture simply
 give unionist
 choice
 count
i.e.

- A good topic model is one that generates more coherent topics

31

Word Intrusion

- Idea: inject one random word to a topic
 - {farmers, farm, food, rice, agriculture}
 - ↓
 - {farmers, farm, food, rice, **cat**, agriculture}
- Ask users to guess which is the **intruder word**
- Correct guess → topic is coherent
- Try guess the intruder word in:
 - ▶ {choice, count, village, i.e., simply, unionist}
- Manual effort; does not scale

32

Estimate Coherence Automatically?

- PMI!
- Compute pairwise PMI of top- N words in a topic
 - ▶
$$\text{PMI}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$$
- Given topic: {farmers, farm, food, rice, agriculture}
- Sum logPMI for all word pairs in the topic:
 - ▶ $\text{logPMI}(\text{farmers}, \text{farm}) + \text{logPMI}(\text{farmers}, \text{food}) + \dots + \text{logPMI}(\text{rice}, \text{agriculture})$

33

Variants

- Normalised PMI

$$\text{NPMI}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \frac{\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)}$$

- Conditional probability

$$\text{LCP}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \log \frac{P(w_i, w_j)}{P(w_i)}$$

- Good correlation with human perception of topic coherence
- Better correlation if we use a different corpus to estimate PMI

i.e. don't use corpus that we run the topic model on

34

PMI Examples

Topic	PMI	NPMI
cell hormone insulin muscle receptor	0.61	0.59
electron laser magnetic voltage wavelength	0.54	0.52
magnetic neutrino particle quantum universe	0.55	0.55
album band music release song	0.37	0.56
college education school student university	0.38	0.57
city county district population town	0.34	0.52

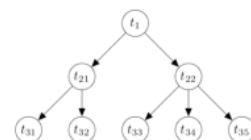
35

Improvements

36

Topic Model Variants

- Use phrases or n-grams instead of words
- Learn hierarchical topics
- Non-parametric models
 - ▶ #topics automatically learned
- Supervised models
 - ▶ Takes into account document labels



37

Topic Labelling

vmware server virtual oracle update	→	virtualisation
church archway building window gothic	→	church architecture
investigation fbi official department federal	→	criminal investigation
rate population prevalence study incidence	→	mortality rate

- Use wikipedia article titles as labels
- Measure distance between a label and topic words based on document embeddings and word embeddings

38

A Final Word

- Topic model: an unsupervised model for learning latent concepts in a document collection
- LDA: a popular topic model
 - ▶ Learning
 - ▶ Hyper-parameters
- How to evaluate topic models?
 - ▶ Topic coherence

39



Summarisation

COMP90042
Natural Language Processing
Lecture 21



COPYRIGHT 2020, THE UNIVERSITY OF MELBOURNE

1

COMP90042

L21

Summarisation

- Distill the most important information from a text to produce shortened or abridged version
- Applications
 - ▶ outlines of a document
 - ▶ abstracts of a scientific article
 - ▶ headlines of a news article
 - ▶ snippets of search result

w: en.wikipedia.org : wiki : Natural_language_processing ↗
Natural language processing - Wikipedia
Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.
Natural-language understanding · Natural-language generation · ↑ the Road

towardsdatascience.com : your-guide-to-natural-langu... ↗
Your Guide to Natural Language Processing (NLP) - Towards ...

Jan 15, 2019 · Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages. It is a discipline that focuses on the interaction between data science and human language, and is scaling to lots of industries.

2

What to Summarise?

- **Single-document summarisation**
 - ▶ Input: a single document
 - ▶ Output: summary that characterise the content
- **Multi-document summarisation**
 - ▶ Input: multiple documents
 - ▶ Output: summary that captures the gist of all documents
 - ▶ E.g. summarise a news event from multiple sources or perspectives

3

How to Summarise?

- **Extractive summarisation**
 - ▶ Summarise by selecting representative sentences from documents
- **Abstractive summarisation**
 - ▶ Summarise the content in your own words
 - ▶ Summaries will often be paraphrases of the original content

4

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that this nation might live. It is altogether fitting and proper that we should do this. But, in a larger sense, we cannot dedicate...we cannot consecrate...we cannot hallow... this ground. The brave men, living and dead, who struggled here, have consecrated it far above our poor power to add or detract. The world will little note nor long remember what we say here, but it can never forget what they did here. It is for us, the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us...that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion; that we here highly resolve that these dead shall not have died in vain; that this nation, under God, shall have a new birth of freedom; and that government of the people, by the people, for the people, shall not perish from the earth.

Extract from the Gettysburg Address:

Four score and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field. But the brave men, living and dead, who struggled here, have consecrated it far above our poor power to add or detract. From these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion — that government of the people, by the people for the people shall not perish from the earth.

important sentences

Abstract of the Gettysburg Address:

This speech by Abraham Lincoln commemorates soldiers who laid down their lives in the Battle of Gettysburg. It reminds the troops that it is the future of freedom in America that they are fighting for.

5

COMP90042

L21

Why Summarise?

- **Generic summarisation**

- ▶ Summary gives important information in the document(s)

- **Query-focused summarisation**

- ▶ Summary responds to a user query
 - ▶ Similar to question answering
 - ▶ But answer is much longer (not just a phrase)

6

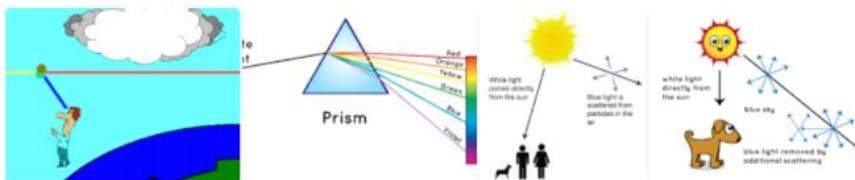
Query-Focused Summarisation

why is the sky blue



All Videos Images Shopping News More Settings Tools

About 4,580,000,000 results (0.55 seconds)



Blue light is scattered in all directions by the tiny molecules of air in Earth's atmosphere. **Blue** is scattered more than other colors because it travels as shorter, smaller waves. This is why we see a **blue sky** most of the time. ... Also, the surface of Earth has reflected and scattered the light.

[spaceplace.nasa.gov › blue-sky](https://spaceplace.nasa.gov/blue-sky)

[Why Is the Sky Blue? | NASA Space Place – NASA Science for ...](https://spaceplace.nasa.gov/blue-sky)

7

Outline

- Extractive summarisation
 - ▶ Single-document
 - ▶ Multi-document
- Abstractive summarisation
 - ▶ Single-document (deep learning models!)
- Evaluation

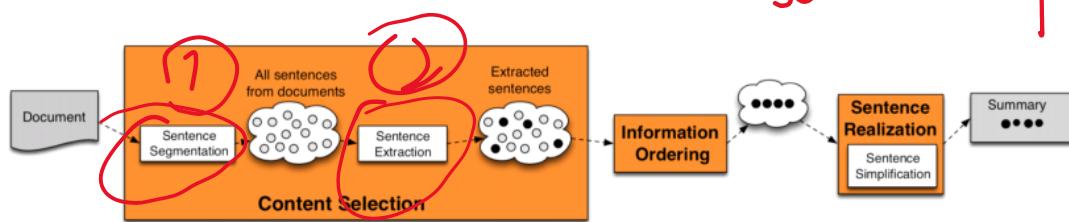
8

Extractive: Single-Doc

9

Summarisation System

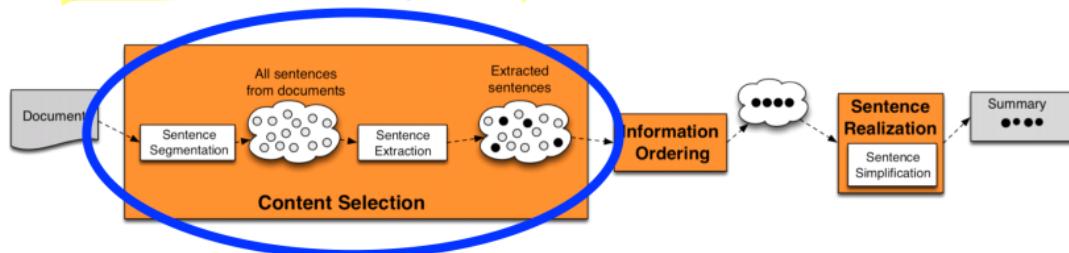
- **Content selection:** select what sentences to extract from the document
 - **Information ordering:** decide how to order extracted sentences
 - **Sentence realisation:** cleanup to make sure combined sentences are fluent
- sentence simplification



10

Summarisation System

- We will focus on **content selection**
- For single-document summarisation, information ordering not necessary
 - ▶ present extracted sentences in original order
- Sentence realisation also not necessary if they are presented in dot points



11

Content Selection

- Not much data with ground truth extractive sentences
- Mostly unsupervised methods
- Goal: Find sentences that are important or salient

12

Method 1: TF-IDF

- Frequent words in a doc → salient
- But some generic words are very frequent but uninformative
 - ▶ function words
 - ▶ stop words
- Weigh each word w in document d by its inverse document frequency:
 - ▶ $\text{weight}(w) = \underline{tf_{d,w}} \times \underline{idf_w}$

13

Method 2: Log Likelihood Ratio

- Intuition: a word is salient if its probability in the input corpus is very different to a background corpus

$$\text{weight}(w) = \begin{cases} 1, & \text{if } -2\log\lambda(w) > 10 \\ 0, & \text{otherwise} \end{cases}$$

} binomial distribution

$\lambda(w)$ is the ratio between:

- ▶ $P(\text{observing } w \text{ in } I)$ and $P(\text{observing } w \text{ in } B)$, assuming $P(w|I) = P(w|B) = p$
- ▶ $P(\text{observing } w \text{ in } I)$ and $P(\text{observing } w \text{ in } B)$, assuming $P(w|I) = p_I$ and $P(w|B) = p_B$

14

Saliency of A Sentence?

- $\text{weight}(s) = \frac{1}{|S|} \sum_{w \in S} \text{weight}(w)$

- Only consider non-stop words in S

15

Method 3: Sentence Centrality

- Alternative approach to ranking sentences
- Measure distance between sentences, and choose sentences that are closer to other sentences
- Use tf-idf to represent sentence
- Use cosine similarity to measure distance

- $\text{centrality}(s) = \frac{1}{\#\text{sent}} \sum_{s'} \cos_{tfidf}(s, s')$

16

Final Extracted Summary

- Use top-ranked sentences as extracted summary
 - ▶ Saliency (tf-idf or log likelihood ratio)
 - ▶ Centrality

17

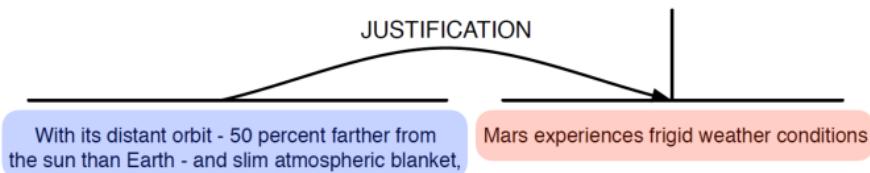
Method 4: RST Parsing

With its distant orbit – 50 percent farther from the sun than Earth – and slim atmospheric blanket, Mars experiences frigid weather conditions. Surface temperatures typically average about -70 degrees Fahrenheit at the equator, and can dip to -123 degrees C near the poles. Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion, but any liquid water formed in this way would evaporate almost instantly because of the low atmospheric pressure. Although the atmosphere holds a small amount of water, and water-ice clouds sometimes develop, most Martian weather involves blowing dust or carbon dioxide.

18

Method 4: RST Parsing

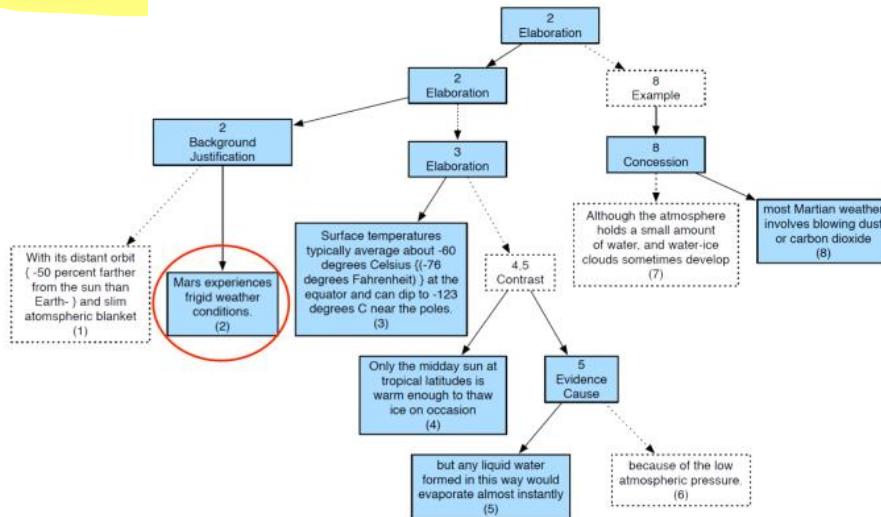
- Rhetorical structure theory (L12, Discourse): explain how clauses are connected
- Define the types of relations between a **nucleus** (main clause) and a **satellite** (supporting clause)



19

Method 4: RST Parsing

- Nucleus more important than satellite
- A sentence that functions as a nucleus to more sentences = more salient



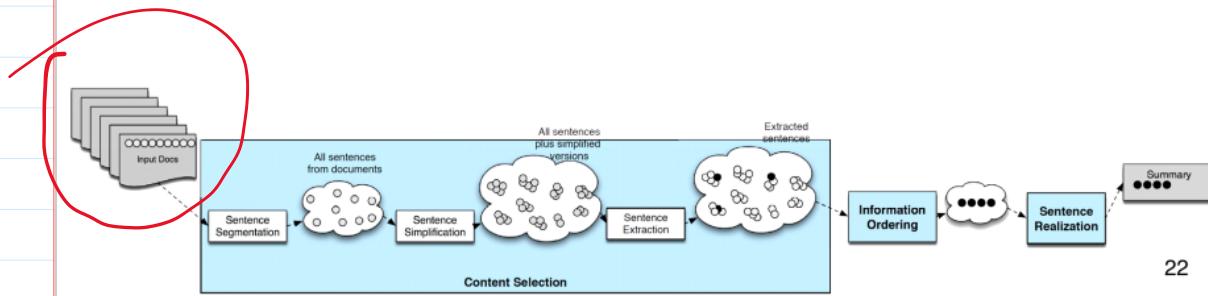
20

Extractive: Multi-Doc

21

Summarisation System

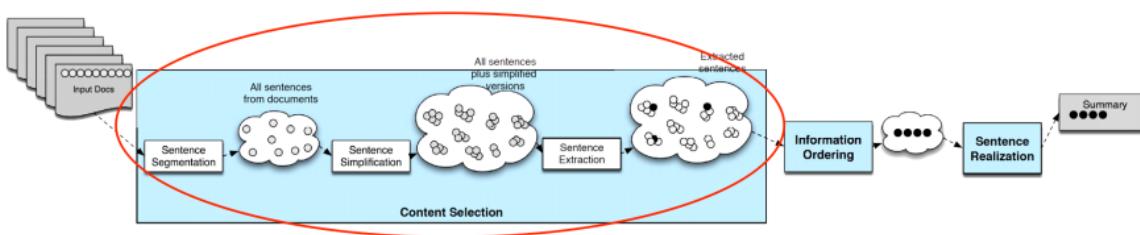
- Similar to single-document extractive summarisation system
- Challenges:
 - ▶ Redundancy in terms of information
 - ▶ Sentence ordering



22

Content Selection

- We can use the same unsupervised content selection methods (tf-idf, log likelihood ratio, centrality) to select salient sentences
- But ignore sentences that are redundant

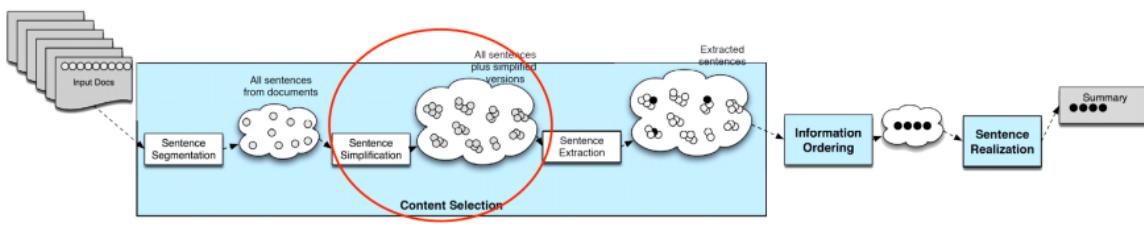


Maximum Marginal Relevance

- Iteratively select the best sentence to add to summary
- Sentences to be added must be novel
- Penalise a candidate sentence if it's similar to extracted sentences:
 - ▶ $\text{MMR-penalty}(s) = \lambda \max_{s_i \in \mathcal{S}} \text{sim}(s, s_i)$
- Stop when a desired number of sentences are added

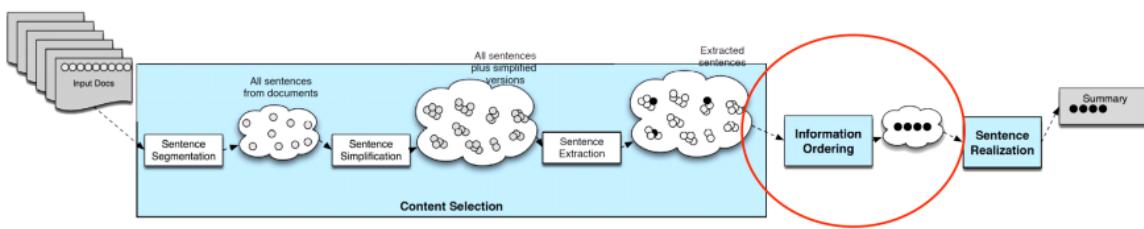
Sentence Simplification

- Create multiple simplified versions of sentences before extraction
- Former Democratic National Committee finance director Richard Sullivan faced more pointed questioning from Republicans during his second day on the witness stand in the Senate's fund-raising investigation
 - ▶ Richard Sullivan faced pointed questioning
 - ▶ Richard Sullivan faced pointed questioning from Republicans during day on stand in Senate fundraising investigation
- MMR to make sure only non-redundant sentences are selected



Information Ordering

- Chronological ordering:
 - ▶ Order by document dates
- Coherence:
 - ▶ Order in a way that makes adjacent sentences similar
 - ▶ Order based on how entities are organised (centering theory, L12)



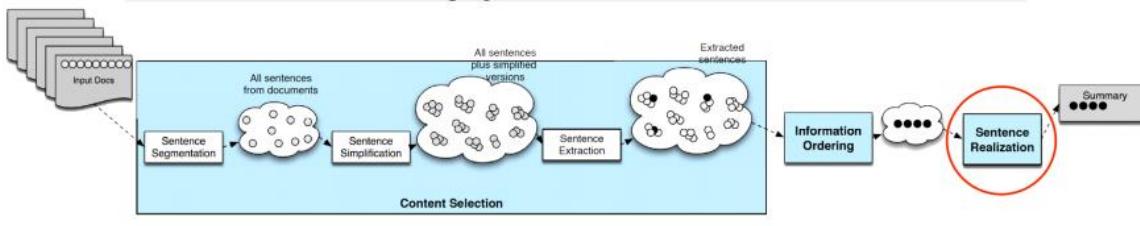
Sentence Realisation

Original summary:

Presidential advisers do not blame **O'Neill**, but they've long recognized that a shakeup of the economic team would help indicate **Bush** was doing everything he could to improve matters. **U.S. President George W. Bush** pushed out **Treasury Secretary Paul O'Neill** and top economic adviser Lawrence Lindsey on Friday, launching the first shake - up of his administration to tackle the ailing economy before the 2004 election campaign.

Rewritten summary:

Presidential advisers do not blame **Treasury Secretary Paul O'Neill**, but they've long recognized that a shakeup of the economic team would help indicate **U.S. President George W. Bush** was doing everything he could to improve matters. **Bush** pushed out **O'Neill** and White House economic adviser Lawrence Lindsey on Friday, launching the first shake-up of his administration to tackle the ailing economy before the 2004 election campaign.



Sentence Realisation

- Make sure entities are referred coherently
 - ▶ Full name at first mention
 - ▶ Last name at subsequent mentions
- Apply coreference methods to first extract names
- Write rules to clean up

Abstractive: Single-Doc

29

Example

*a detained **iranian-american academic** accused of acting against national security has been **released** from a **tehran** prison after a hefty **bail** was posted, a top judiciary official said tuesday*

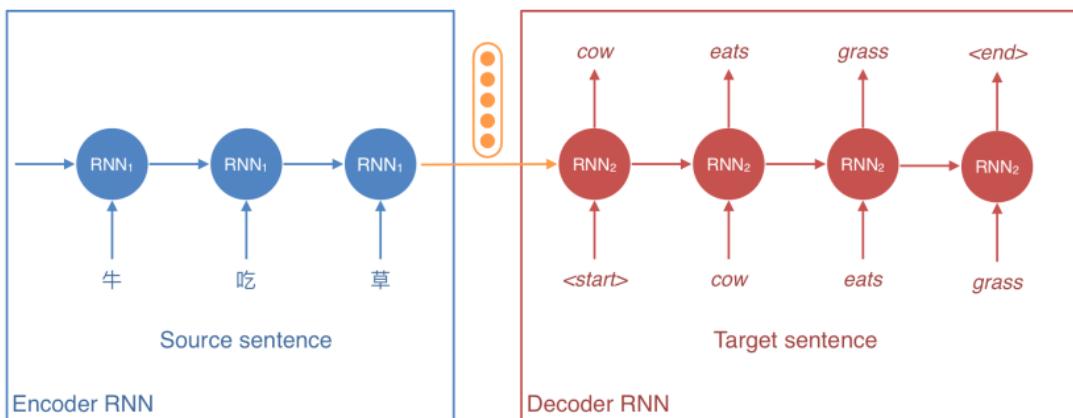


iranian-american academic held in tehran released on bail

- Paraphrase
- A very difficult task
- Can we train a neural network to generate summary?

30

Encoder-Decoder?

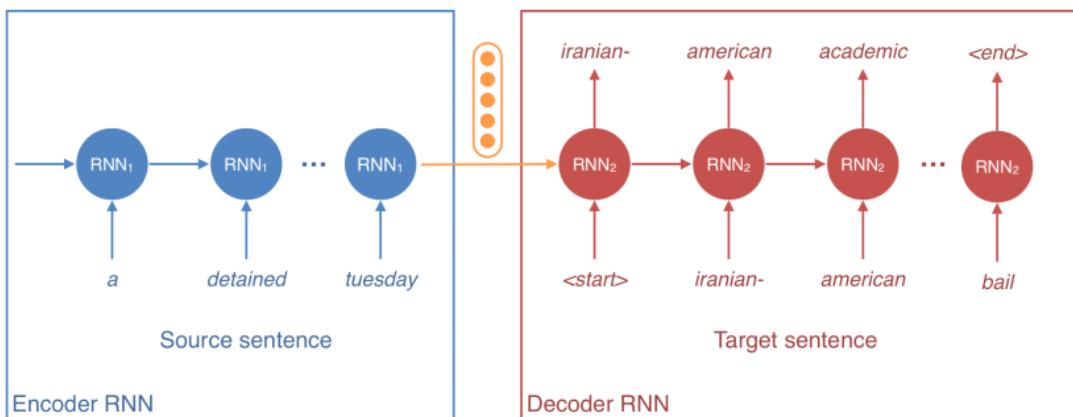


- What if we treat:

- ▶ **Source sentence = “document”**
- ▶ **Target sentence = “summary”**

31

Encoder-Decoder?



a detained iranian-american academic accused of acting against national security has been released from a tehran prison after a hefty bail was posted, a top judiciary official said tuesday

iranian-american academic held in tehran released on bail

32

Data

- News headlines
- Document: First sentence of article
- Summary: News headline/title
- Technically more like a “headline generation task”

33

And It Kind of Works...

I(1): a detained iranian-american academic accused of acting against national security has been released from a tehran prison after a hefty bail was posted , a to p judiciary official said tuesday .

G: iranian-american academic held in tehran released on bail

A: detained iranian-american academic released from jail after posting bail

A+: detained iranian-american academic released from prison after hefty bail

I(2): ministers from the european union and its mediterranean neighbors gathered here under heavy security on monday for an unprecedeted conference on economic and political cooperation .

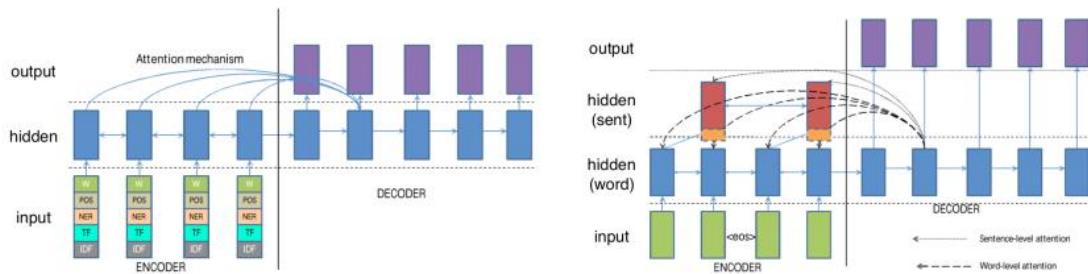
G: european mediterranean ministers gather for landmark conference by julie bradford

A: mediterranean neighbors gather for unprecedeted conference on heavy security

A+: mediterranean neighbors gather under heavy security for unprecedeted conference

Improvements

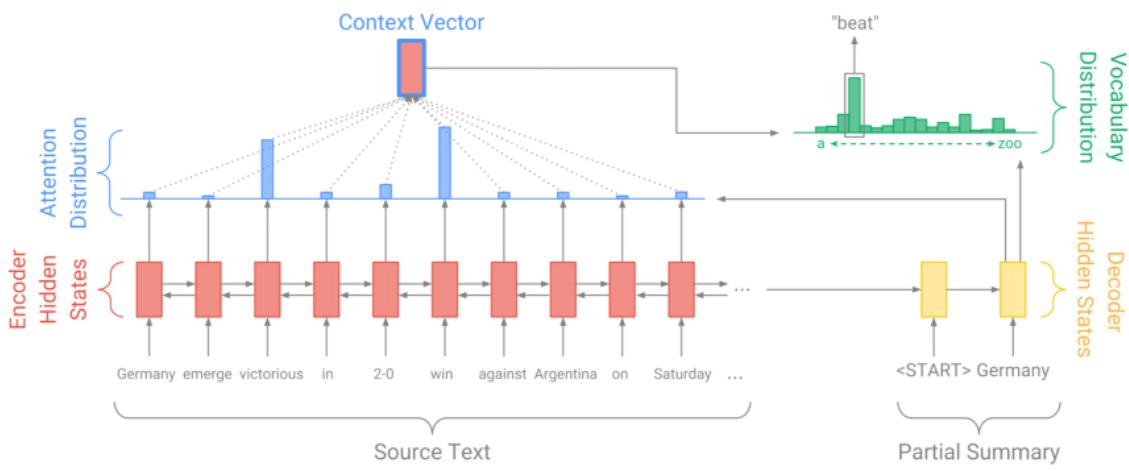
- Attention mechanism
- Richer word features: POS tags, NER tags, tf-idf
- Hierarchical encoders
 - ▶ One LSTM for words
 - ▶ Another LSTM for sentences



Nallapati et al. (2016): Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond 35

Issues

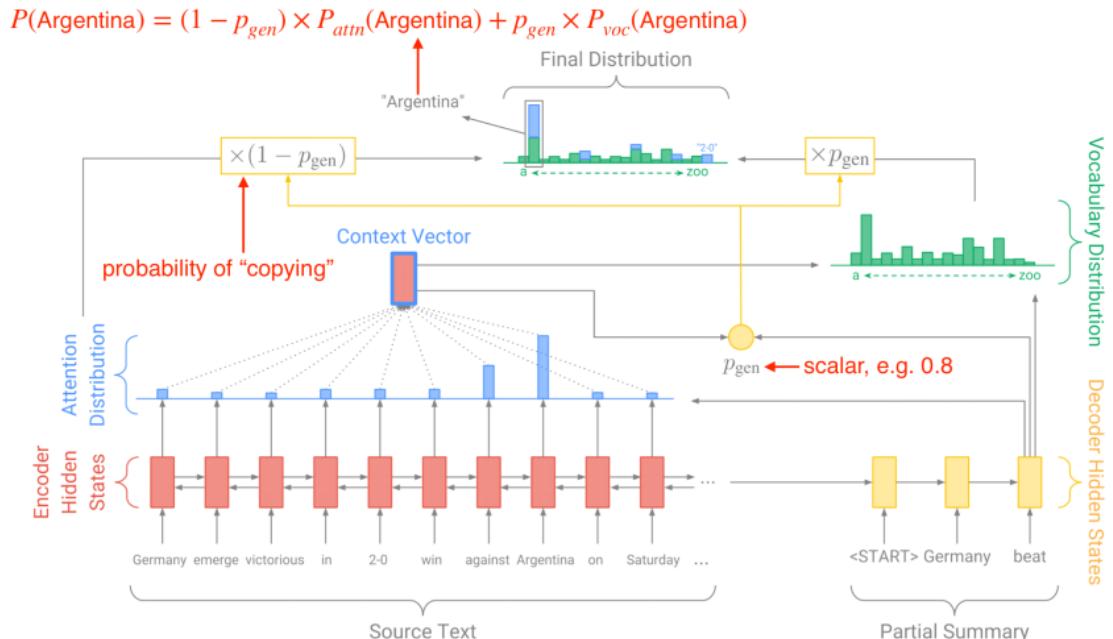
- Occasionally reproduce statements incorrectly
(hallucinate new details!)
- Unable to handle out-of-vocab words in document
 - ▶ Generate UNK in summary
 - ▶ E.g. new names in test documents
- Solution: allow decoder to **copy words** directly
from input document during generation



Encoder-decoder with Attention

See et al. (2017): Get To The Point: Summarization with Pointer-Generator Networks

37



Encoder-decoder with Attention + Copying

See et al. (2017): Get To The Point: Summarization with Pointer-Generator Networks

38

Copy Mechanism

- Generate summaries that reproduce details in the document
- Can produce out-of-vocab words in the summary by copying them in the document
 - e.g. *smerge* = out of vocabulary
 - $p(smerge) = \text{attention probability} + \text{generation probability} = \text{attention probability}$

39

Generated Summaries

Article: smugglers lure arab and african migrants by offering discounts to get onto overcrowded ships if people bring more potential passengers, a cnn investigation has revealed.
(...)

Summary: cnn investigation uncovers the business inside a **human smuggling ring**.

Article: andy murray (...) is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic thiem, who pushed him to 4-4 in the second set before going down 3-6 6-4, 6-1 in an hour and three quarters. (...)

Summary: andy murray defeated dominic thiem 3-6 6-4, 6-1 in an hour and three quarters.

40

More Summarisation Data

- But headline generation isn't really exciting...
- Latest summarisation data:
 - CNN/Dailymail: 300K articles, summary in bullets
 - Newsroom: 1.3M articles, summary by authors
 - Diverse; 38 major publications
 - XSum: 200K BBC articles
 - Summary is more abstractive than other datasets

41

Latest Development

- State-of-the-art models use transformers instead of RNNs
- Lots of pre-training
- Note: BERT not directly applicable because we need a unidirectional decoder (BERT is only an encoder)

42

Evaluation

43

ROUGE

(Recall Oriented Understudy for Gisting Evaluation)

- Similar to BLEU, evaluates the degree of word overlap between generated summary and reference/human summary
- But recall oriented
- Measures overlap in N -grams (e.g. from 1 to 3)
- ROUGE-2: calculates the percentage of bigrams from the reference that are in the generated summary

44

ROUGE-2: Example

$$\text{ROUGE-2} = \frac{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}$$

- Ref 1: Water spinach is a green leafy vegetable grown in the tropics.
- Ref 2: Water spinach is a commonly eaten leaf vegetable of Asia.
- Generated summary: Water spinach is a leaf vegetable commonly eaten in tropical areas of Asia.
- ROUGE-2 = $\frac{3 + 6}{10 + 9}$

45

A Final Word

- Research focus on single-document abstractive summarisation
 - ▶ Mostly news data
- But many types of data for summarisation:
 - ▶ Images, videos
 - ▶ Graphs
 - ▶ Structured data: e.g. patient records, tables
- Multi-document abstractive summarisation

46