# 目录

[**目录**](#_l4zf7dwgfejo) **1**

[**Course Overview**](#_910zvdy02co) **6**

[**索引**](#_tzre8htuvno7) **7**

[**lecture 1**](#_842qabhef6nl) **7**

[Turing Test](#_3ou85by5w6d1) 7

[A brief history of NLP](#_wbr0u4ku6o5o) 8

[Future of NLP](#_tcgcx2h0g2h2) 8

[**lecture 2 text preprocessing**](#_989c7u8pv81s) **8**

[名词解释](#_6ipf4recbahi) 8

[目的](#_co1vd6hi8mmj) 8

[步骤steps](#_c47hswucafuq) 8

[1. Remove unwanted formatting (e.g. HTML)](#_esqlocr6hdgw) 8

[2. Sentence segmentation: break documents into sentences](#_d0j254a1lh0a) 8

[Binary Classifier](#_dv2ao0o0s3fm) 9

[3. Word tokenisation: break sentences into words](#_j3yg34w5izsa) 9

[byte-pair encoding (BPE)](#_8lc5s0tvzlc5) 9

[4. Word normalisation: transform words into canonical forms](#_dj79lvttl2vy) 9

[lemmatisation](#_iyh1wr2xrxwh) 10

[The Porter Stemmer](#_3tuz9ghljcr2) 10

[Fixing Spelling Errors](#_t2gw9pcn0z8) 10

[Other Word Normalisation](#_qccehc6gtaum) 10

[5. Stopword removal: delete unwanted words](#_x7czsk7bv1c1) 11

[**lecture 3 n-gram**](#_9q1sz78ig8nm) **11**

[• Deriving n-gram language models](#_cxoyf4bdy25p) 11

[Markov](#_s1iss4e6x7b3) 11

[Maximum Likelihood Estimation](#_mncdalnykicf) 12

[Several Problems](#_cq3ubs2ydelz) 12

[• Smoothing to deal with sparsity](#_wv9g6ed61rqs) 12

[Laplacian (Add-one) Smoothing](#_wc4tqqno0l5) 13

[Add-k Smoothing(Lidstone Smoothing)](#_dl5yeqq4ty4q) 13

[Absolute Discounting](#_m6gu3eloexei) 13

[Katz Backoff:](#_32m6izpzcmkx) 13

[Kneser-Ney Smoothing](#_t1mkx815d5o4) 13

[Interpolation](#_qwj7coihf164) 14

[Interpolated Kneser-Ney Smoothing](#_l5wo11r0drq1) 14

[• Generating Language](#_9ibg3is2hman) 15

[选择下一个词的方法（遍历方法）](#_e1x5og33118k) 15

[• Evaluating language models](#_kgc8y0kfxv1n) 15

[Perplexity](#_4fwjxakdd6ef) 15

[**lecture 4 text classification**](#_td278lkon8v9) **15**

[**• Text classification tasks**](#_vg34e06bbwmg) **16**

[Topic classification](#_q3nriqh7bra2) 16

[Sentiment Analysis](#_urwiw1p6ypnm) 16

[Authorship Attribution](#_gf4b9rsxa6h) 16

[Native-Language Identification](#_8tpma3y4pjfb) 17

[Automatic Fact-checking](#_nrolelxo7vvd) 17

[• Algorithms for classification](#_5bvyuehqrmdh) 17

[Building a Text Classifier 步骤：](#_uwlt6upi5cto) 17

[Choose a machine learning algorithm](#_gqqoxajnoyjd) 18

[Naïve Bayes](#_lj98p2d2d92l) 18

[Logistic Regression](#_ja17owxgx0fa) 18

[Support Vector Machines](#_v5d8hvd2c8e7) 18

[K-Nearest Neighbour](#_awufxpa6xl44) 18

[Decision tree](#_gekflobnj6cc) 19

[Random Forests](#_sxol0t7jdv3w) 19

[Neural Networks](#_2ac4gw9crzat) 19

[Hyperparameter Tuning](#_8wv3wanb2qb9) 19

[• Evaluation](#_o29w8du1cb2g) 20

[**lecture 5 Part of speech tagging**](#_68bzywjwpmhs) **20**

[Automatic Taggers](#_7tnqjsk7ch1u) 22

[Why Automatically POS tag?](#_e9cpe96jy9z2) 22

[分类](#_bv1fwecxtnop) 22

[• Rule-based taggers](#_9tbjiwl9decw) 22

[• Statistical taggers](#_odpgyfuaprdk) 22

[Unigram tagger](#_e6agz8ld2pnj) 22

[Classifier-Based Tagging](#_v7g22ga44r6z) 22

[Unknown Words](#_6iscet4il253) 23

[**lecture 6 Sequence Tagging: Hidden Markov Models**](#_p2do4ed2d633) **23**

[independence assumptions](#_hl7ztqnwpynq) 23

[The Viterbi Algorithm](#_6ue4fkivsjxx) 25

[实际操作技巧](#_2qf2uzkntnnd) 25

[State-of-theart use tag trigrams and backoffOther Variant Taggers](#_jc3h2r3uixdv) 26

[**lecture 7 deep learning: Feedforward Networks**](#_edm2eys84clm) **26**

[Feed-forward NN](#_kjpyvz4my9x6) 26

[Learning from Data（训练）](#_5qxwuzk4c55e) 27

[应用及具体优化](#_nz2ctnfzd6pc) 28

[Topic Classification](#_rjflwjfktj7w) 28

[Authorship Attribution](#_2xz9iy35iamr) 28

[Language Models](#_ksvyrfm5rfh) 28

[Feed-forward NN for Tagging](#_ch39amyyprwd) 29

[Convolutional Networks](#_oq5p1o27h15s) 29

[Word Embeddings](#_1j4muzhyna5z) 30

[**lecture 8:Recurrent Networks**](#_xfv49fnu04sp) **30**

[RNN](#_337hqqpwv4f0) 30

[backpropagation algorithm：](#_c8573xwrhxgo) 31

[Long Short-term Memory (LSTM)](#_6vohm6im190v) 31

[Variants](#_ike2uwlknpon) 32

[Peephole connections](#_qnb1em5igtqp) 33

[Gated recurrent unit (GRU)](#_2rqv7b64td4n) 33

[**lecture 9:Lexical Semantics**](#_d0d6m23xuhig) **33**

[lexical database](#_esp8swso5qnp) 33

[Meaning Through Relations](#_wccdn65bz81k) 33

[Word Similarity](#_b8g0uilmaz0v) 34

[include depth information](#_5slmqiikky5v) 34

[Concept Probability](#_q99034nqpbya) 35

[Word Sense Disambiguation](#_irxnirolabr6) 35

[Supervised WSD](#_6r5fp8a2ofx) 35

[Other Databases - FrameNet](#_ukjpbaha1wxv) 36

[**lecture 10：Distributional Semantics**](#_fxkvgvyqeper) **36**

[Guessing Meaning from Context](#_hdopmo5qjl30) 36

[Count-Based Methods](#_r9cxaeab3mo9) 36

[The Vector Space Model](#_zdpsowboguaq) 36

[Tf-idf](#_v88pvgki999f) 37

[Dimensionality Reduction](#_xvb6jg9e2w9s) 37

[Singular Value Decomposition](#_5dh2nuy9kkbl) 37

[Truncating – Latent Semantic Analysis](#_f93rbbquy6oc) 37

[Words as Context](#_1wtmk5uqp152) 37

[Pointwise Mutual Information](#_4vcihn98ds9t) 38

[Similarity（based vector）](#_nwacetpd41ks) 38

[Neural Methods](#_oigglvhpgf1p) 38

[Word Embeddings](#_j0jtrd8tw5y8) 38

[Word2Vec](#_bk4sy56pyq4) 39

[Skip-gram Model](#_hgnwy15by29k) 39

[Evaluating Word Vectors](#_2vmc0qddk1z7) 40

[**lecture 11:Contextual Representation**](#_c6ezstctt6fm) **40**

[ELMo](#_psome0ed29kt) 40

[BERT](#_pokxsltvodkw) 42

[BERT vs. ELMo](#_684i90m99ma0) 43

[**lecture 12:Discourse**](#_i9cntbflv8qa) **44**

[Discourse segmentation](#_u5ynlzkxloy5) 44

[Unsupervised Approaches](#_vy01ch5j5jjb) 44

[Supervised Approaches](#_tqosdueyde95) 44

[Discourse Parsing](#_ldmu25xn7bsv) 45

[Rhetorical Structure Theory (RST)](#_cluuhrb74g8w) 45

[Parsing Using Machine Learning](#_uqe8yosf8t8k) 46

[Anaphora Resolution](#_12c4mdbokit5) 46

[Centering Theory](#_ull2tpqx57v3) 47

[Centering Algorithm](#_b4o2y4qsvrls) 47

[Supervised Anaphor Resolution](#_2152p2d2y9v3) 47

[Anaphora Resolution Tools](#_rzwdycu3hiy0) 48

[**lecture 13:Formal Language Theory & Finite State Automata**](#_xb2ric1ixp9a) **48**

[Regular Languages](#_jhf0oya80x91) 48

[Finite state acceptors](#_rlif7vmp8x71) 48

[Weighted FSA](#_etkroimq09m4) 49

[N-gram LMs as WFSA](#_v0xiasxaetp5) 50

[Finite State Transducer（FST)](#_ommxq0xv7ftt) 51

[FST Composition](#_58nrjhhvwqh5) 51

[**lecture 14:Context-Free Grammar**](#_71lra11pw53) **51**

[Syntactic Constituents](#_behwb4vtkehg) 52

[CFG](#_1wf3r2z9wcj) 52

[Parsing CFG](#_xknoym61z58v) 52

[CYK Algorithm](#_kkuh2rat154h) 52

[Chomsky Normal Form](#_51wdg1e4tv2b) 53

[CNF (cont)](#_g08dbm2lxt4f) 53

[Representing English with CFGs](#_6zot5dt19jwt) 54

[**lecture 15:Probabilistic Context-Free Grammar**](#_f5l7ld5brbl) **54**

[Probabilistic context-free grammars (PCFGs)](#_rdciv2gokdo1) 54

[Stochastic Generation with PCFGs](#_slpuu3lbtfcj) 54

[Parsing PCFGs](#_mmcet62lwb5k) 55

[CNF](#_tkehquu8fhfj) 55

[CYK](#_6psf0rliz9iz) 55

[Issues with PCFG](#_je5cjy6uzllf) 56

[Solution: Parent Conditioning](#_y8m8z81q5j94) 56

[Solution: Head Lexicalisation](#_l2j2wxm51x0a) 57

[**lecture 16:Dependency Grammars**](#_cgibwdq4epsk) **57**

[Projectivity](#_k3thrjseslo7) 58

[Parsing](#_y6bg1pqo7vtx) 58

[Transition-Based Parsing](#_4sbaobzcfb4r) 59

[Intuition](#_p9w19e46oy4a) 59

[步骤](#_qjrd1iymzbdr) 59

[Dependency Labels](#_umtt7bl6of2m) 60

[Parsing Model](#_uvjcgqb2i627) 60

[Graph-Based Parsing](#_3hs4l0ro6657) 60

[**lecture 17:Machine Translation**](#_apbk2khl0wc0) **61**

[Statistical Machine Translation](#_t8fe7x2el7us) 62

[Rule-based system](#_h2xuyfug9n1a) 62

[Alignment](#_b0qbt3l1uk0i) 62

[Neural Machine Translation](#_cifdcbmboxgm) 63

[Neural MT Training Loss](#_v9srdsdv5lu0) 64

[Decoding at Test Time](#_1xbtwesq5qfo) 64

[• argmax(greedy decoding): take the word with the highest probability at every step](#_3ie6trf47bb1) 65

[Beam Search Decoding](#_s17cob9juayn) 65

[Attention](#_lo1yek3kinir) 66

[MT Evaluation](#_dalznxtbkeio) 67

[**lecture 18 Information Extraction**](#_42e0ixbhm67d) **68**

[Named Entity Recognition](#_brfpceqa6ce) 68

[IO tagging](#_x2kjk0fz2qq7) 68

[IOB tagging](#_khb5wlh331h) 69

[NER as Sequence Labelling](#_dme1fma1ez3t) 69

[Relation Extraction](#_1pgxziw3ouxk) 69

[Rule-Based Relation Extraction](#_34tzuh3jwpp6) 70

[Supervised Relation Extraction](#_1pup73skrsps) 70

[Semi-supervised Relation Extraction](#_ykjrfibrtzyr) 71

[Distant Supervision](#_dj1t8vmnxcx) 71

[Unsupervised Relation Extraction](#_ky5vh5n4k2bu) 72

[Evaluation](#_uz7ke5vh5vr2) 72

[**lecture 19:Question Answering**](#_cwjslysjfshu) **72**

[Information retrieval-based QA](#_g9z4vvylqzf4) 72

[Question Processing](#_ne544o5ieq7o) 72

[Answer Types](#_nf4yv17jov7n) 72

[Retrieval](#_pnqszltrkrvl) 73

[Answer Extraction](#_bw4nyufezgx5) 73

[Feature-Based Answer Extraction](#_5h9jgnm8at3y) 73

[Neural Answer Extraction](#_oz92lcihfmea) 73

[Knowledge-Based QA](#_f5dv88dp0q2y) 74

[IBM’s WATSON](#_kiuhn65aiwem) 75

[QA Evaluation](#_6ostmsi4vmne) 75

[**lecture 20:Topic Modelling**](#_jn65s45orw78) **75**

[A Brief History of Topic Models](#_1jqg7eva5zej) 76

[Latent Semantic Analysis (L10): SVD+Truncate](#_c2s9cd2pjc8c) 76

[Probabilistic LSA](#_mfo33qbeewbc) 76

[Latent Dirichlet Allocation](#_xax2l8i0mgh) 76

[Sampling Method (Gibbs)](#_g73l6g49it02) 77

[Evaluation](#_ktex8720g00a) 78

[Intrinsic evaluation:](#_t9o41yp1ygvm) 78

[Topic Coherence](#_124nizzgvr1i) 78

[Word Intrusion](#_66a5fthy8r3m) 79

[PMI](#_edgjj57fyat6) 79

[Topic Model Variants](#_ehcaygktvrke) 80

[**lecture 21:Summarisation**](#_bwzapwlm2y4c) **80**

[Extractive: Single-Doc](#_g8y2hinmntby) 81

[Content Selection](#_80wj2w6g3trh) 81

[Method 1: TF-IDF](#_h8m8pejl8it8) 81

[Method 2: Log Likelihood Ratio](#_u7giuvvrzxxx) 81

[Method 3: Sentence Centrality](#_9jwfbwyfuu54) 82

[Method 4: RST Parsing](#_6sgyku8kskyr) 82

[Extractive: Multi-Doc](#_el1da5jan5nb) 82

[MMR](#_tsphts1ctuar) 83

[Information Ordering](#_s4z521gx2nkr) 83

[Sentence Realisation](#_vrlj6jbreulu) 83

[Abstractive: Single-Doc](#_5x3sq1am6710) 84

[Improvements](#_83eoxhxa3v8s) 84

[Copy Mechanism](#_pl39w8gero4c) 84

[Evaluation](#_cydh4d26avc7) 85

# Course Overview

• Word, sequences, and documents

• Text preprocessing

• Language models

• Text classification

• Structure learning

• Sequence tagging (e.g. part-of-speech)

• Deep learning for NLP

• Feedforward and recurrent models

• Semantics

• How words form meaning

• Syntax

• How words are arranged

• Applications

• Machine translation

• Information extraction

• Question answering

# 索引

# lecture 1

## Turing Test

• 3 participants: 2 humans and a computer

• One of the humans is an interrogator

• Test involves a conversation between 2 parties

• The role of the interrogator is to determine which participant is the machine by asking a series of questions

• Machine is intelligent if it can fool the interrogator into thinking that he/she is talking to a human

## A brief history of NLP

## Future of NLP

# lecture 2 text preprocessing

## 名词解释

• 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.不考虑是否重复

• Word type: the distinct word “the”. 重复的只算一个

‣ Lexicon (“dictionary”): a group of word types.

## 目的

language is **compositional（隔断的）**. As humans, we can break these documents into individual components. To understand language, a computer should do the same

## 步骤steps

### 1. Remove unwanted formatting (e.g. HTML)

### 2. Sentence segmentation: break documents into sentences

Naïve approach: break on sentence punctuation ([.?!])

问题： But periods are used for abbreviations!（缩略语需要被字符隔开）

Second try: use regex to require capital ([.?!] [A-Z])

问题：‣ But abbreviations often followed by names (Mr. Brown)

Better yet: have lexicons（字典）

问题：‣ But difficult to enumerate all names and abbreviations

#### Binary Classifier

•方法：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:

- Determiners tend to start a sentence

### 3. Word tokenisation: break sentences into words

优点：Easier for machine to understand.

不同语言不同

较好的方法：Subword Tokenisation

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

操作过程：

在dictionary中的次，统计词频，然后综合来找频率较高的单词或者单词组合，然后将较高的加入vocabulary，直到所有大于一的都被加入。

1. Break the entire piece of text into single characters tokens.

2. Count frequency of two tokens being together.

3. Merge most frequent pair of characters into one token.

4. Repeat from step 2.

结果：

• 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

### 4. Word normalisation: transform words into canonical forms

步骤包括：

• Lower casing (Australia → australia)

• Removing morphology（形态）

• Correcting spelling

• Expanding abbreviations扩展缩写 (U.S.A → USA)

目的：

‣ Reduce vocabulary

‣ Maps words into the same type

Inflectional morphology creates grammatical variants不同形态的变体，不同语言都有很多

#### lemmatisation

Inflectional morphology creates grammatical variants不同形态的变体，不同语言都有很多

**Lemmatisation** means removing any inflection to reach the uninflected form, the lemma

（目的就是将变形变回去）A lexicon of lemmas needed for accurate lemmatisation

Derivational Morphology：衍生词

Remove all inflections

Matches with lexicons

Product: Lemma

#### The Porter Stemmer

**Stemming** strips off all suffixes, leaving a stem（将衍生词变回最基本的词）

Remove all suffixes

No matching required

Product: Stem

‣ First strip inflectional suffixes

‣ Then derivational suffixes

• c (lowercase) = consonant; 辅音

• v (lowercase) = vowel;元音

• C = a sequence of consonants

• V = a sequence of vowels

所有的词都可以转化为[C] (VC)m [V] m = measure

Step 1: plurals and past participles

根据rules去除附数形式

Step 2, 3, 4: derivational inflections

Step 5: tidying up

实际操作中，针对m的条件指的是如果这样操作，剩下的应该满足的条件

#### 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

#### 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

### 5. Stopword removal: delete unwanted words

• Definition: a list of words to be removed from the document

‣ Typical in bag-of-word (BOW) representations

缺点：‣ Not appropriate when sequence is important

• How to choose them?

‣ All closed-class or function words - E.g. the, a, of, for, he, …

‣ Any high frequency words

‣ NLTK, spaCy NLP toolkits

# lecture 3 n-gram

重点为ngram计算

We measure ‘goodness’ using probabilities estimated by language models

• Language model can also be used for generation

Language Models Useful for

‣ Speech recognition

‣ Spelling correction

‣ Query completion

‣ Optical character recog.

Other generation tasks

‣ Machine translation

‣ Summarisation

‣ Dialogue systems

## • Deriving n-gram language models

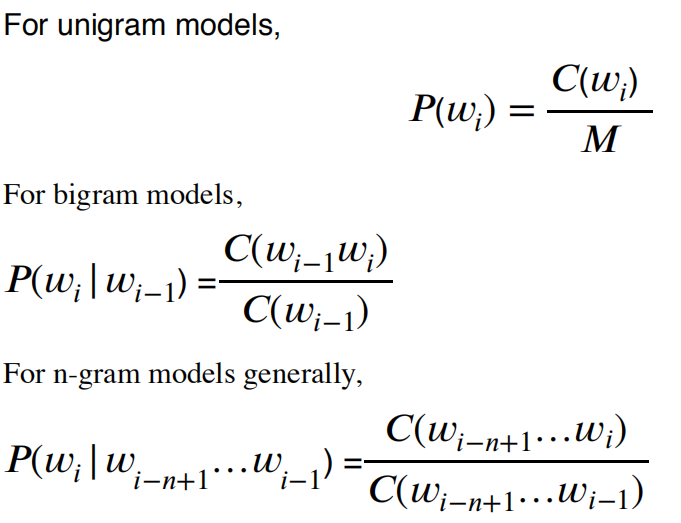
‣ <s> = sentence start

‣ </s> = sentence end

### Markov



### Maximum Likelihood Estimation



### Several Problems

• Language has long distance effects — need large n

• 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

‣ Need to smooth the LM!

## • Smoothing to deal with sparsity

• Basic idea: give events you’ve never seen before some probability. Must be the case that P(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…

### Laplacian (Add-one) Smoothing

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

给每个频率都加1，即计算概率是分子加1，分母加|V|

### Add-k Smoothing(Lidstone Smoothing)

给每个频率都加k，即计算概率是分子加1，分母加k|V|(Adding one is often too much，k一般小于1，需要选择）

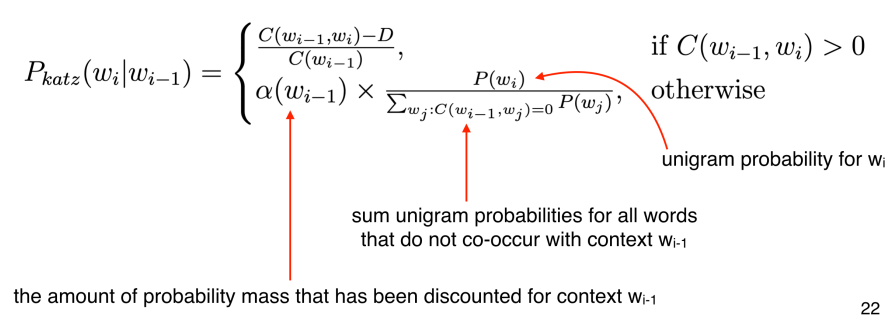
### Absolute Discounting

•‘Borrows’ a fixed probability mass from observed n-gram counts

从有频率的词，给没有出现过的词。每个出现过的词频率减少d，然后平均分给没有出现过的。(unseen)

### Katz Backoff:

不平均分配借来的频率，而是根据低一级的n-gram来判断重新分配

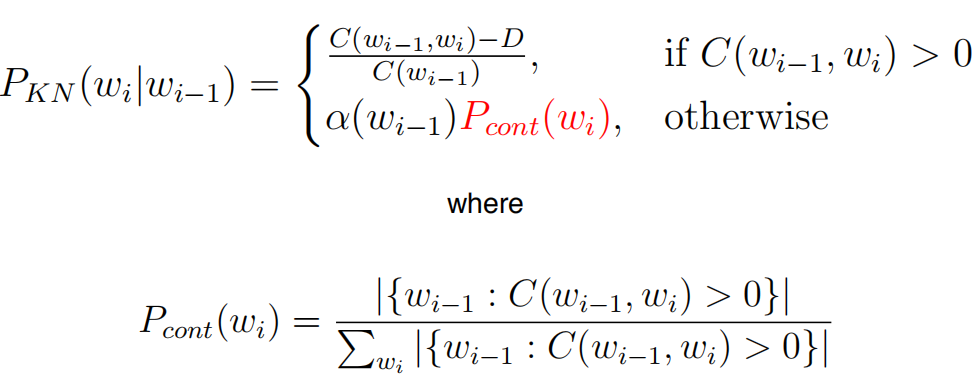


Issues with Katz Backoff：没有根据后一个词分配，都是一样的。

### Kneser-Ney Smoothing

• Redistribute probability mass based on how many number of different contexts word w has

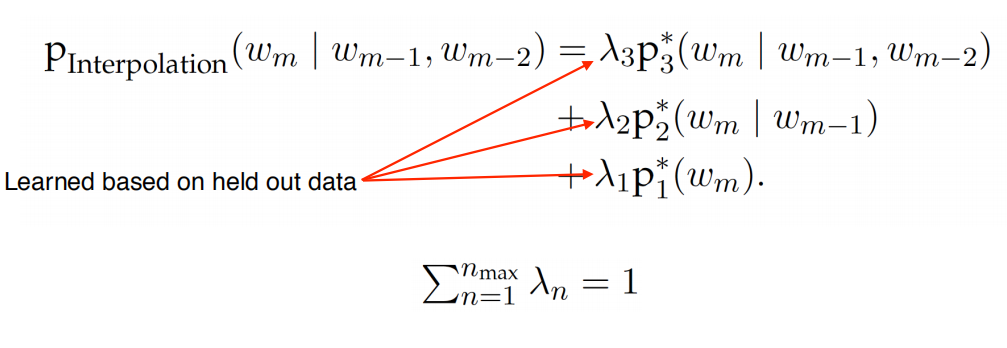
appeared in. This measure is called “continuation probability”



### Interpolation

• A better way to combine different order n-gram models

• Weighted sum of probabilities across progressively shorter contexts



### Interpolated Kneser-Ney Smoothing

**Commonly used Kneser-Ney language models use 5-grams as max order**

## • Generating Language

Given an initial word, draw the next word according to the probability distribution defined by the language model.

也会产生</s>来判断句子是否终结。

根据频率判断下一个词可能性。

### 选择下一个词的方法（遍历方法）

• 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)

## • Evaluating language models

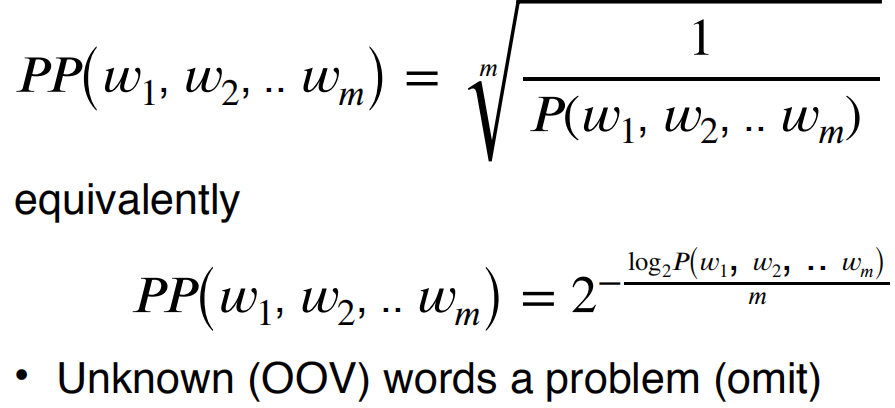
外在方法：spelling correction, machine translation

自身方法：Perplexity on held-out test set

### Perplexity

• Inverse probability of entire test set

• The lower the better



# lecture 4 text classification

• Input

‣ A document d (Often represented as a vector of features)

‣ A fixed output set of classes C = {c1,c2,…ck} • Categorical

• Output

‣ A predicted class c ∈ C

# • Text classification tasks

应用例子：

‣ Topic classification

‣ Sentiment analysis

‣ Authorship attribution

‣ Native-language identification

‣ Automatic fact-checking

通常句子或者tweet的比整篇文章的更常用。

challenging?

How to learn document representation?

How to do feature selection?

How to deal with data sparsity?

### Topic classification

• 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

### Sentiment Analysis

• Motivation: opinion mining, business analytics

• Classes: Positive/Negative/(Neutral)

• Features

‣ N-grams

‣ Polarity lexicons（词的极性+-）

• Examples of corpora

‣ Polarity movie review dataset (in NLTK)

‣ SEMEVAL Twitter polarity datasets

### Authorship Attribution

• Motivation: forensic linguistics, plagiarism detection(剽窃抄袭）

• Classes: Authors (e.g. Shakespeare)

• Features

‣ Frequency of function words

‣ Character n-grams

‣ Discourse （语言）structure

• Examples of corpora

‣ Project Gutenberg corpus (see NLTK sample)

### Native-Language Identification

• Motivation: forensic linguistics, educational applications

• Classes: first language of author (e.g. Chinese)

• Features

‣ Word N-grams

‣ Syntactic（句法） patterns (POS, parse trees)

‣ Phonological features

• Examples of corpora

‣ TOEFL/IELTS essay corpora

### Automatic Fact-checking

• Motivation: social media, journalism (fake news)

• Classes: True/False/(Can’t be sure)

• Features

‣ N-grams

‣ Non-text metadata

• Examples of corpora

‣ Emergent, LIAR: political statements

‣ FEVER

## • Algorithms for classification

### Building a Text Classifier 步骤：

1. Identify a task of interest

2. Collect an appropriate corpus

3. Carry out annotation

4. Select features

5. Choose a machine learning algorithm

• Bias vs. Variance

‣ Bias: assumptions we made in our model

‣ Variance: sensitivity to training set

• Underlying assumptions, e.g., independence

• Complexity

• Speed

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

这里重点第五步模型选择和第六步超参数调整，和evaluate

• 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

### Choose a machine learning algorithm

#### Naïve Bayes

#### 

• Pros: Fast to “train” and classify; robust, lowvariance; good for low data situations; optimal

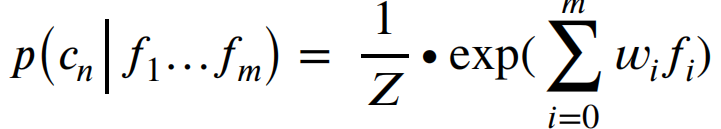
classifier if independence assumption is correct; extremely simple to implement.

• Cons: Independence assumption rarely holds; low accuracy compared to similar methods in most situations; smoothing required for unseen class/feature combinations

#### Logistic Regression

• A linear model, but uses softmax“squashing” to get valid probability

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



• 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

#### 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

#### K-Nearest Neighbour

• Classify based on majority class of k-nearest training examples in feature space

• Definition of nearest can vary

‣ Euclidean distance :d(A,B) √∑(ai − bi )2 但是容易受到长度影响

‣ Cosine distance Cosine similarity cos(A,B) =A B/∣∣A∣∣ ∣∣B∣

• 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

#### Decision tree

#### 

Tends to prefer rare features that might only appear in a few documents.

• 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

#### 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

#### Neural Networks

• An interconnected set of nodestypically 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

### 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

‣ Used to prevent overfitting

• For multiple hyperparameters, use grid search

### • Evaluation

Accuracy = correct classifications/total classifications

Precision = correct classifications of B (tp) / total classifications as B (tp + fp)

Recall = correct classifications of B (tp)/ total instances of B (tp + fn)

F1 = 2\*precision\*recall/(precision + recall)

• Like precision and recall, defined relative to a specific positive class

• But can be used as a general multiclass metric(优点)

‣ Macroaverage: Average F-score across classes

‣ Microaverage: Calculate F-score using sum of counts

# lecture 5 Part of speech tagging

定义：Label of word’s grammatical (primarily syntactic) properties of in the sentence.

AKA word classes, morphological classes, syntactic categories

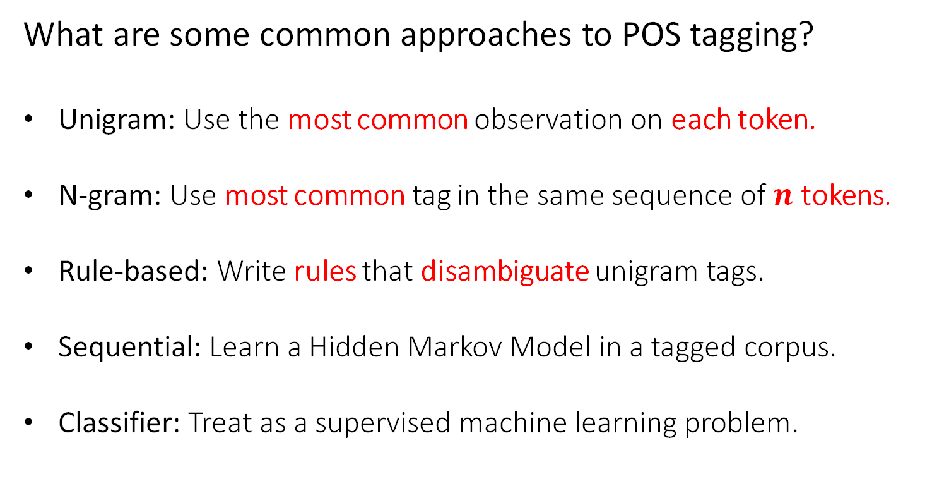
POS tells us quite a bit about a word and its neighbours:

‣ nouns are often preceded by determiners

‣ verbs preceded by nouns

‣ content as a noun pronounced as CONtent

‣ content as a adjective pronounced as conTENT



因为简单的BOW不能反映出句子的结构，所以对于很多应用是不足够的。

POS Closed Classes (English)

• Prepositions (in, on, with, for, of, over,…)

• Particles ‣ brushed himself off

• Determiners‣ Articles (a, an, the) ‣ Demonstratives (this, that) ‣ Quantifiers (each, every)

• Pronouns ‣ Personal (I, me, she,…) ‣ Possessive (my, our,…) ‣ Interrogative or (who)

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

**Ambiguity歧义或有多种可能，比如很多词都有多种tags**

**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)

## Automatic Taggers

### 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 vs cross/VB cross/JJ

• Can use them to create larger structures (parsing)

### 分类

#### • Rule-based taggers

• Typically starts with a list of possible tags for each word from a lexical resource, or a corpus

• Often includes other lexical information

• 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

#### • Statistical taggers

‣ Unigram tagger

‣ Classifier-based taggers

‣ Hidden Markov Model (HMM) taggers

##### 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

• Often considered the baseline for more complex approaches

##### Classifier-Based Tagging

• Use a standard discriminative classifier (e.g. logistic regression, neural network)

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(缺点)

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,优点：**

‣ Less sparsity

‣ Fast algorithms for sequential prediction, i.e. finding

the best tagging of entire word sequence

#### Unknown Words

• Can use things we’ve seen only once to best guess for things we’ve never seen before

• Can use sub-word representations to capture morphology

# lecture 6 Sequence Tagging: Hidden Markov Models

原来tag模型的问题Problems:

‣ Exponentially many combinations: |Tags|M ,for length M

而且人们判断词性其实是根据整个句子来判断的

**定义:**

‣ 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.

**Main drawback:** not very flexible in terms of feature representation, compared to MEMMs and CRFs.

1.Markov assumption

the likelihood of transitioning into a given state depends only on the current

state, and not the previous state(s) (or output(s))

2.Output independence assumption

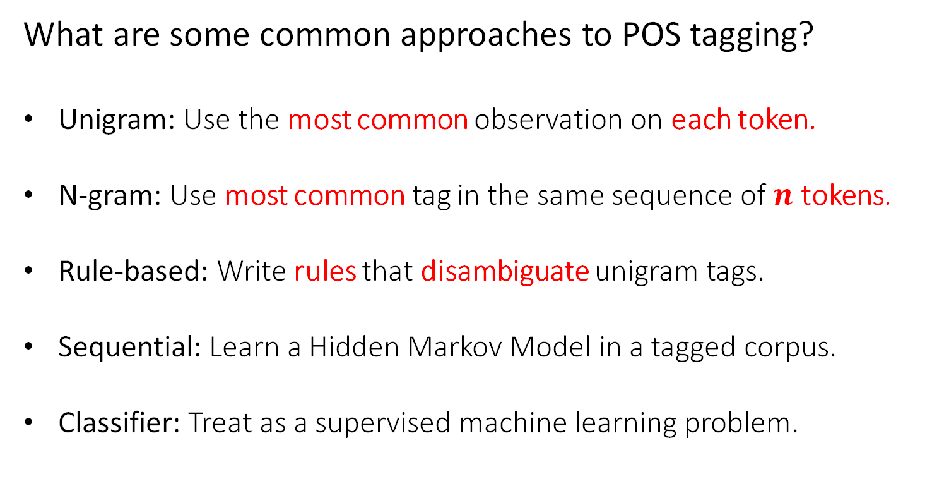
the likelihood of a state producing a certain word (as output) does not depend

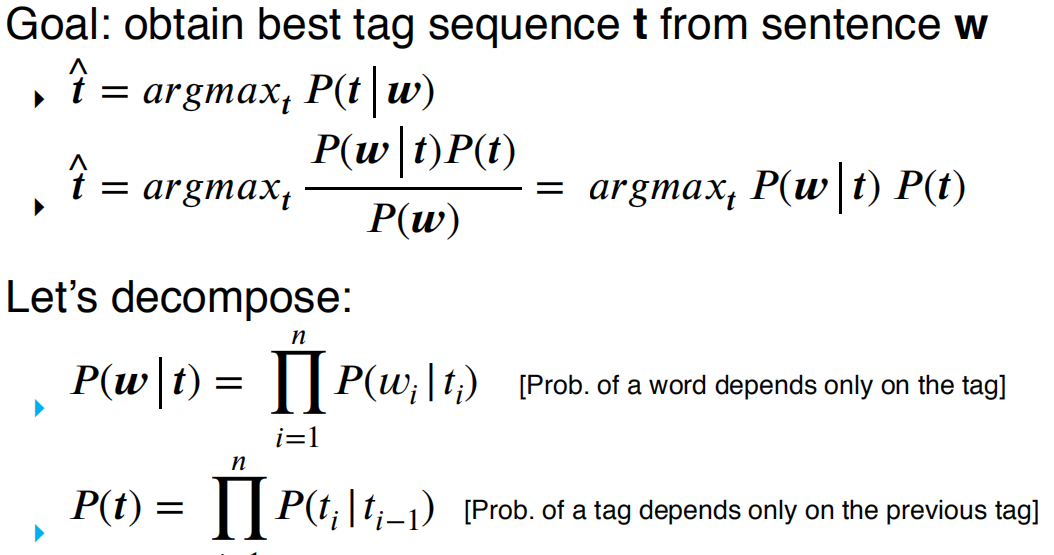
on the preceding (or following) state(s) (or output(s)).

## independence assumptions

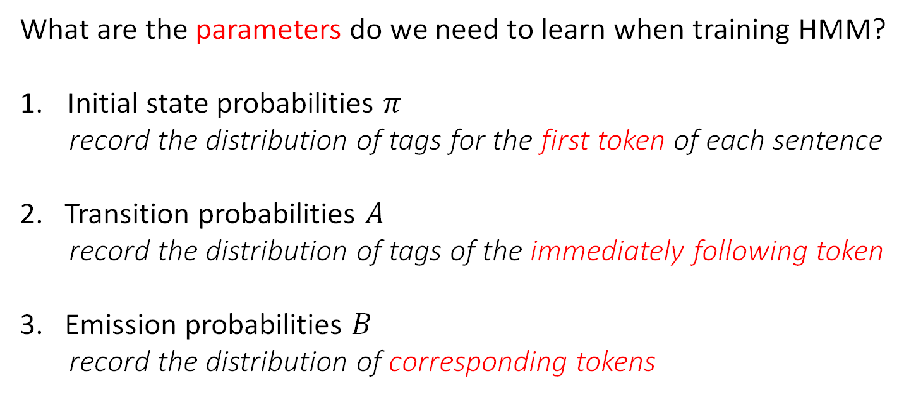
a Hidden Markov Model (HMM) ：probability of an event (tag) depends only on the previous event (last tag)

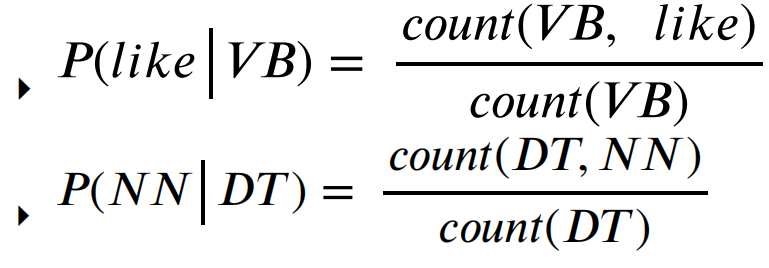
Hidden：Because the events (tags) are not seen: goal is to find the best sequence





emission (O) and transition (A) probabilities first tag is <s> </s>the end of sentence

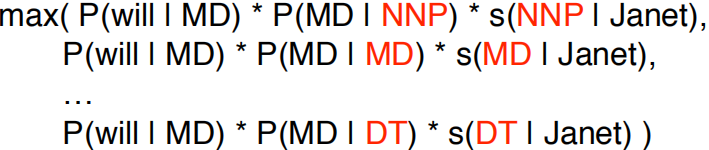
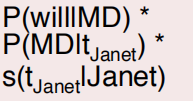
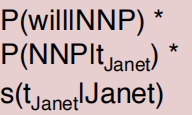




greedy: take all possible tag combinations, evaluate them, take the max (like Naïve Bayes)

## The Viterbi Algorithm

在每一步，上一步的词取各种情况，和这一步的词取各种tag的情况都考虑进去，只留下非零的即可。同时记录路径，等到句子结束，回溯概率最大路径



O(T2N), where T is the size of the tagset and N is the length of the sequence.

### 实际操作技巧

work with log probabilities to prevent underflow (multiplications become sums)

Vectorisation (use matrix-vector operations)

### State-of-theart use tag trigrams and backoffOther Variant Taggers

**HMM is generative**

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

**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

# lecture 7 deep learning: Feedforward Networks

Deep Learning定义：

• 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

• Many layers are chained together in modern deep learning models

## Feed-forward NN

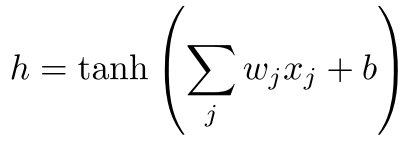
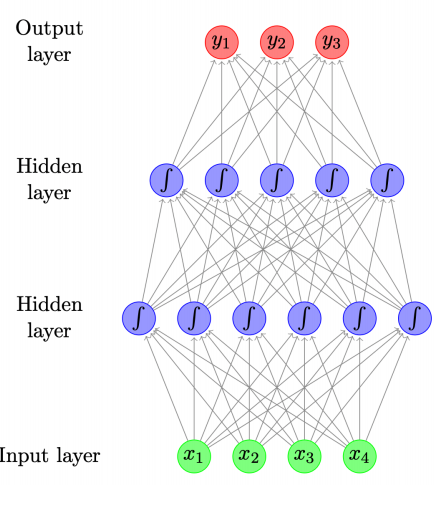
• Aka multilayer perceptrons

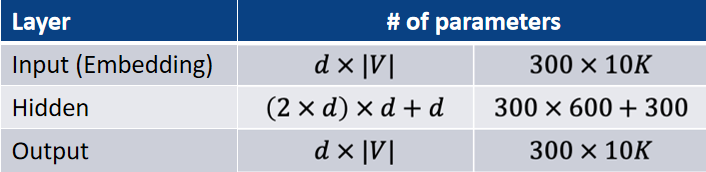
• Each arrow carries a weight, reflecting itsimportance

• Sigmoid function represents a non-linear function

For a unit, it’s a function. applies a non-linear function, such as logistic sigmoid, hyperbolic sigmoid (tanh), or rectified linear unit.

input x, weight w and offset bias b to compute the value of h. x, w, b, h 也可以用matrix代表。





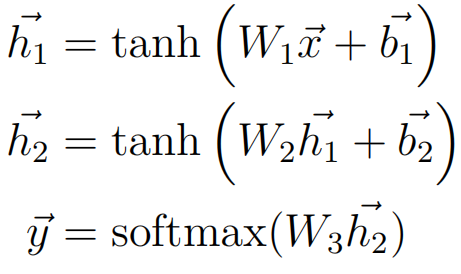
Output layer:

二分类问题：

‣ sigmoid activation function (aka logistic function)

多分类问题：

‣ softmax ensures probabilities > 0 and sum to 1



**优缺点：**

‣ Robust to word variation, typos, etc

‣ Excellent generalization

‣ Flexible — customised architecture for different tasks

RNN can capture longer context, whereas the context size of N-gram LM is fixed.

(与ngram相比）

• 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

## Learning from Data（训练）

learn the parameters from data：

how well the model “fits” the training data, in terms of the probability it assigns to the correct output

‣ 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

## 应用及具体优化

### Topic Classification

• x = [0, 2, 3, 0] for the first document

• y = [0.1, 0.6, 0.3]: probability distribution over the 3 classes

• + 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)

### Authorship Attribution

• 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

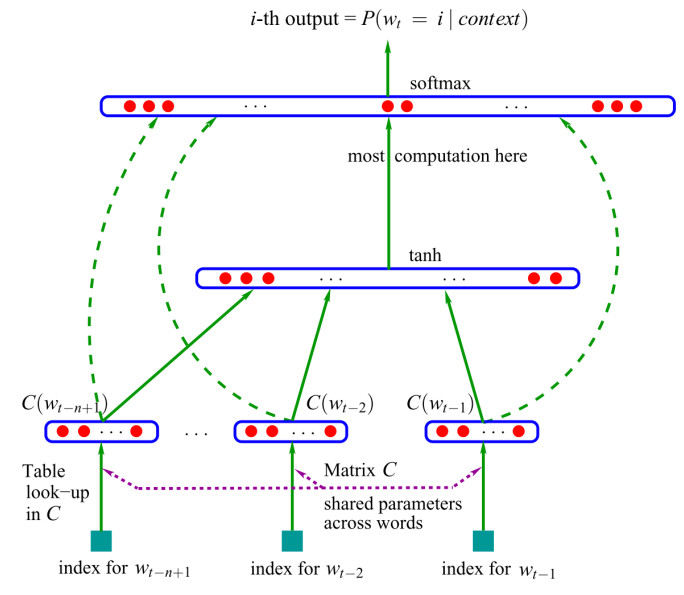
### Language Models

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

P(

Use neural network as a classifier to model

‣ input features = the previous two words ‣ output class = the next word



user word embeddings to represent words

Most parameters are in the word embeddings (size = d x |V|) and the output embeddings (size = |V| x d)

缺点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)

• 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)

### Feed-forward NN for Tagging

• MEMM tagger takes as input:

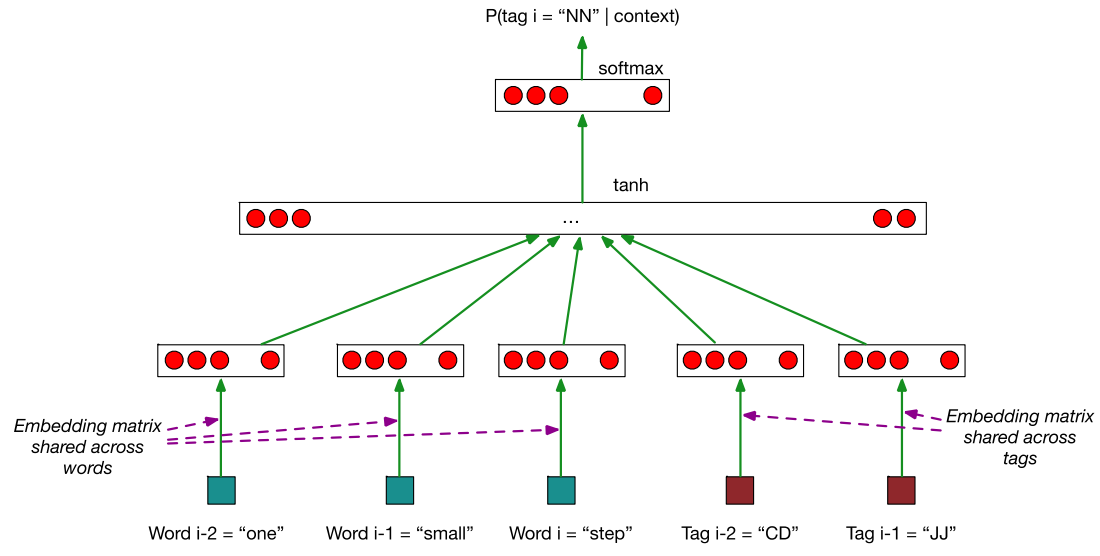
‣ recent words

‣ recent tags

• And outputs: current tag

• Frame as neural network with 5 inputs: 3 x word embeddings and 2 x tag embeddings

‣ 1 output: vector of size |T|, using softmax



### Convolutional Networks

• Commonly used in computer vision

• Identify indicative local predictors

• Combine them to produce a fixed-size representation

convolution filter :linear transformation+tanh

max-pool to produce a fixed-size representation

## 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

**the first layer of neural network**

# lecture 8:Recurrent Networks

N-gram Language Models 回顾

• Can be implemented using counts (with smoothing)

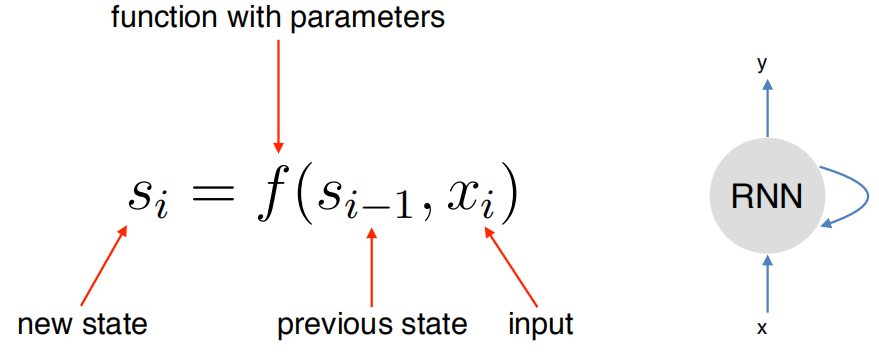
• Can be implemented using feed-forward neural networks

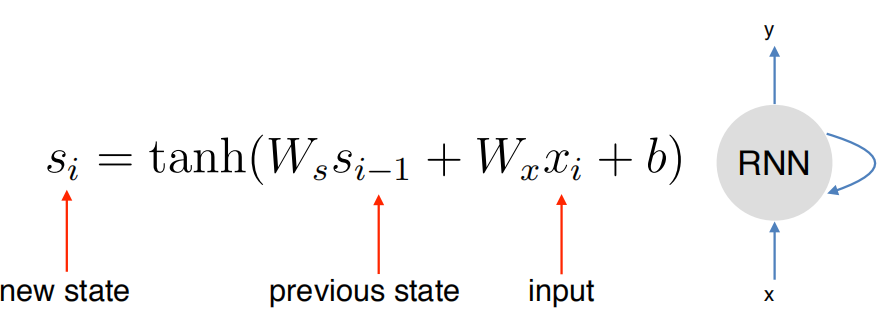
Problem: limited context

## RNN

• 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



yi = σ(Wysi)

RNN for Language Model

• Input words mapped to an embedding

• Output = next word

简单模型的问题vanishing gradients”

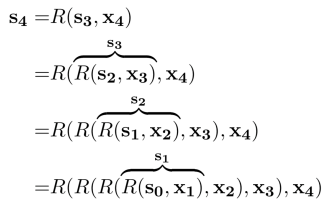
• Gradients in later steps diminish quickly during backpropagation

• Earlier inputs do not get much update

## backpropagation algorithm：

used for trainning RNN.

To train RNN, we just need to create the unrolled computation graph given an input sequence



• 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)Vanishing Gradient Problem

## 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” g

•For each input, a gate decides:

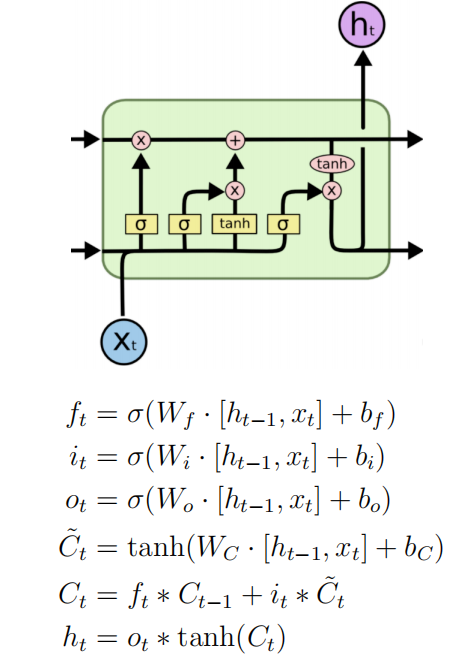
‣ how much the new input should be written to the memory cell

‣ and how much content of the current memory cell should be forgotten

• 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 （g保持元素在01之间，然后通过与v相乘，来决定v的保留和删除）

• Use sigmoid function to keep values of g close to either 0 or 1



Forget Gate：公式第一行，图例上最左边的。Controls how much information to “forget” in the memory cell (Ct-1)

Input Gate：第二个第四个公式，图例中间两个。Input gate controls how much new information to put to memory cell

Update Memory Cell：公式第五个，图例最上面两个

Use the forget and input gates to update memory cell

Output Gate：第三第六个公式，图例最右边Output gate controls how much to distill the

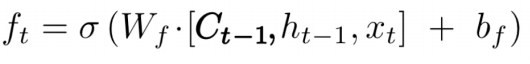
content of the memory cell to create the next state (ht)

然后ct和ht将传入下一个阶段。而ht也是这个阶段的结果。

## Variants

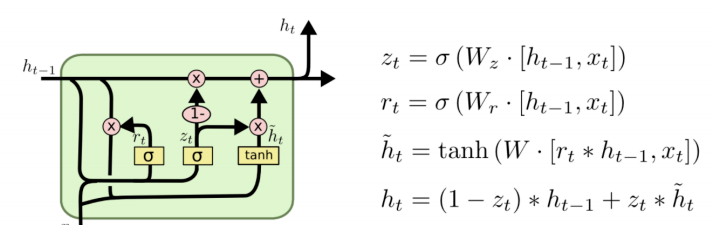
### Peephole connections

‣ Allow gates to look at cell state



### Gated recurrent unit (GRU)

‣ Simplified variant with only 2 gates



# lecture 9:Lexical Semantics

## lexical database

联系word和semantic：

add this information explicitly through a lexical database

之前的方法：

• 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

• Their relationships with other words

‣ Also circular, but more practical

名词解释：

**word sense** describes one aspect of the meaning of a word

**Word Glosses**: textual definition of a sense, given by a dictionary

a word has multiple senses, it is **polysemous**

## Meaning Through Relations

Another way to define meaning: by looking at how it relates to other words

Synonymy近义词

Antonymy反义词

Hypernymy从属于（分类）

Meronymy作为部分属于

**WordNet**:A database of lexical relations

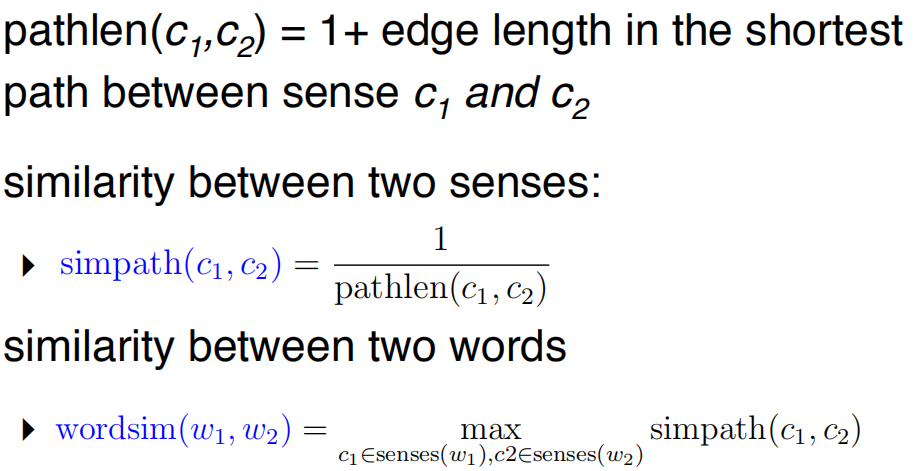
**Synsets**:包括很多lemmas

• Nodes of WordNet are not words or lemmas, but senses

• There are represented by sets of synonyms, or synsets

## Word Similarity

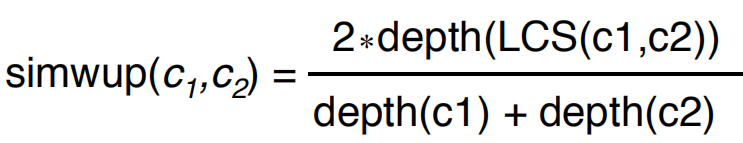
word similarity is a spectrum. We can use lexical database (e.g. WordNet) or thesaurus to estimate word similarity



Problem: edges vary widely in actual semantic distance

‣ Much bigger jumps near top of hierarchy

### include depth information

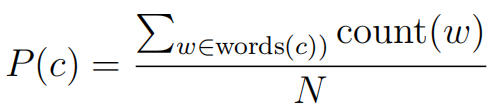


LCS is the the lowest common subsumer

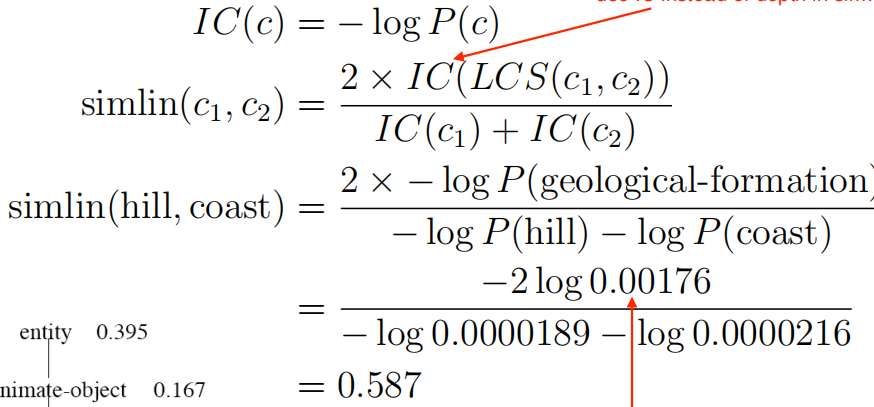
depth：从root往下数第几层

问题：实际问题中，Abstract Nodes很常见，这个应该如何衡量similarity

### Concept Probability

probability that a randomly selected word in a corpus is an instance of concept c

用这个概率可以衡量两个词的相似度



但是实际问题中，一般很难用similarity作为featuress,一般是用整个句子的信息作为features.

Solution: map words in text to senses in WordNet explicitly.

## Word Sense Disambiguation

• Task: selects the correct sense for words in a sentence

• Baseline:简单方法

‣ 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

### 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!

**Less Supervised Approaches:**

Choose sense whose dictionary gloss from WordNet most overlaps with the context

### Other Databases - FrameNet

• Based on frame semantics

• A lexical database of frames, typically prototypical situations

# lecture 10：Distributional Semantics

Lexical Databases（lecture9) - Problems

• Manually constructed

‣ Expensive

‣ Human annotation can be biased and noisy

• Language is dynamic

‣ New words: slang, terminology, etc.

‣ New senses

## Guessing Meaning from Context

• Learn unknown word from its usage

• Look at other words in same (or similar) contexts

需要用到 word vectors

word embeddings!

other ways

‣ Count-based methods

‣ More efficient neural

methods designed just forlearning word vectors

## Count-Based Methods

### The Vector Space Model

Fundamental idea: represent meaning as a vector

文档和词汇的对应统计

Manipulating the VSM：

• Weighting the values (beyond frequency)

• Creating low-dimensional dense vectors

• Comparing vectors

### Tf-idf

df是一个词在所有文章中出现的频率

idf等于一个词在本文出现的频率除以df

## Dimensionality Reduction

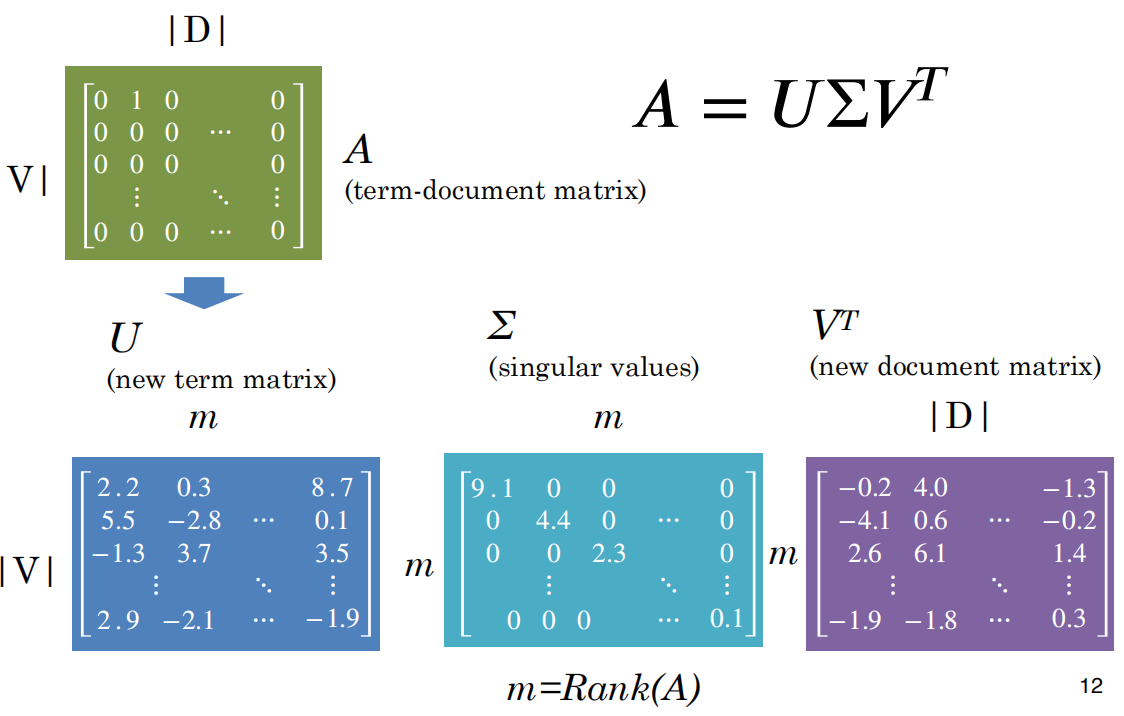
• Term-document matrices are very sparse

• Dimensionality reduction: create shorter, denser vectors

• More practical (less features)

• Remove noise (less overfitting)

### Singular Value Decomposition



### Truncating – Latent Semantic Analysis

• Truncating U, Σ, and VT to k dimensions produces best possible k rank approximation of original matrix

• So truncated, Uk (or VkT ) is a new low dimensional representation of the word

• Typical values for k are 100-5000

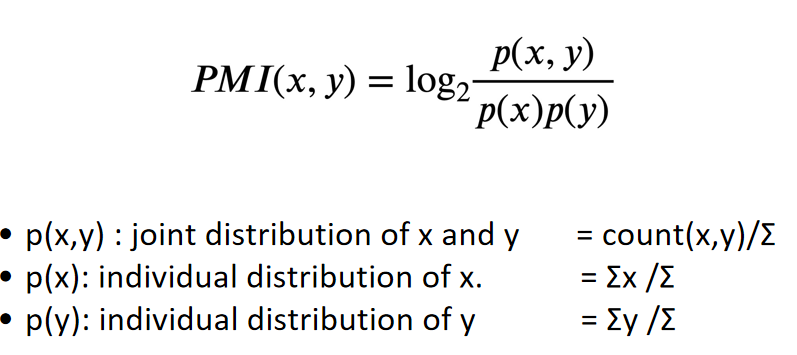
将矩阵U 只留下k列

## Words as Context

Lists how often words appear with other words

但是这样的话，常出现的词频率肯定较高

### Pointwise Mutual Information



优点：

• PMI does a better job of capturing interesting semantics

缺点：

• But it is obviously biased towards rare words 需要Smooth probabilities

• And doesn’t handle zeros well 需要Zero all negative values (PPMI)

也可以用SVD

SVDcan be applied to create dense vectors

### Similarity（based vector）

• 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)

## Neural Methods

### Word Embeddings

也可以用来

‣ Classification

‣ Language modelling

Desiderata:

‣ Unsupervised

‣ Efficient

‣ Useful representation

### Word2Vec

Key idea

‣ Word embeddings should be similar to embeddings of neighbouring words

‣ And dissimilar to other words that don’t occur nearby

Framed as learning a classifier

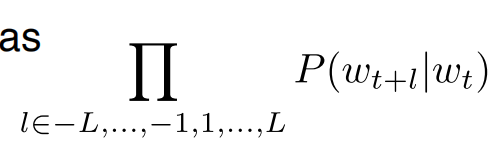
‣ Skip-gram: predict words in local context surrounding given word

‣ CBOW: predict word in centre, given words in the local surrounding context

#### 

#### Skip-gram Model

• Generates each word in context given centre word



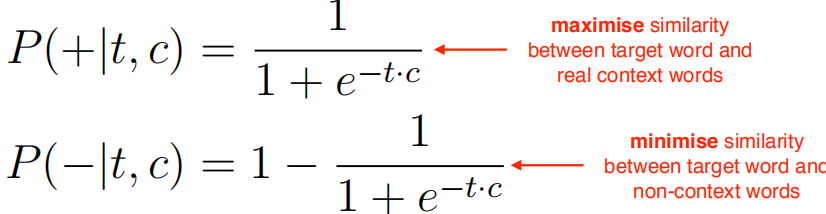
**Embedding parameterisation**

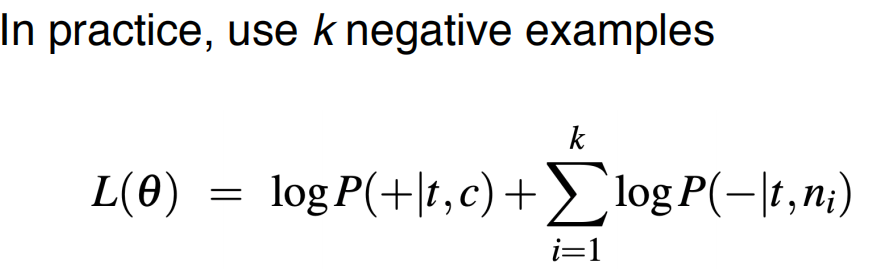
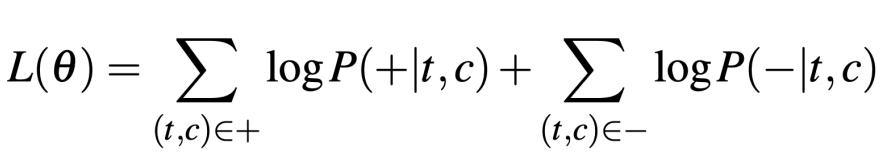
Two parameter matrices, with d-dimensional embedding for all words

**Training the skip-gram model**

Reduce problem to binary classification, distinguish real context words from non-context words aka “negative samples

在训练过程中，将问题简化，为是否是文中的单词





**each step moves embeddings closer for context words**

**and moves embeddings apart for noise samples**

### 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!)

Pointers to Software（备注：实际操作中可能用到的方法）

• 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/

# lecture 11:Contextual Representation

Contextual representation = representation of words based on context

What is contextual representation?

Representations based on a particular usage. It captures the different senses or nuances of the word depending on the context.

Different to word embeddings (e.g.Word2Vec).

Contextual representations that are pre-trained on large data can be seen as a model that has obtained fairly comprehensive knowledge about the language.

优点

Pre-trained contextual representations work really well for downstream applications!

RNN可以capture context to the left 所以需要**bidirectional RNN**

## transformer

What is transformer?

**Encoder-Decoder structure: The Encoder is on the left and the Decoder is on the right.**

**Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times.**

**Positional Encoding: The model need to somehow give every word/part in the sequence a relative position since a sequence depends on the order of its elements.**

**How does a transformer captures dependencies between words?**

**Transformer uses attention to capture dependencies between words.**

**For each target word in a sentence, transformer attends to every other words in the sentence to create its contextual embeddings.**

**What advantages does transformer have compared to RNN?**

**Transformer allows for significantly more parallelization. While RNN relies on sequential processing. This allows transformer-based models to scale to very large data that is difficult for RNN-based models.**

## ELMo

Embeddings from Language Models

• Trains a bidirectional, multi-layer LSTM language model over 1B word corpus

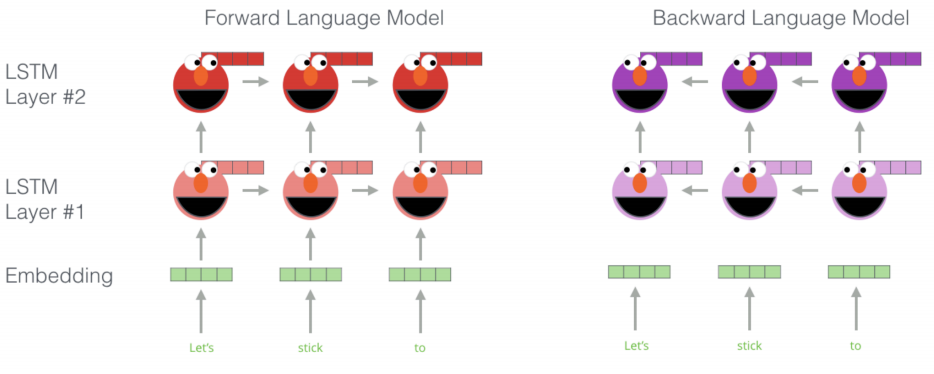
• Combine hidden states from multiple layers of LSTM for downstream tasks

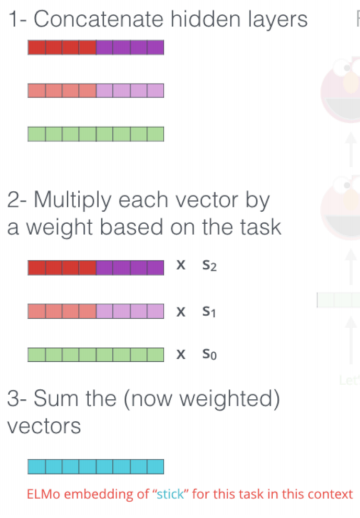
• Improves task performance significantly!

• Number of LSTM layers = 2

• LSTM hidden dimension = 4096

• Character convolutionalnetworks (CNN) to create word embeddings





优点：

use ELMo contextual embedding for words as input of RNN

• Lower layer representation = captures syntax

‣ good for POS tagging, NER

• Higher layer representation = captures semantics

‣ good for QA, textual entailment, sentiment analysis

缺点

• 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

## BERT

定义：Bidirectional Encoder Representations from Transformers

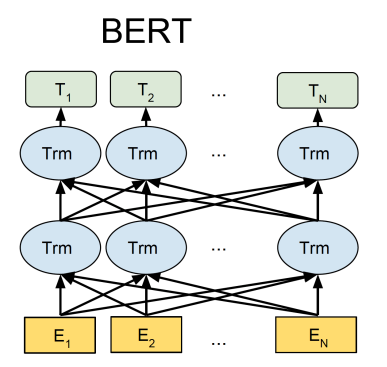
特点

• 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



Objective 1: Masked Language Model

• ‘Mask’ out k% of tokens at random

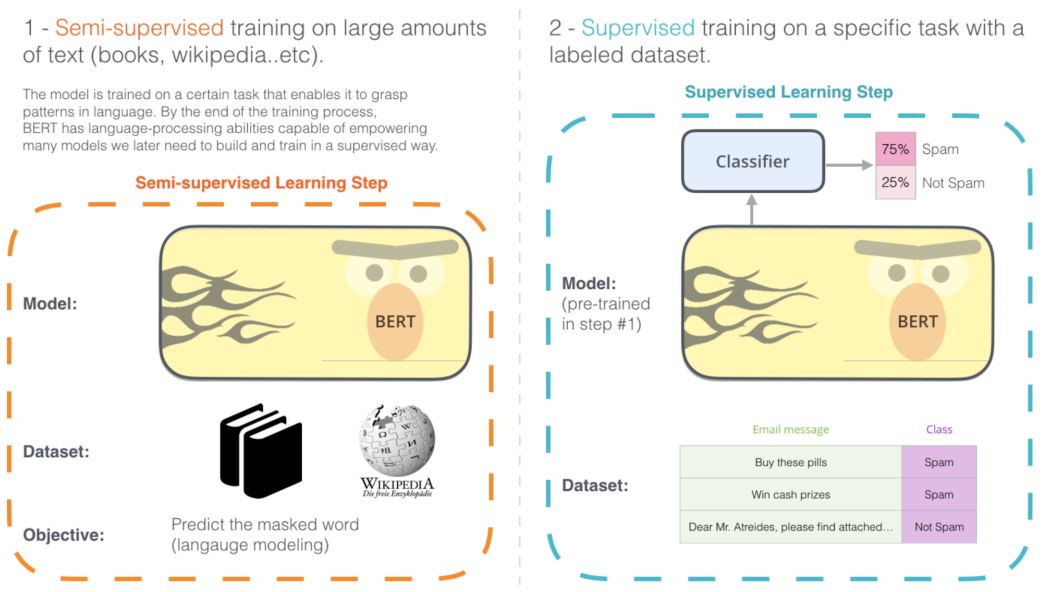
• Objective: predict the masked words

Objective 2: Next Sentence Prediction

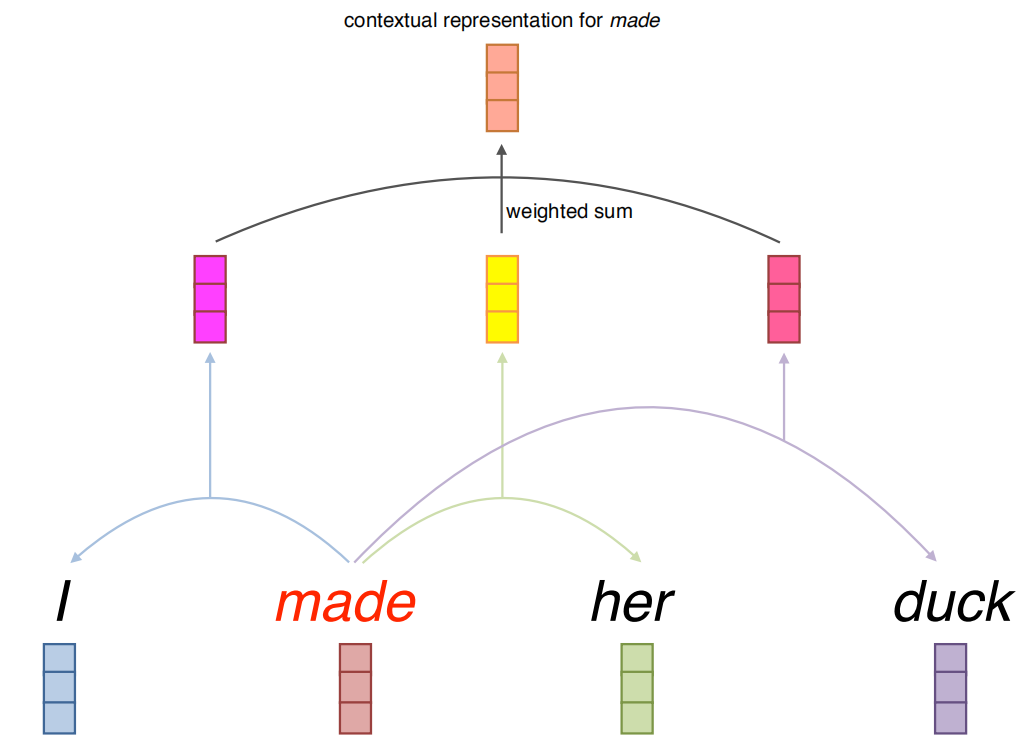
• Learn relationships between sentences

• Predicts whether sentence B follows sentence A

• Useful pre-training objective for downstream applications that analyse sentence pairs (e.g. textual entailment)



transformers 是 attention

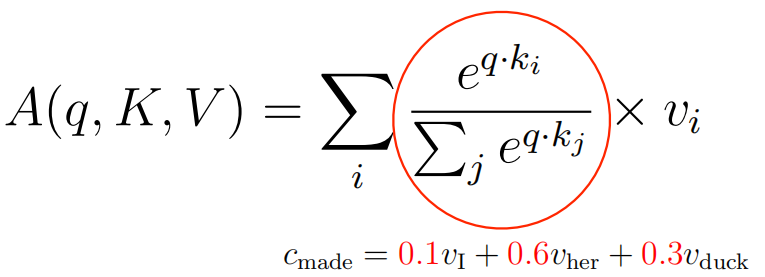


Self-Attention: Implementation

• Input:

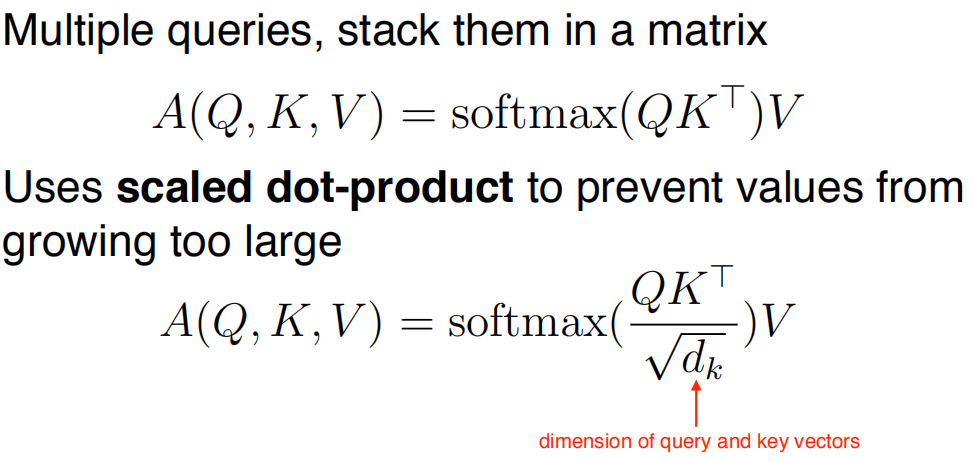
‣ query q (e.g. made )

‣ key k and value v (e.g. her) • Query, key and value are all vectors



• Only one attention for each word pair

• Uses multi-head attention to allow multiple interactions



## 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

• BERT adds a classification layer for downstream tasks

‣ No task-specific model needed

• BERT updates all parameters during fine-tuning

# lecture 12:Discourse

定义：understanding how sentences relate to each other in a document

• Discourse segmentation(分块)

• Discourse parsing(确定结构和关系)

• Anaphora resolution(代词指代判断)

## Discourse segmentation

A segment: a span of cohesive text organised around a particular topic or function

一系列相关的文本

### Unsupervised Approaches

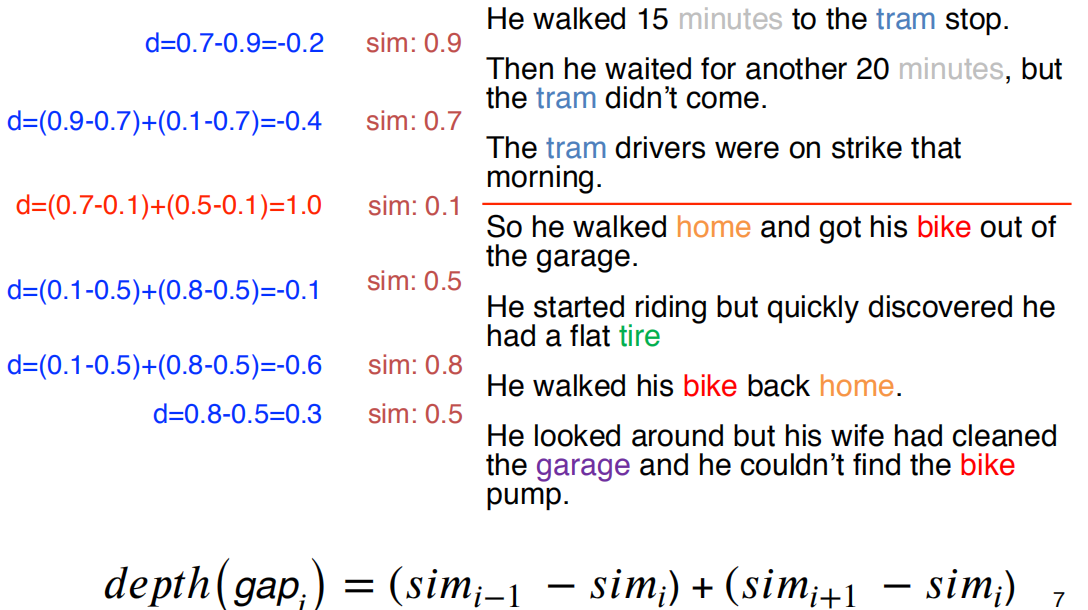
TextTiling algorithm: looking for points of low lexical cohesion between sentences

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



### Supervised Approaches

• Get labelled data from easy sources

‣ Scientific publications

‣ Wikipedia articles

• Apply a binary classifier to identify boundaries

• Or use sequential classifiers

• Potentially include classification of section types (introduction, conclusion, etc.)

• Integrate a wider range of features, including

‣ distributional semantics

‣ discourse markers (therefore, and, etc)

## Discourse Parsing

Identify discourse units, and the relations that hold between them

作用：

• Summarisation

• Sentiment analysis

• Argumentation

• Authorship attribution

• Essay scoring

### Rhetorical Structure Theory (RST)

a framework to do hierarchical analysis of discourse structure in documents

**Basic element: elementary discourse units (EDUs)**

‣ Typically clauses of a sentence

‣ EDUs do not cross sentence boundary

**RST relations between discourse units:**

conjuction, justify, concession, elaboration, etc

Within a discourse relation, one argument is the **nucleus**(the primary argument)

• The supporting argument is the **satellite**

RST Tree

• An RST relation combines two or more DUs into composite DUs

• Process of combining DUs is repeated to create an RST tree

Some discourse markers (cue phrases) explicitly indicate relations can be used to build a **simple rule-based parser**

缺点：

‣ 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

### Parsing Using Machine Learning

Basic idea:

‣ Segment document into EDUs

‣ Combine adjacent DUs into composite DUs iteratively to create the full RST tree

具体方法种类：

• Transition-based parsing (lecture 16):

‣ Bottom-up

‣ Greedy, uses shift-reduce algorithm

• CYK/chart parsing algorithm (lecture 14)

‣ Bottom-up

‣ Global, but some constraints prevent CYK from finding globally optimal tree for discourse parsing

Top-down parsing

‣ Sequence labelling problem

‣ BERT

**Features**

• Bag of words

• Discourse markers

• Starting/ending n-grams

• Location in the text

• Syntax features

• Lexical and distributional similarities

## Anaphora Resolution

作用：Essential for deep semantic analysis

Anaphors: linguistic expressions that refer back to earlier elements (antecedent先行词)in the text

代词的规则：

• Pronouns must agree in number with their antecedents

• Pronouns must agree in gender with their antecedents

• Pronouns whose antecedents are the subject of the same syntactic clause must be reflexive (…self)

先行词判断的规则：

• The antecedents of pronouns should be recent

• The antecedent should be salient, as determined by grammatical position

### 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

Forward-looking centers: ‣ All entities in Un: Cf(Un) = [e1, e2, …]

Backward-looking center:‣ Highest ranked entity in previous utterance’s (Cf(Un-1))

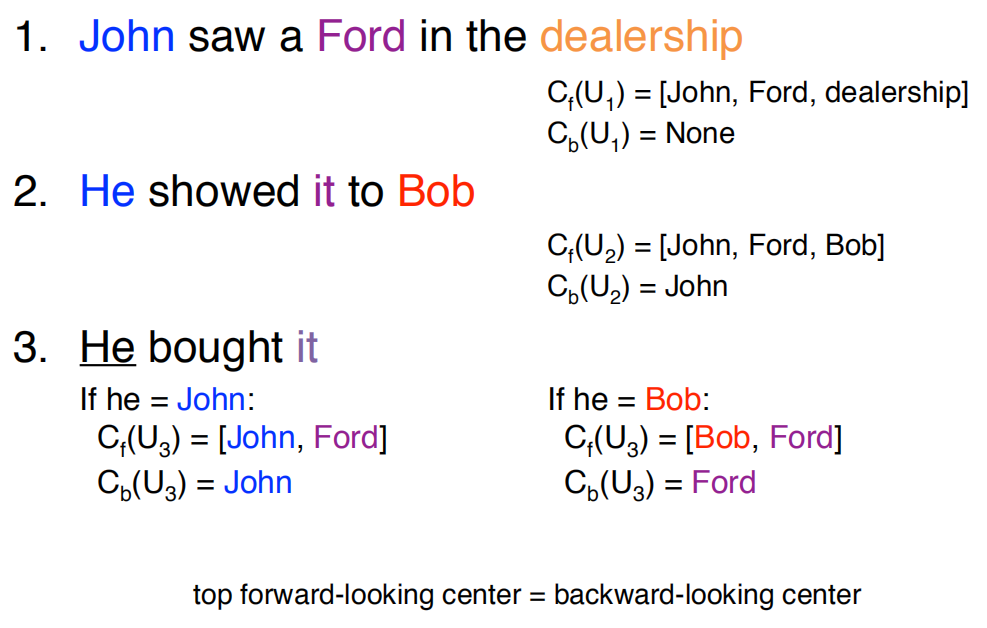
forward-looking centers that is also in current utterance (Un)

eg: ‣ Cb(16.b) = [John]

### Centering Algorithm

• When resolving entity for anaphora resolution,

choose the entity such that the top fowardlooking center matches with the backwardlooking center



### 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

### Anaphora Resolution Tools

• Stanford CoreNLP includes pronoun coreference models

‣ rule-based system does very well

‣ considerably faster than learned models

# lecture 13:Formal Language Theory & Finite State Automata

A language = set of strings

A string = sequence of elements from a finite alphabet

• 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

应用：

• 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!)

Overview

• Regular languages

• Finite state acceptors & transducers

• Modelling word morphology

## Regular Languages

• Regular language: the simplest class of languages

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

规则运算：

‣ concatenation and union — follows from definition

‣ intersection: strings that are valid in both L1 and L2

‣ negation: strings that are not in L

## Finite state acceptors

Finite state acceptors (FSA) describes the computation involved for membership checking

FSA consists:

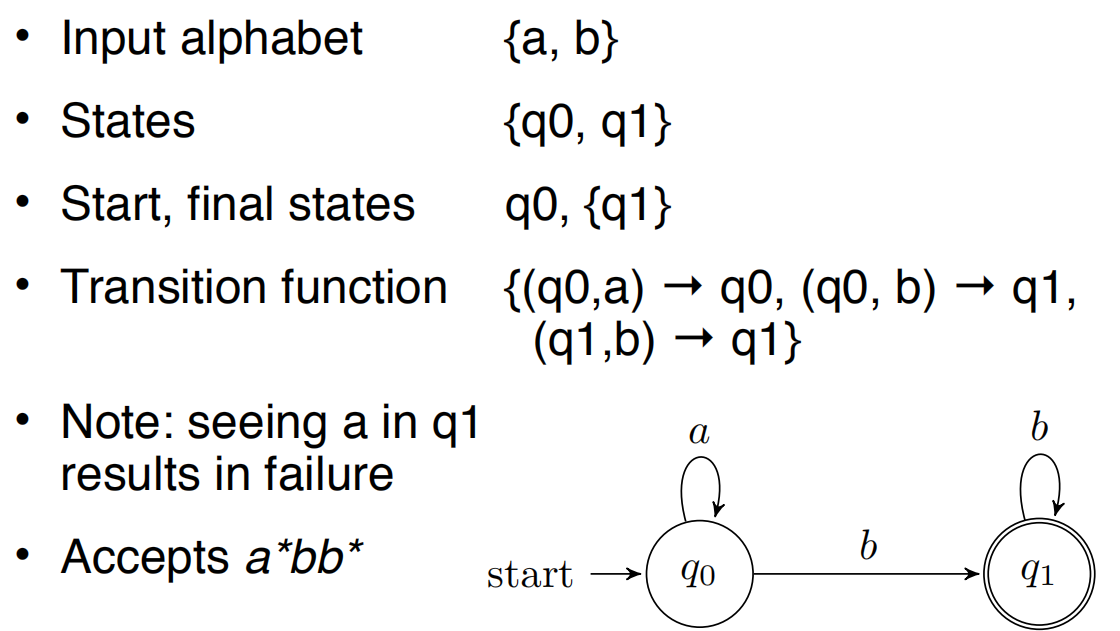
‣ alphabet of input symbols, Σ ‣ set of states, Q

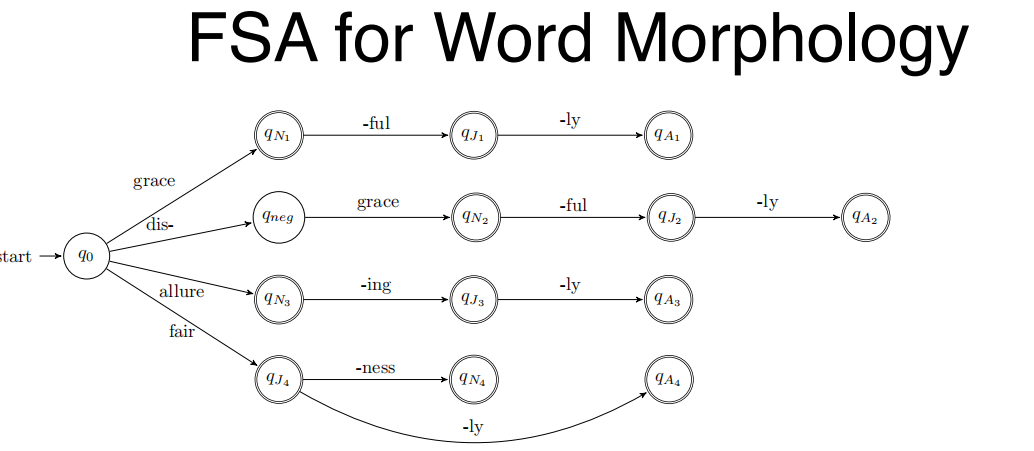
‣ start state, q0 ∈ Q

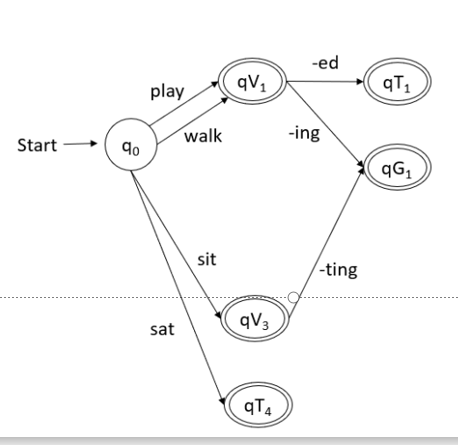
‣ final states, F ⊆ Q

‣ transition function

symbol and state → next state







## Weighted FSA

weighted FSA adds/changes the following:

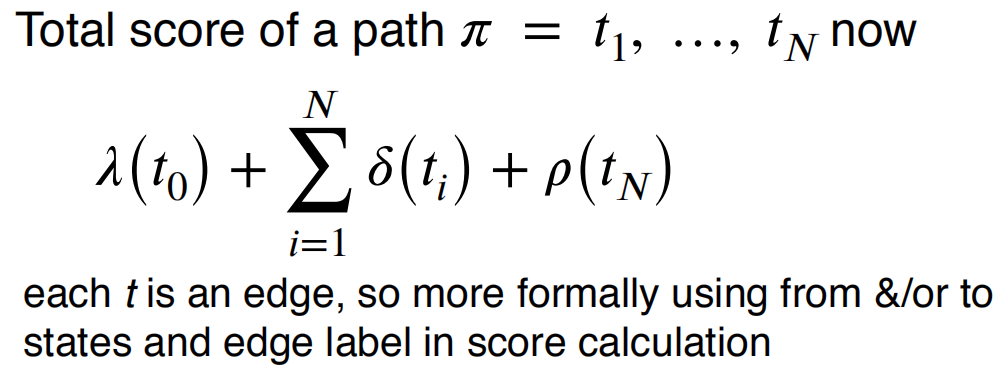
‣ set of states Q

‣ alphabet of input symbols Σ

‣ start state weight function, λ: Q → ℝ

‣ final state weight function, ρ: Q → ℝ

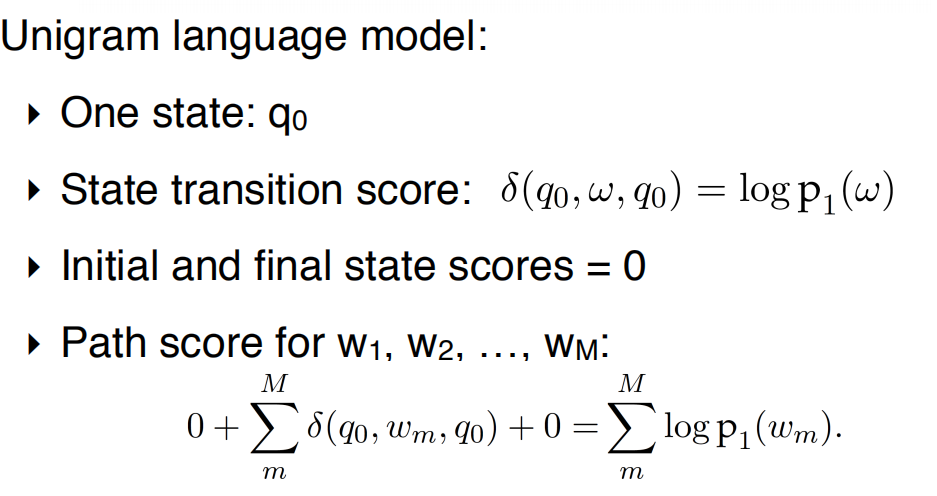
‣ transition function, δ: (Q, Σ, Q) → ℝ

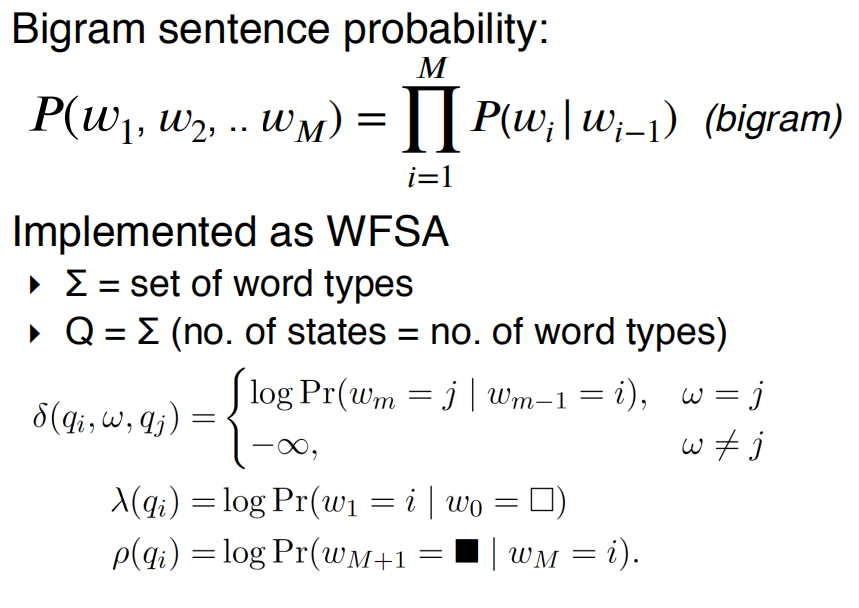


Accepts strings if there is path from q0 to a final state with transitions matching each symbol

‣ Djisktra’s shortest-path algorithm, O(V log V + E)

### N-gram LMs as WFSA





## Finite State Transducer（FST)

作用

Often don’t want to just accept or score strings, want to translate them into another language, correct grammar, parse their structure, etc

eg:

FSA: allure + ing = allureing

FST: allure + ing = alluring

FST add string output capability to FSAs

‣ includes an output alphabet

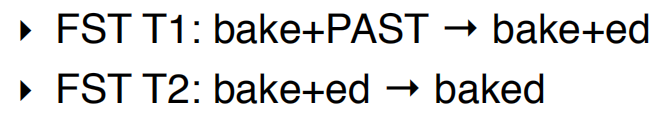
‣ and transitions now take input symbol and emit output symbol (Q, Σ, Σ, Q)

### FST Composition

• Compose two FSTs by taking output of one FST, T1, and giving this as input to FST T2

‣ denoted T1 ○ T2; and results in another FST

‣ can also compose FST with FSA, resulting in a FST



# lecture 14:Context-Free Grammar

目的：Center embedding

重要组成

**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)**

‣ 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

free 这个词的原因：仅仅和LHS有关

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

和Regular languages相比：more general

‣ Allows recursive nesting

If English can be represented with CFG:

‣ we can build a “parser” to automatically judge whether a sentence is grammatical!

但是不是所有的语言都是，有的语言有严重的 cross-serial：相关的词不是相邻的

但是仍然具有的好处：

‣ CFG cover most syntactic patterns

‣ CFG parsing is computational efficient

## Syntactic Constituents

• Sentences are broken into constituents

‣ word sequence that function as a coherent unit for linguistic analysis

• Constituents have certain key properties:

‣ movement：Constituents can be moved around sentences

‣ substitution：Constituents can be substituted by other phrases of the same type

‣ coordination：Constituents can be conjoined with coordinators like and and or

## CFG

Once we identify constituents, we use phrases to describe them

Phrases are determined by their head word、

We can use CFG to formalise these intuitions

## Parsing CFG

Parsing is the reverse process

Parse Ambiguity

• Often more than one tree can describe a string

### CYK Algorithm

• Bottom-up approach to parsing in CFG

Core idea: form small constituents first, and merge them into larger constituents

Requirement: CFGs must be in Chomsky Normal Forms

### 

### Chomsky Normal Form

Convert rules A → B C D into:

‣ A → B Y

‣ Y → C D

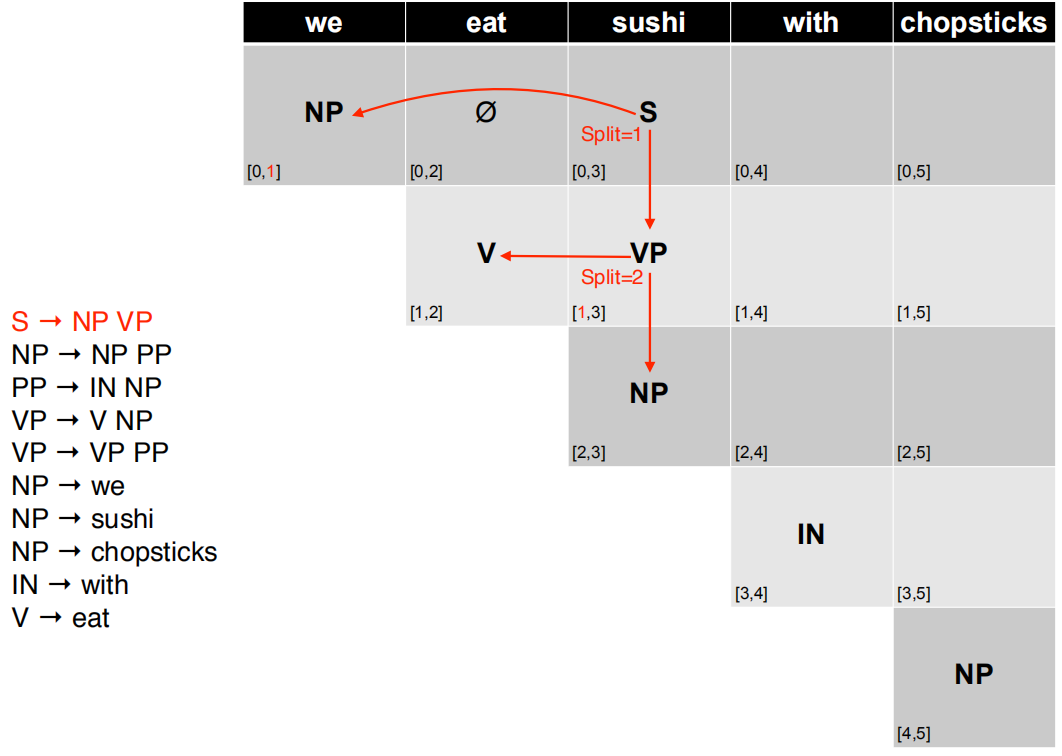
步骤：

• Fill in a parse table

• Use table to derive parse

• S in top right corner of table = success!

• Convert result back to original grammar



### CNF (cont)

• CNF disallows unary rules, A → B.

• Imagine NP → S; and S → NP … leads to infinitely many trees with same yield.

• Replace RHS non-terminal with its productions

• E.g convert A → B, B → 1, B → 2 into A → 1, A →2

## 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

# lecture 15:Probabilistic Context-Free Grammar

Ambiguity In Parsing

Context-free grammars assign hierarchical structure to language

结果有很多种

• Probabilistic context-free grammars (PCFGs)

• Parsing using dynamic programming

• Limitations of ‘context-free’ assumption and some solutions:

‣ parent annotation

‣ head lexicalisation

优点

• 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

## Probabilistic context-free grammars (PCFGs)

跟Context-free grammars相比In addition, store a probability with each production

‣ must be positive values, between 0 and 1

‣ must sum to one for given LHS

### Stochastic Generation with PCFGs

1. Start with S, the sentence symbol

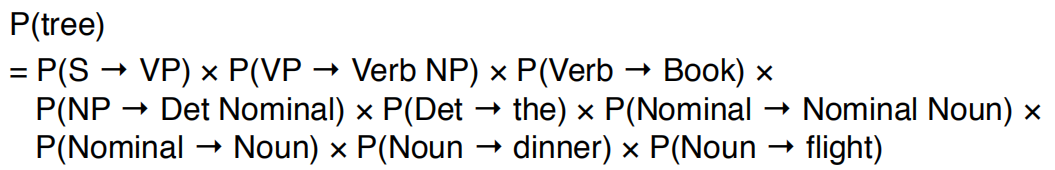
2. Choose a rule with S as the LHS

‣ Randomly select a RHS according to Pr(RHS | LHS)e.g., S → 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



## Parsing PCFGs

we select

a tree from the set of all possible tree, before we looked at

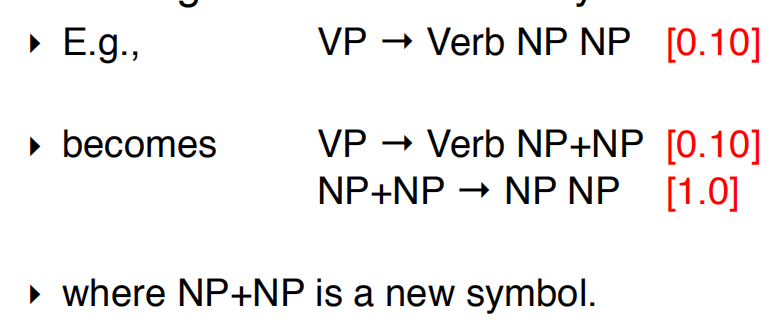
‣ CYK

‣ for unweighted grammars (CFGs)

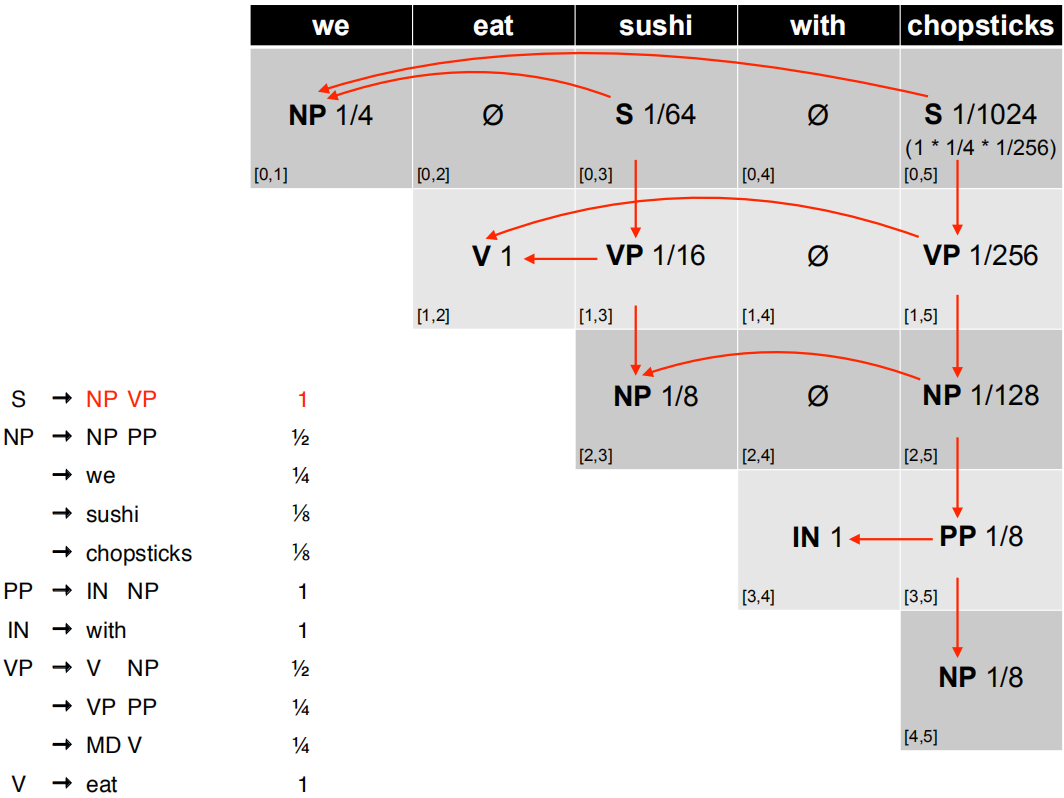
‣ finds all possible trees

但是这样复杂度和花费的时间很高

### CNF



### CYK



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 nonterminals

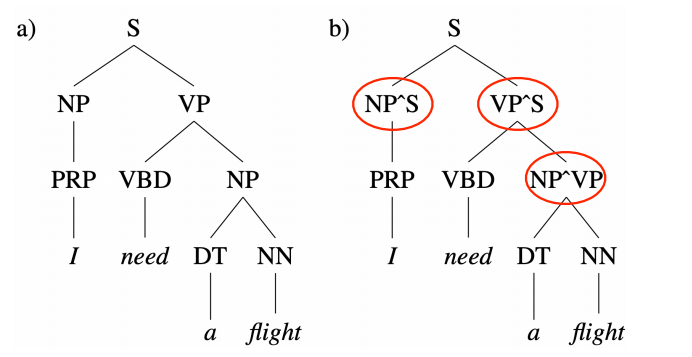
## Issues with PCFG

Poor Independence Assumptions

• Rewrite decisions made independently, whereas interdependence is often needed to capture g 比如转化时候会根据主语宾语的情况概率不同

### Solution: Parent Conditioning

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



Lack of Lexical Conditioning

• Lack of sensitivity to words in tree

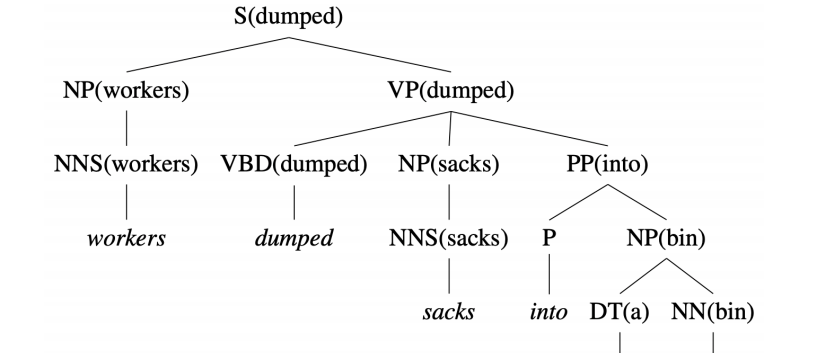
• Prepositional phrase (PP) attachment ambiguity

有的时候可以选择放在不同的地方，但是无法判断哪个更合适

### 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



Incorporate head words into productions, such that the most important links between words is captured

Grammar symbol inventory expands massively!

‣ Many of the productions much too specific, seen very rarely

‣ Learning more involved to avoid sparsity problems

# lecture 16:Dependency Grammars

Dependency grammar offers a simpler approach

‣ describe relations between pairs of words

‣ namely, between heads and dependents

• Head = central word

• Dependent = supporting word

• Grammatical relation = subject, direct object, etc

**Universal Dependency**: a framework to create a set of dependency relations that are computationally useful and cross-lingual

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

优点：

• 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

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

## 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

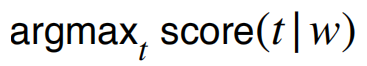
• Most sentences are projective, however exceptions exist

• Common in languages with flexible word order

其实就是没有交叉的弧线

## Parsing

task of finding the best structure for a given input sentence



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

### Transition-Based Parsing

Transition-based parsers can only produce projective trees

#### Intuition

• Examine the words in a single pass from left to right

maintain two data structures

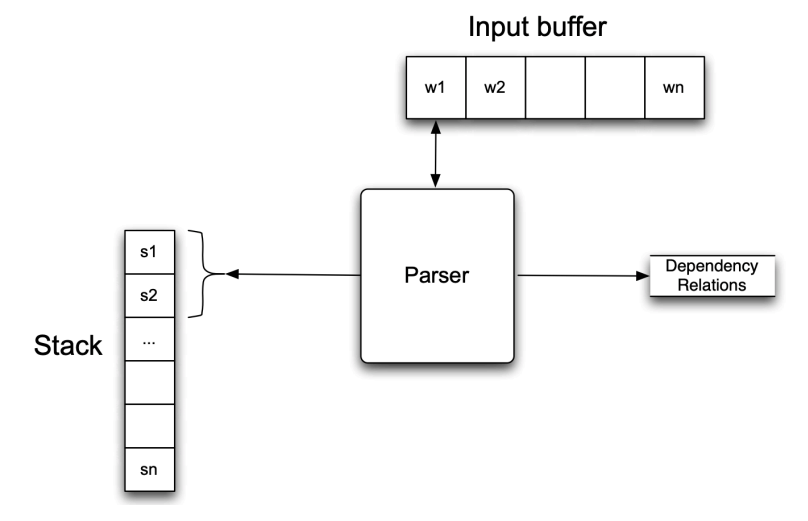
- buffer: input words yet to be processed

- 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



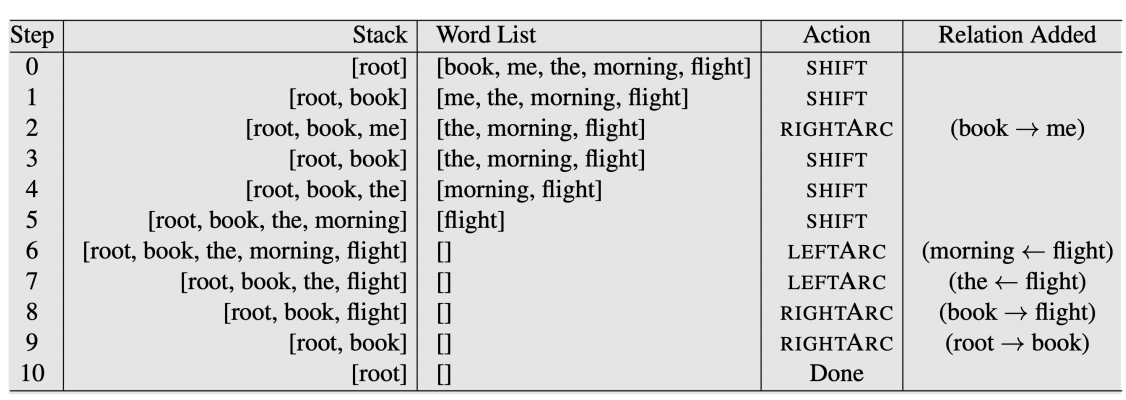
#### 步骤

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]



#### Dependency Labels

• For simplicity, we omit labels on the dependency relations

• In practice, we parameterise the left-arc and rightarc actions with dependency labels:

‣ E.g. left-arc-nsubj or right-arc-dobj

#### 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

Parsing As Classification

• Input:

‣ Stack (top-2 elements: s1 and s2) ‣ Buffer (first element: b1) • Output

‣ 3 classes: shift, left-arc, or, right-arc

• Features

‣ word (w), part-of-speech (t)

• Traditionally SVM works best

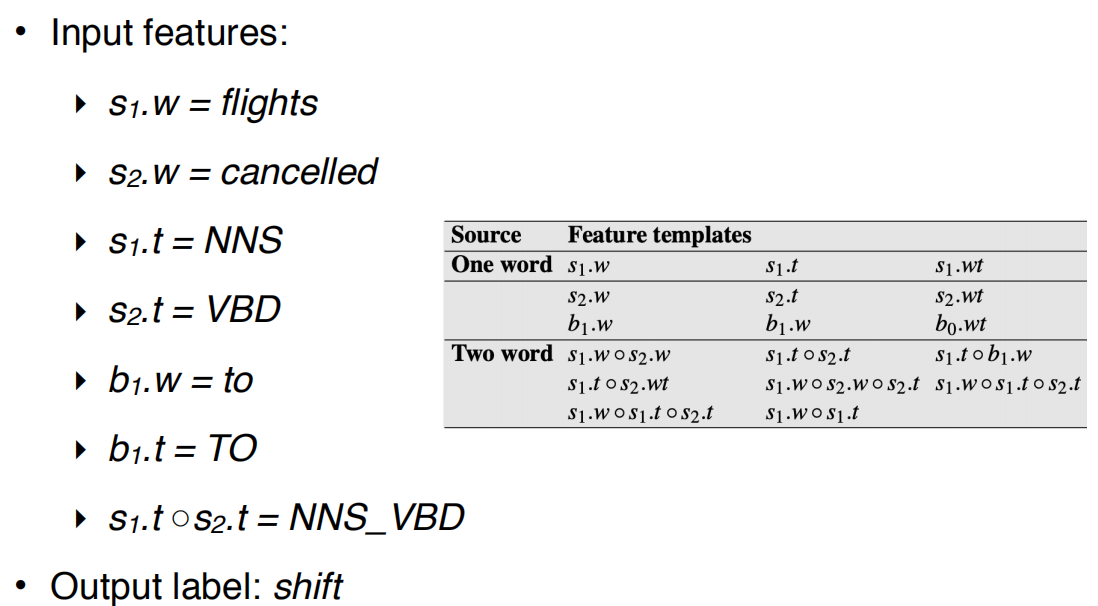
• Nowadays, deep learning models are the state-ofthe-art

• 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



### Graph-Based Parsing

• Given an input sentence, construct a fullyconnected, 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

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

# lecture 17:Machine Translation

Machine translation (MT) is the task of translating text from one source language to another target language

好处：

• Removes language barrier

• Makes information in any languages accessible to anyone

难点：

• But translation is a classic “AI-hard” challenge

‣ Difficult to preserve the meaning and the fluency of the text after translation

• Not just simple word for word translation

• Structural changes, e.g., syntax and semantics

• Multiple word translations, idioms

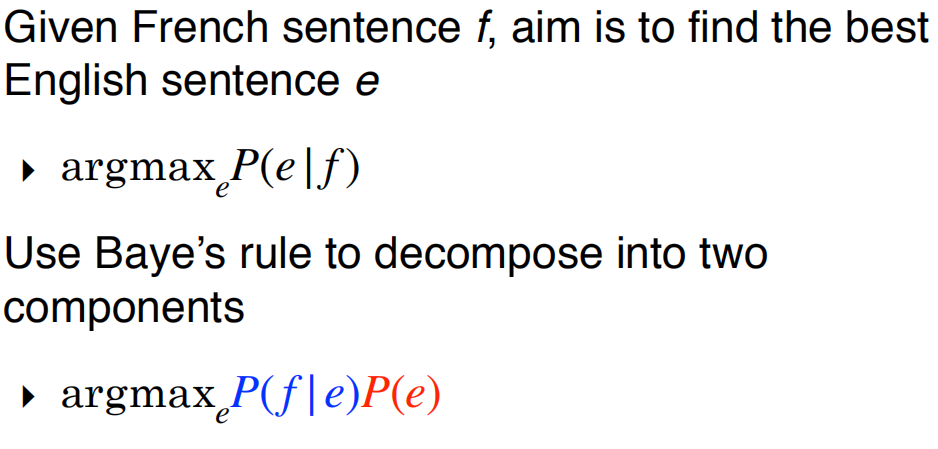
• Inflections for gender, case etc

• Missing information (e.g., determiners)

## Statistical Machine Translation

## Rule-based system

‣ Use bilingual dictionary to map Russian words to English words



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

• Language model:

‣ Based on text frequency in large monolingual corpora

• Translation model:

‣ Based on word co-occurrences in parallel corpora

特点（缺点）

• 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

Parallel Corpora

• One text in multiple languages

• Produced by human translation

### Alignment

• Idea: introduce word alignment as a latent variable into the model

Complexity of Alignment

• Some words are dropped and have no alignment

• One-to-many alignment

• Many-to-one alignment

• Many-to-many alignment

• Word alignments are rarely provided in parallel corpora

‣ As such alignment a introduced as a latent variable

• Use algorithms such as expectation maximisation (EM) to learn

## Neural Machine Translation

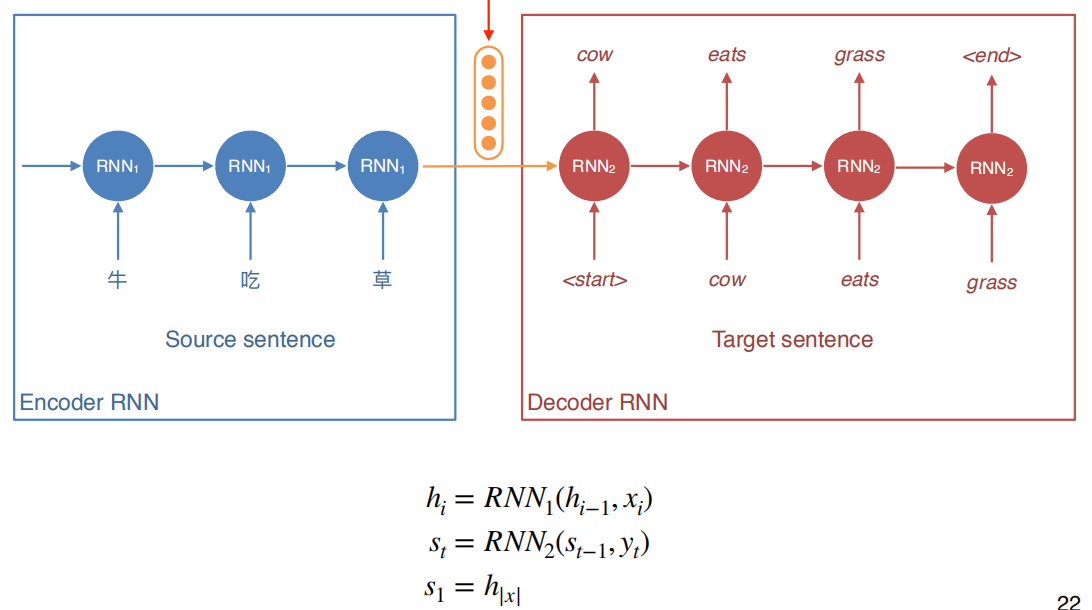
• Neural machine translation is a new approach to do machine translation

• 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



This vector encodes the whole source sentence;it is used as the initial state for decoder RNN

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

优点：

• Single end-to-end model

‣ Statistical MT systems have multiple sub-components

• Less feature engineering

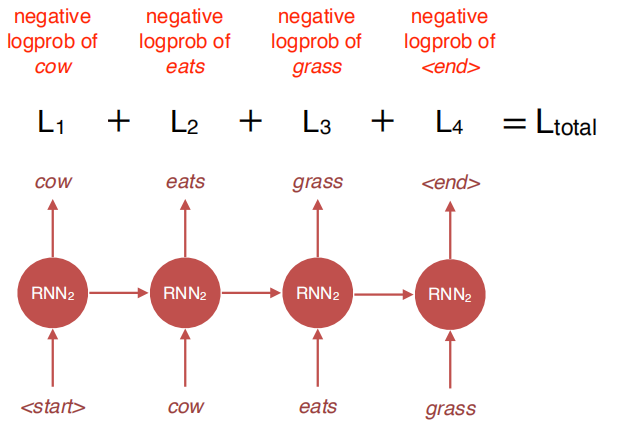
• Generated translation is more fluent

• But can produce new details that are not in the source

sentence (hallucination)

### Neural MT Training Loss

利用对照词汇表training

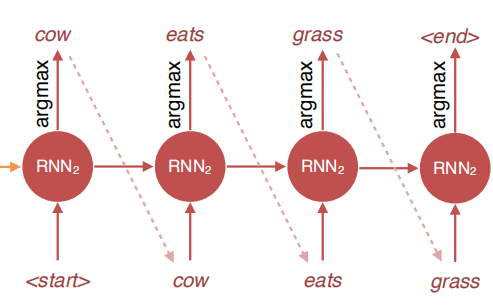


• During training, we have the target sentence. We can therefore feed the right word from target sentence, one step at a time

### Decoding at Test Time

But at test time, we don’t have the target sentence (that’s what we’re trying to predict!)

#### • argmax(greedy decoding): take the word with the highest probability at every step



To find optimal , we need to consider every word at every step to compute the probability of all possible sequences

#### Beam Search Decoding

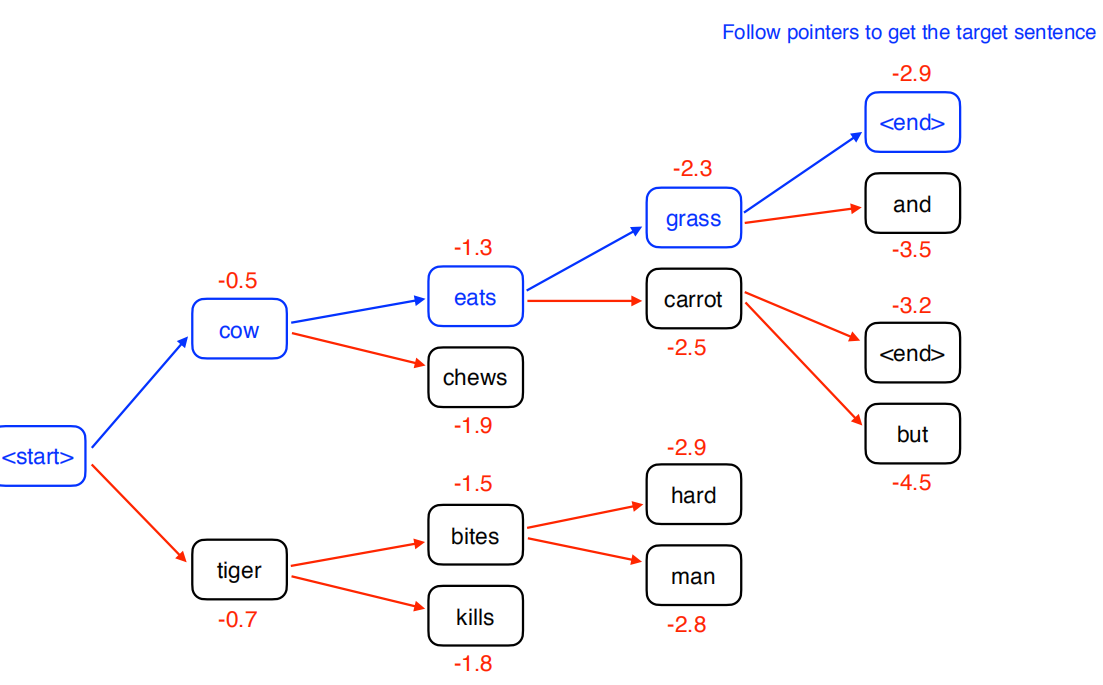
considering all possible words at every step, consider k best words

• That is, we keep track of the top-k words that produce the best partial translations (hypotheses)

• k = beam width (typically 5 to 10)

• k = 1 = greedy decoding

• k = V = exhaustive search decoding



When to Stop?

• When decoding, we stop when we generate <end> token

• But multiple hypotheses may terminate their sentence at different time steps

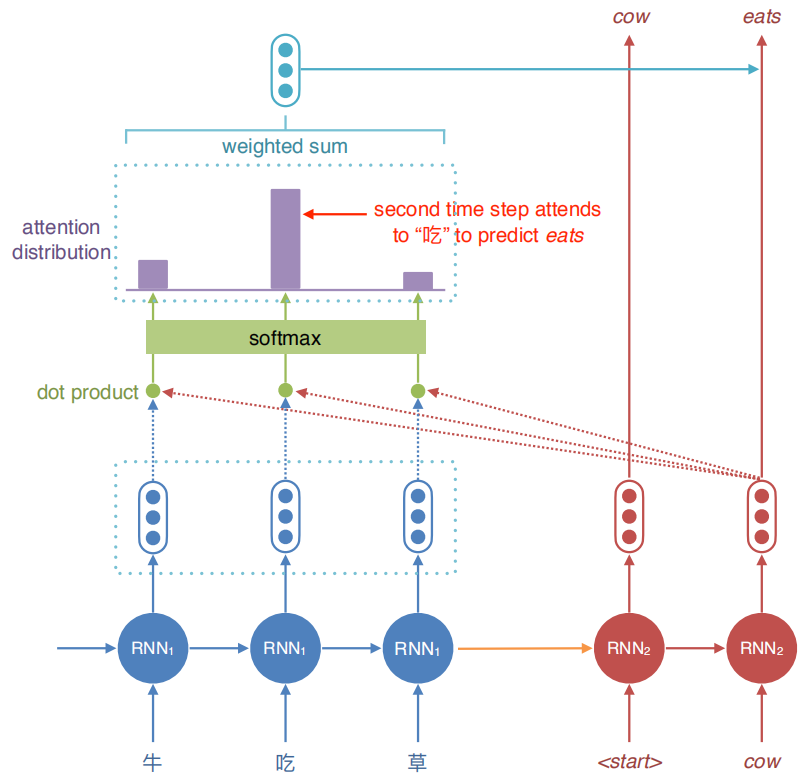
• We store hypotheses that have terminated, and continue explore those that haven’t

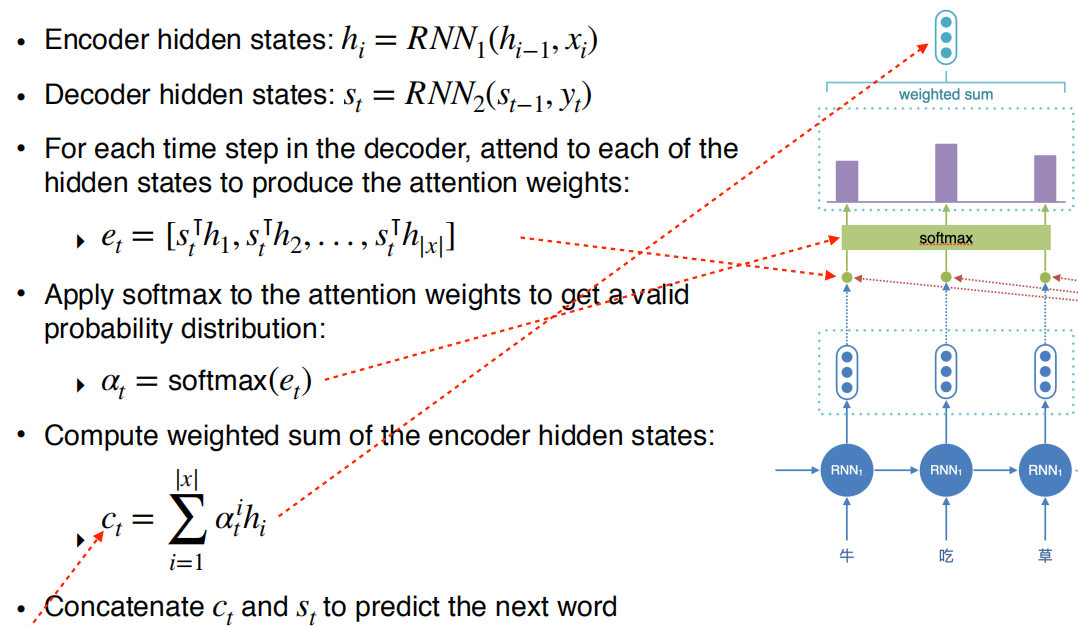
• Typically we also set a maximum sentence length that can be generated (e.g. 50 words)

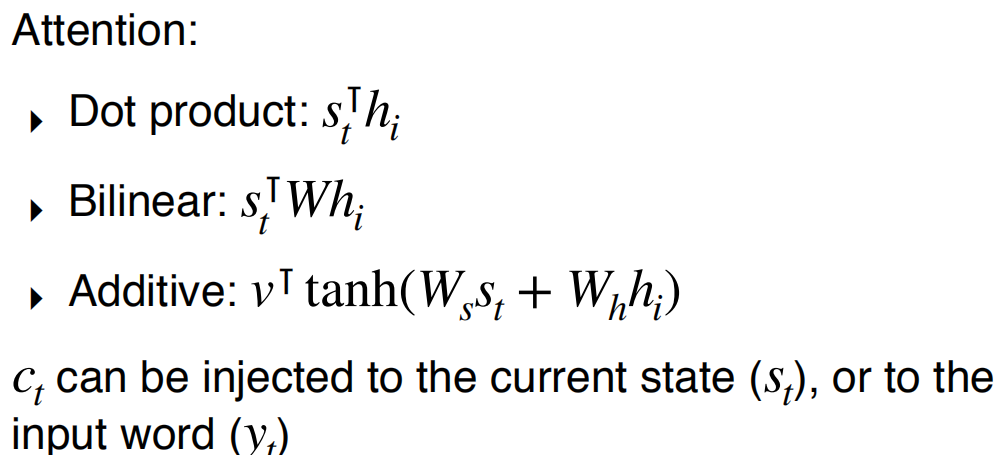
### Attention

• With a long source sentence, the encoded vector is unlikely to capture all the information in the sentence

• This creates an information bottleneck







优点：

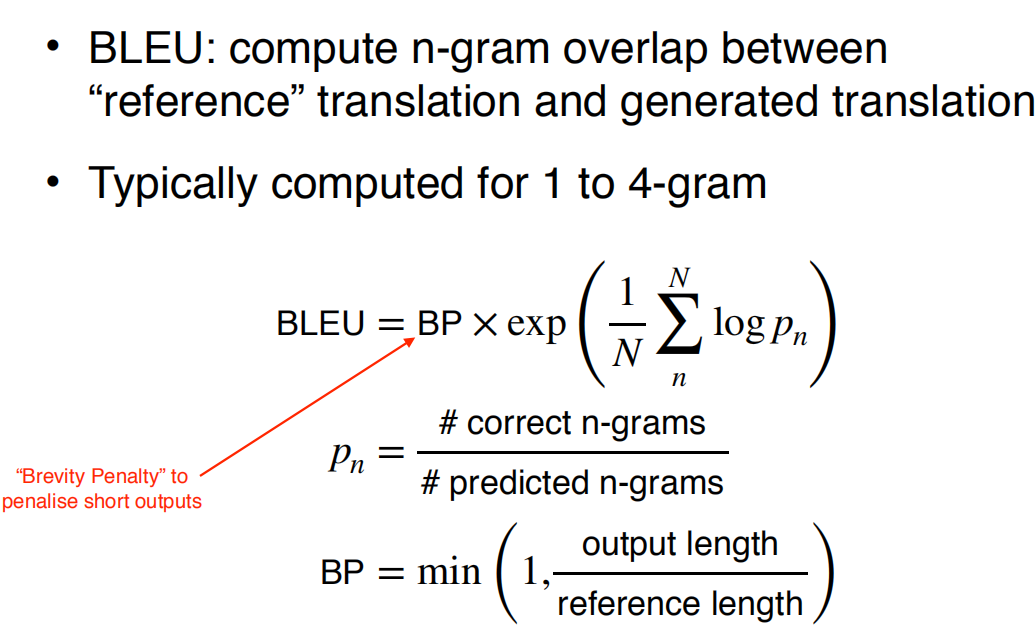
• Solves the information bottleneck issue by allowing decoder to have access to the source sentence words directly

• 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

## MT Evaluation



# lecture 18 Information Extraction

Main goal: turn text into structured data such as databases, etc.

• Help decision makers in applications

• Stock analysis

‣ Gather information from news and social media → summarise into a structured format → decide whether to buy/sell at current stock price

• 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

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”)

• Named Entity Recognition (NER): sequence models such as RNNs, HMMs or CRFs.

• Relation Extraction: mostly classifiers, either binary or multi-class

## Named Entity Recognition

NE tags can be ambiguous

solutions

### IO tagging

• ‘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

### IOB tagging

• B-ORG represents the beginning of an ORG entity.

• If the entity has more than one token, subsequent tags are represented as I-ORG

### 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

Features

• Character and word shape features (ex: “L’Occitane”)

• Prefix/suffix:

‣ L / L’ / L’O / L’Oc / …

‣ e / ne / ane / tane / …

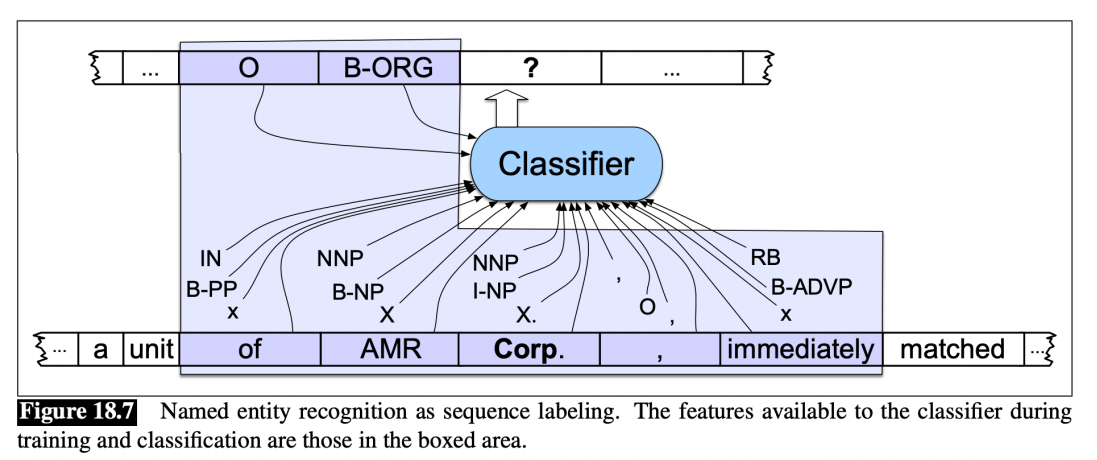
• Word shape:

‣ X’Xxxxxxxx / X’Xx

‣ XXXX-XX-XX (date!)

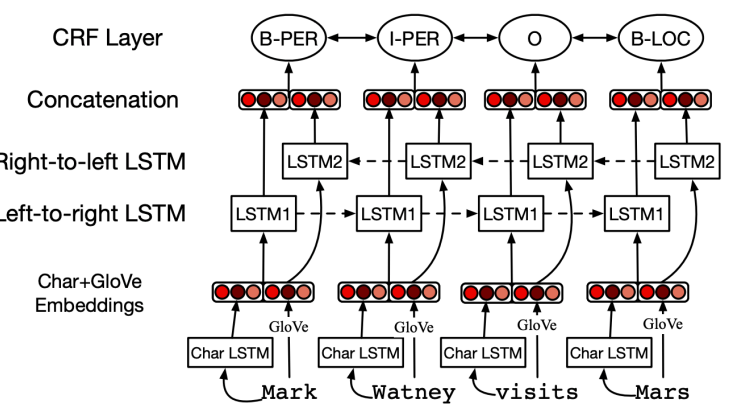
• POS tags / syntactic chunks: many entities are nouns or noun phrases.

• Presence in a gazeteer: lists of entities, such as place names, people’s names and surnames, etc.



Deep Learning for NER

• State of the art approach uses LSTMs with character and word embeddings (Lample et al. 2016)



## Relation Extraction

• Traditionally framed as triple extraction:

‣ unit(American Airlines, AMR Corp.)

‣ spokesman(Tim Wagner, American Airlines)

• 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

### Rule-Based Relation Extraction

NP0 such as NP1 → hyponym(NP1, NP0) • hyponym(Gelidium, red algae)

特点

Lexico-syntactic patterns: high precision, low

recall, manual effort required

### Supervised Relation Extraction

Assume a corpus with annotated relations

• Two steps. First, find if an entity pair is related or not (binary classification)

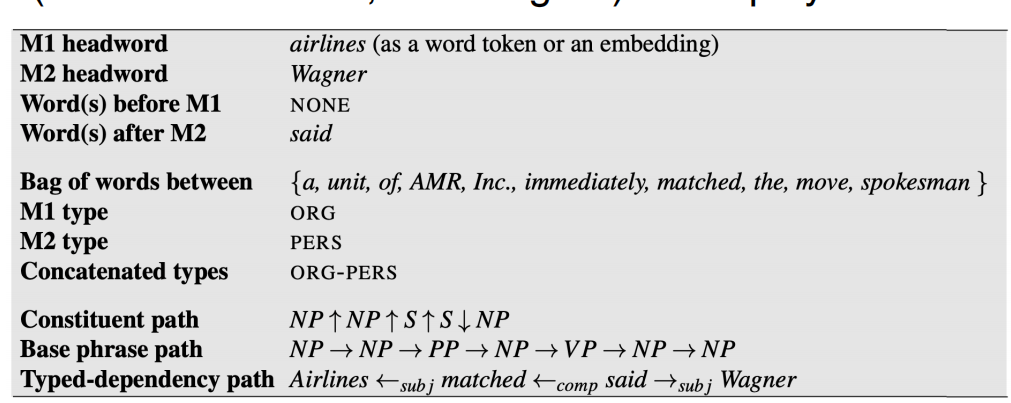
‣ For each sentence, gather all possible entity pairs

‣ 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

features:



### Semi-supervised Relation Extraction

• Annotated corpora is very expensive to create.

• Use seed tuples to bootstrap a classifier

1. Given a set of seed tuples

2. Find sentences containing these seed tuples

3. Extract general patterns from these sentences

4. Use these patterns to find new tuples

5. Repeat from step 2

Issue: Semantic Drift

• Pattern: [NP] has a {NP}\* hub at [LOC]

• Sydney has a ferry hub at Circular Quay

‣ hub(Sydney, Circular Quay)

• More erroneous patterns extracted from this tuple…

• Should only accept patterns with high confidences

### Distant Supervision

We mine new tuples directly? • Distant supervision obtain new tuples from a range of sources:

‣ DBpedia

‣ Freebase

优点

• Generate massive training sets, enabling the use of richer features, and no risk of semantic drift

缺点：

• Still rely on a fixed set of relations

### 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

Main problem: mapping the substring relations into canonical forms

## 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.

# lecture 19:Question Answering

question answering (“QA”) is the task of automatically determining the answer for a natural language question

• 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

## Information retrieval-based 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)

‣ 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)

### Answer Types

• Knowing the type of answer can help in:

‣ finding the right passage containing the answer

‣ finding the answer string

• Treat as classification

‣ given question, predict answer type

‣ key feature is question headword

‣ Generally not a difficult task

### 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)

### Answer Extraction

• Find a concise answer to the question, as a span in the text

#### 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

#### Neural Answer Extraction

• Use a neural network to extract answer

• AKA reading comprehension task

data to train comprehension:

MCTest

SQuAD

Reading Comprehension

• Answer span starts/ends at which token in passage?

• Compute:

‣ prob. of token i is the starting token

‣ :prob. of token i is the ending token

pstart(i) pend(i)

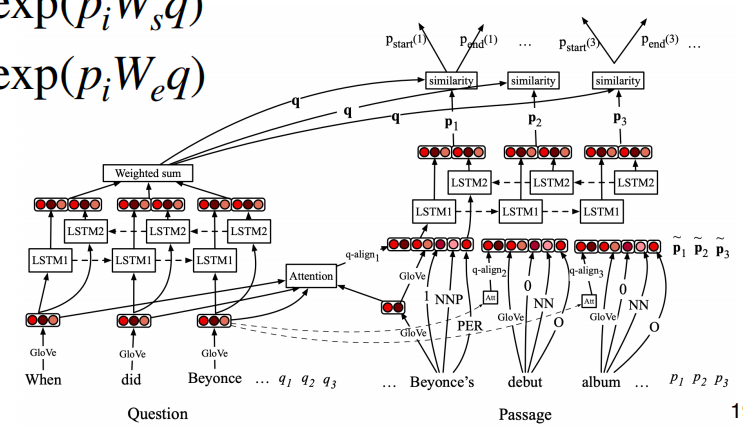
LSTM-Based

More than just word embeddings as input

‣ A feature to denote whether the word matches a question word

‣ POS feature

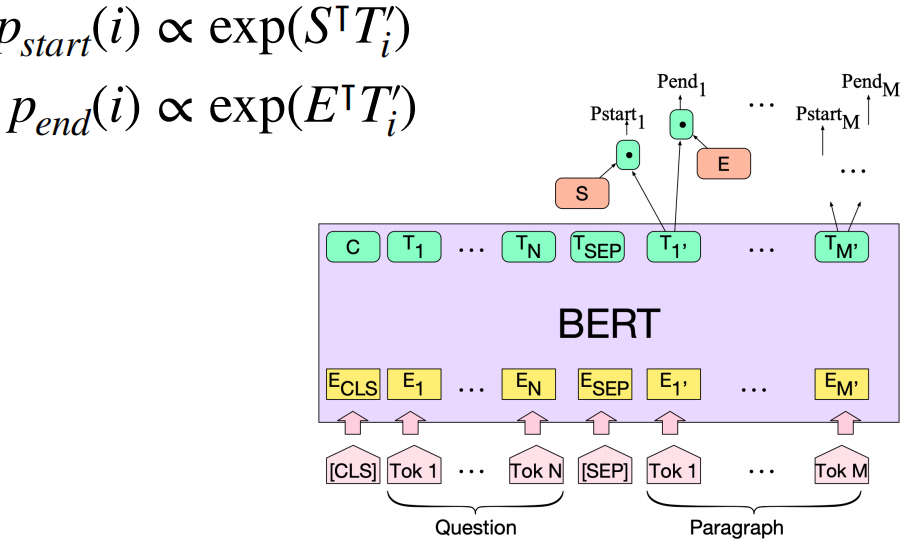
‣ Weighted question embedding: produced by attending to each question words



Bert-Based

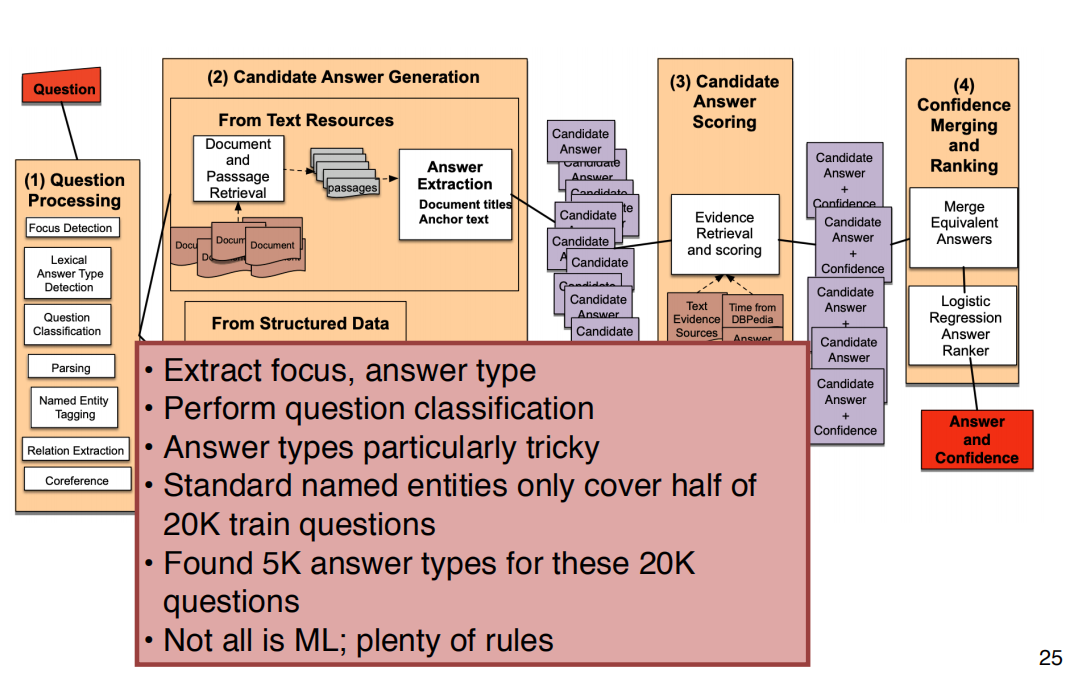
• Fine-tune BERT to predict answer span

• pstart(i) ∝ exp(S⊺T′i) pend(i) ∝ exp(E⊺T′i)



## Knowledge-Based QA

### IBM’s WATSON



## 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 = ⅓

• SQuAD:

‣ Exact match of string against gold answer

‣ F1 score over bag of selected tokens

• MCQ reading comprehension: Accuracy

# lecture 20:Topic Modelling

Topic models learn common, overlapping themes in a document collection

• Unsupervised model

‣ No labels; input is just the documents!

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

## A Brief History of Topic Models

### Latent Semantic Analysis (L10): SVD+Truncate

Issues

• Positive and negative values in the U and VT

• Difficult to interpret

### Probabilistic LSA

• Based on a probabilistic model

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

## Latent Dirichlet Allocation

• Introduces a prior to the document-topic and topicword distribution

• Fully generative: trained LDA model can infer topics on unseen documents!

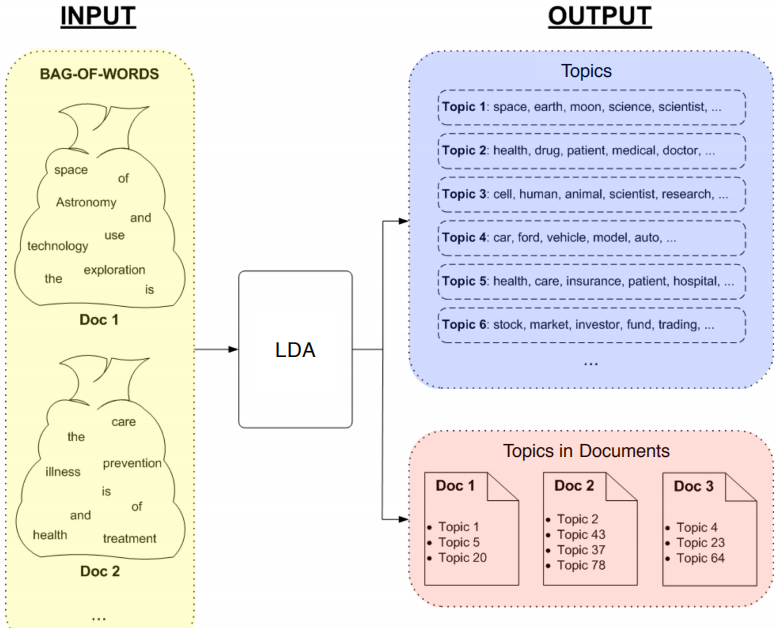
• LDA is a Bayesian version of PLSA

• 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



Input

• A collection of documents

• Bag-of-words

• Good preprocessing practice:

‣ Remove stopwords

‣ Remove low and high frequency word types

‣ Lemmatisation

Output

• Topics: multinomial distribution over words in each topic

• Topics in documents: multinomial distribution over topics in each document

• Two main family of algorithms:

‣ Variational methods

‣ Sampling-based methods

### Sampling Method (Gibbs)

1. Randomly assign topics to all tokens in documents

2. Collect topic-word and document-topic co-occurrence statistics based on the assignments

3. Go through every word token in corpus and sample a new topic:

4. Repeat until convergence

Need to de-allocate the current topic assignment and update the co-occurrence matrices before sampling

Convergence = model probability of training set becomes stable

Hyper-Parameters

T: number of topic

•α : prior on the topic-word distribution

generally larger (> 0.1)

‣ Multiple topics within a document

•β : prior on the document-topic distribution

generally small (< 0.01)

‣ Large vocabulary, but we want each topic to focus

• Analogous to k in add-k smoothing in N-gram LM

• Pseudo counts when computing:

High prior values → flatter distribution

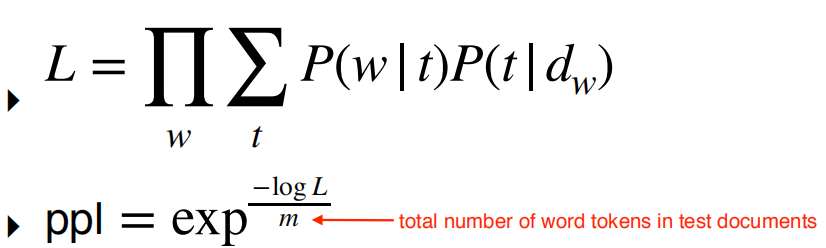
‣ a very very large value would lead to a uniform distribution

Low prior values → peaky distribution

## Evaluation

### Intrinsic evaluation:

‣ model logprob / perplexity on test documents



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

### Topic Coherence

• A better intrinsic evaluation method

• Measure how coherent the generated topics

• A good topic model is one that generates more coherent topics

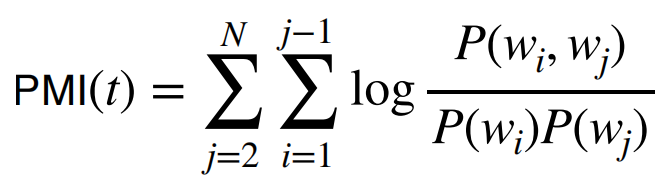
### Word Intrusion

Idea: inject one random word to a topic

• Ask users to guess which is the intruder word

### PMI

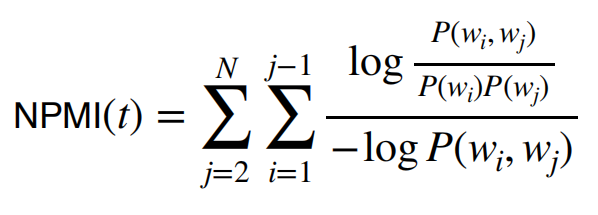
• Compute pairwise PMI of top-N words in a topic



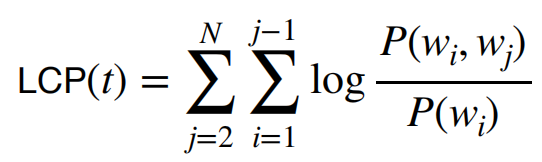
Sum logPMI for all word pairs in the topic

Variants

Normalised PMI



Conditional probability



Good correlation with human perception of topic coherence

• Better correlation if we use a different corpus to estimate PMI

## 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

• Use wikipedia article titles as labels

• Measure distance between a label and topic words based on document embeddings and word embeddings

# lecture 21: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

分类

• 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

步骤

• 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

作用

• 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)

## Extractive: Single-Doc

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

• 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

### Content Selection

• Not much data with ground truth extractive sentences

• Mostly unsupervised methods

• Goal: Find sentences that are important or salient

#### Method 1: TF-IDF

• Frequent words in a doc → salient

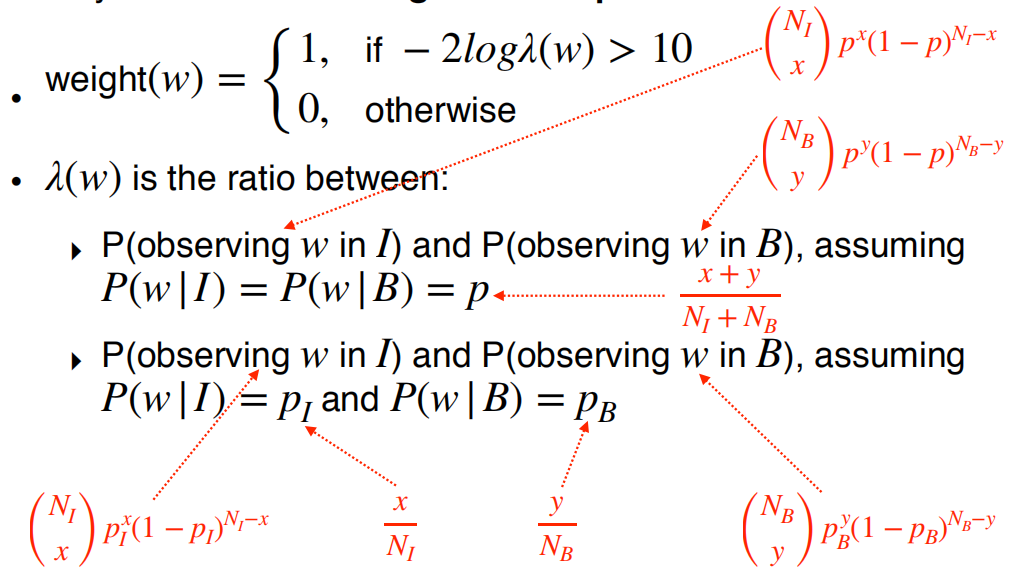
• But some generic words are very frequent but uninformative

‣ function words

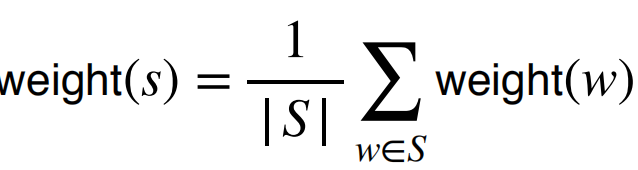
‣ stop words

#### Method 2: Log Likelihood Ratio

Intuition: a word is salient if its probability in the input corpus is very different to a background corpus



for sentence



#### 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

Use top-ranked sentences as extracted summary

‣ Saliency (tf-idf or log likelihood ratio)

‣ Centrality

#### 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)

• Nucleus more important than satellite

• A sentence that functions as a nucleus to more sentences = more salient

## Extractive: Multi-Doc

• Similar to single-document extractive summarisation system

• Challenges:

‣ Redundancy in terms of information

‣ Sentence ordering

• 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

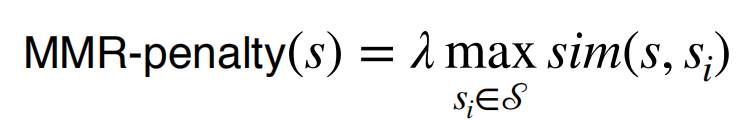
### MMR

MMR to make sure only non-redundant sentences are selected

• 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



Stop when a desired number of sentences are added

Create multiple simplified versions of sentences before extraction

### 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

• 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

train a neural network to generate summary

Data

• News headlines

• Document: First sentence of article

• Summary: News headline/title

• Technically more like a “headline generation task”

### Improvements

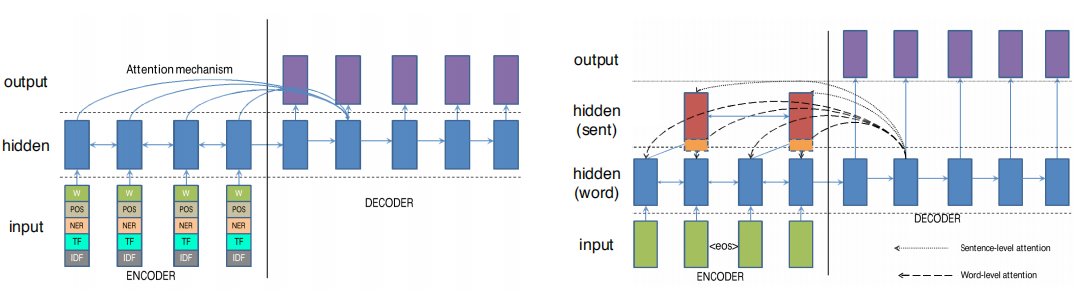
• Attention mechanism

• Richer word features: POS tags, NER tags, tf-idf

• Hierarchical encoders

‣ One LSTM for words

‣ Another LSTM for sentences



Issues

• Occasionally reproduce statements incorrectly

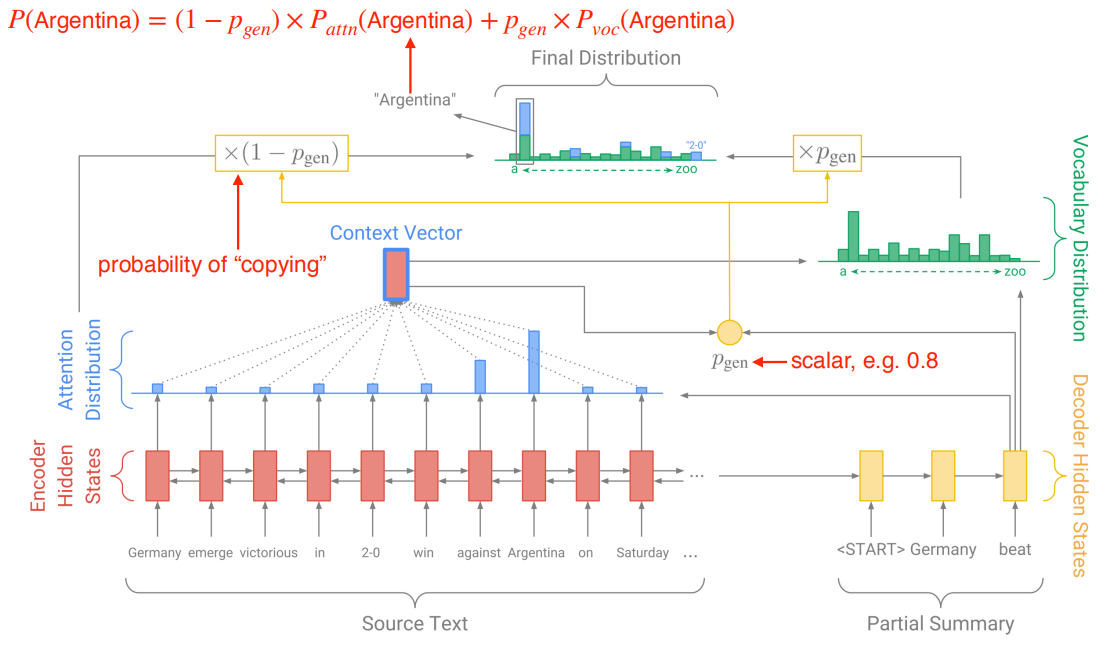
(hallucinate new details!)

• 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



### 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. smergle = out of vocabulary

‣ p(smergle) = attention probability + generation probability = attention probability

## Evaluation

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

