COMP90042 Natural Language Processing

Lecture 1 Introduction

Natural Language Processing

- **Interdisciplinary**(跨学科的) study that involves **linguitics**(语言学), **computer science** and **artificial intelligence**
- Aim: design algorithms to process and analyse human language data

Why process text?

- **unstructured** text contains massive information
- However, first need to understand the structure

Are Machines Intelligent yet?

- **Turing test** to assess whether a machine is intellignet
- no, the smartest conversational agent we have todat are far away from being truly intelligent

Models and Algorithms

- state machines
  - Formal models that consist of states, transitions between states and input. E.g. finite-state-automata

- formal language theory
  - regular grammars(正则文法), context-free grammars(上下文无关语法) to explain syntax

- machine learning
  - Hidden Markov models for understanding sequences
  - Logistic regressions, SVMs for classifying text
  - Neural networks (deep learning)

Challenges of Language

- Ambiguity
  - duck

- *I made her duck:*
  - *I cooked* *for her*
  - *I cooked* *belonging to her*
  - *I caused her to quickly lower her head or body*
  - *I waved my magic wand and turned her into a*
    - **duck** can be noun or verb
    - **her** can be a dative (与格) pronoun(代词) (i.e. indirect object to a verb) or possessive pronoun(物主代词)
    - **make** is syntactically ambiguous(句法歧义)
      - Transitive(及物动词) (takes one object: duck)
      - Ditransitive(双宾动词) (1st object: her, 2nd object: duck)
      - Can take a direct object and verb: object (her) is caused to perform the verbal action (duck)
- different languages
- contexts

Lecture 2 Preprocessing

Definitions

- **words**: Sequence of characters with a meaning and/or function
- **sentence**: "The student is enrolled at the University of Melbourne."
- **document**: one or more sentences
- **corpus**(语料库): a collection of documents
- **word token**: each *instance* of a word.
  - E.g. 9 word tokens in the example sentence
- **word type** *distinct* words.
  - Lexicon(词典): a group of word types
  - E.g. 8 word types in the example sentence

Why Preprocess

- most NLP applications have documents as inputs

- **key point**: language is *compositional*. As humans, we can break these documents into individual components. To understand language, a computer should do the same.
- preprocessing is the first step

Preprocessing Steps

- remove unwanted formatting(e.g. HTML)
- **sentence segmentation**: break documents into sentences
- **word tokenisation**: break sentences into words
- **word normalisation**: transform words into canonical forms
- **stopword removal**: delte unwanted words

Sentence Segmentation

- naive approach: break on sentence punctuation ([.?!])
  - but periods are used for abbreviations (U.S. dollar)

- second try: yse regex to require capital ([.?!][A-Z])
  - but abbreviations often followed by names (Mr. Brown)

- better: have lexicons
  - but difficult to enumerate all names and abbreviations
  - state-of-the-art uses machine learning, not rules

- binary classifier
  - look t every "." and decides whether it is the end of a sentence
    - decision trees, logistic regression

  - features
    - words before and after "."
    - word shapes:
      - uppercase, lowercase, ALL_CAPS, number
      - Character length

    - part-of-speech(词性) tags:
      - determiners(限定词) tend to start a sentence

Word Tokenisation

- different language have different structure and words
  - e.g. words typically seperated by a space in English. words written without spaces in Chinese

- MaxMatch algorithm
  - assumes an existing vocabulary (require lexicon)
  - greedily match the longest word in the vocabulary

    V = {墨,大,的,学,生,与,众,不,同, 墨大,学生,不同,与众不同}

    墨大的学生与众不同

    match 墨大, match 的, match 学生, match与众不同,

    move to 的 move to 学 move to 与 done

  - problems:
    - what the vocabulary is?
    - does not always work

    去 买 新西兰 花

    go buy New Zealand flowers

    去 买 新 西兰花

    go buy new broccoli

- subword tokenisation (byte-pair encoding)
  - colourless green ideas sleep furiously -> [colour] [less] [green] [idea] [s] [sleep] [furious] [ly]
  - one popular algorithm: byte-pair encoding(BPE)
  - core idea: iteratively merge frequent pairs of characters
  - advantage:
    - data-informed tokenisation
    - works for different languages
    - deals better with unkonw words

  - disadvantage:
    - larger vocabulary
    - creates non-sensical subwords

- byte-pair encoding
  - run 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

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

Word Normalisation

1. lower casing (Austrilia->australia)
2. removing morphology(词法) (cokking->cook)
3. correct spelling (definately->definitely)
4. expanding abbreviations (U.S.A->USA)
5. Goal:
   - reduce vocabulary
   - maps words into the same type

- inflectional morphology(屈折词法)
  - creates grammatical variants (但不改变词性)
  - English inflects nouns, verbs, adjectives
    - dog -> dogs
    - take -> takes
    - high -> higher
  - some languages have much richer inflectional morphology
- lemmatisation

    - removes any inflection to reach the uninflected form(the lemma)
        - speaking -> speak
    - some irregularities prevent trival solution:
        - poked -> poke (not pok)
        - stopping -> stop (not stopp)
        - watches -> watch (not watch)
        - was -> be (not wa)
    - a lexicon of lemmas needed for accurate lemmatisation
- derivational morphology(派生词法)
    - creates distinct words
    - English derivational *suffixes* often change the lexical category(后缀通常改变词性)
        - personal -> personally
        - final -> finalise
        - write -> writer
    - English derivational *prefixes* often change the meaning without changing the lexical category(前缀通常不改变词性)
        - write -> rewrite
        - healthy -> unhealthy
- stemming
    - Stemming strips off all **suffixes**, leaving a stem
        - E.g. automate, automatic, automation -> automat
        - often not a actual lexical item
    - less lexical sparsity than lemmatisation
    - popular in information retrieval
    - stem not always interpretable
- the Porter Stemmer
    - most popular stemmer for English
    - applies rewrite rules in stages
        - first strip *inflectional* suffixes (-ies -> -i)
        - then *derivational* suffixes (-isation -> -ise -> -i)
        - tidying up (e.g. controll -> control)
    - c(lowercase) = consonant(辅音: e.g. 'b', 'c', 'd')
    - v(lowercase) = vowel(原音: e.g. 'a','e','i','o','u')
    - C = a sequence of consonants(e.g. 'ss', 'tr')
    - V = a sequence of vowels(e.g. 'oo','ee','io')
    - a word has one of the four forms:
        - CVCV...C
        - CVCV...V

        - VCVC...C
        - VCVC...V
    - represent as(方括号中可有可无):
        - [C]VCVC...[V]
        - [C](VC)^m[V]
        - m = **measure**
        - TREE
            - = CV
            - = C(VC)^{0}V
            - [m=0]
        - TREES
            - = CVC
            - = C(VC)^1
            - [m=1]
        - TROUBLES
            - = CVCVC
            - = C(VC)^2
            - [m=2]
    - rules format: (condition)S1 -> S2
    - Always use the longest matching S1:
        - care**sses** -> caress
        - care**ss** -> caress
        - care**s** -> care
- fixing spelling errors
    - why
        - spelling errors create new, rare types
        - disrupt various kinds of linguistic analysis
        - spelling errors 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
        - normaliza -> normalise

        - U r so coool -> you are so cool

    - expanding abbreviations
        - US, U.S. -> United States
        - imho -> in my humble opinion

Stopward Removal

- stop words
    - definition: a list of words to be removed from the document
        - typically in bag-of-word(BOW) representations
        - Not appropriate when sequence is important

    - how to choose?
        - all *closed-class* or *function* words(e.g. the, a, of, for, he)
        - any high frequency words
        - NLTK, spaCy NLP toolkits

Conclusion

- preprocessing unavoidable in text analysis
- can have a major effect on downstream applications
- exact steps may vary depending on corpus, task
- simple rule-based systems work well, but rarely perfect
- language dependent

Lecture 3 N-gram Language Models

language models

- one application in NLP is about *explaining language*

    - why some sentences are more **fluent** than others

- measure 'goodness' using **probabilities** estimated by language models
- language models can be used for **generation**
- useful for *query completion*, *optical character recognition*, *machine translation*, *summarisation*, *dialugue systems*, etc.
- pretrained language models are the backbone of modern NLP systems

n-gram Language Models

- convert joint to conditional probabilities
    - goal is to get a probability for an arbitrary sequence of *m* words
    - apply chain rule to convert joint to condition:
      $P(w_1, w_2, \ldots, w_m) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_m|w_1, \ldots, w_{m-1})$

- the Markov Assumption
  - simplifying assumption: not condition on all previous words, look at a fixed size window
    - $P(w_m|w_1,\ldots,w_{m-1}) \approx P(w_m|w_{m-n+1},\ldots,w_{m-1})$
  - n = 1 -> unigram model(bag of word): $P(w_1,w_2,\ldots,w_m) = \prod_{i=1}^{m} P(w_i)$
  - n = 2 -> bigram model: $P(w_1,w_2,\ldots,w_m) = \prod_{i=1}^{m} P(w_i|w_{i-1})$
  - n = 3 -> trigram model: $P(w_1,w_2,\ldots,w_m) = \prod_{i=1}^{m} P(w_i|w_{i-2}w_{i-1})$
- maximum likelihood estimation
  - estimates based on counts in our corpus
  - unigram models: $P(w_i) = \frac{C(w_i)}{M}$
  - bigram models: $P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$
  - n-gram models: $P(w_i|w_{i-n+1}\ldots w_{i-1}) = \frac{C(w_{i-n+1}\ldots w_i)}{C(w_{i-n+1}\ldots w_{i-1})}$
- special tags to denote start and end of sequence:
  - <s> = sentence start
  - </s> = sentence end
  - don't want cross sentence combination
- trigram example:

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*) = ½ × 1 × ½ × ⅖ × ½ = 0.05

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>
P(*yes* | *no no*)	2/5	no no no no no no no no yes no no no no no yes
P(</s> | *no yes*)	1/2	no yes </s> yes no </s>

- problems
  - language has long distance effects - need large n
  - resulting probabilities are often very small
  - for unseen words, use special symbol to represent(e.g. <UNK>)

- need smoothing to solve unseen n-grams(whole term = 0)

Smoothing

1. give unseen events some probability
2. P(everything) = 1
3. many kinds: Laplacian(add-one) smoothing, add-k smoothing, absolute discounting, kneser-ney, etc.

- **Laplacian (add-one) smoothing**
  - idea: see each n-gram once more
  - unigram models: $P_{add1}(w_i) = \frac{C(w_i)+1}{M+|V|}$
  - bigram models: $P_{add1}(w_i|w_{i-1}) = \frac{C(w_i w_{i-1})+1}{C(w_{i-1})+|V|}$
  - example:

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

    What's the bigram probability *P(ate* | *rat*) under add-one smoothing?

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

    V = { the, rat, ate, cheese, </s> }

    What's the bigram probability *P(ate* | *cheese*) under add-one smoothing?

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

    <s> is not part of vocabulary because we never need to infer its conditional probability (e.g. P(<s> | …))

    Recall: P(yes no no yes) =
    P(*yes* | *<s> <s>*) × P(*no* | *<s> yes*) ×
    P(*no* | *yes no*) × P(*yes* | *no no*) ×
    P(*</s>* | *no yes*)

- **Add-k smoothing**
  - adding one is often too much
  - add a fraction k
  - AKA Lidstone smoothing: $P_{addk}(w_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-2},w_{i-2},w_i)+k}{C(w_{i-2}w_{i-1})+k|V|}$
  - example:

Context = *alleged*

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

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

0.391 x 20

(8 + 0.1) / (20 + 7 x 0.1)

(0 + 0.1) / (20 + 7 x 0.1)

26

- **absolute discounting**
  - take a **fixed** probability mass from observed n-gram counts
  - redistributes it to unseen n-grams
  - example:

Context = *alleged*

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

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$ effective counts	Lidstone smoothing, $\alpha = 0.1$ smoothed probability	Discounting, $d = 0.1$ effective counts	Discounting, $d = 0.1$ smoothed probability
impropriety	8	0.4	7.826	0.391	7.9	0.395
offense	5	0.25	4.928	0.246	4.9	0.245
damage	4	0.2	3.961	0.198	3.9	0.195
deficiencies	2	0.1	2.029	0.101	1.9	0.095
outbreak	1	0.05	1.063	0.053	0.9	0.045
infirmity	0	0	0.097	0.005	0.25	0.013
alleged	0	0	0.097	0.005	0.25	0.013
	20	1.0	20	1.0	20	1.0

8 - 0.1

(0.1 x 5) / 2

0.25 / 20

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

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

unigram probability for w_i
e.g. P(infirmity)

sum unigram probabilities for all words
that do not co-occur with context w_{i-1}
e.g. P(infirmity) + P(alleged)

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

29

- issue: Katz Backoff gives higher probability to unseen n-grams with higher conuts

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

- *I can't see without my reading* ___
- C(*reading*, *glasses*) = C(*reading*, *Francisco*) = 0
- C(*Fransciso*) > C(*glasses*)
- Katz backoff will give higher probability to *Francisco*

- **Kneser-Ney smoothing**
  - Redistribute probability mass based on the **versatility**(通用性) of the lower order n-gram
  - AKA continuation probability
  - what is versatility
    - high versatility → co-occurs with a lot of unique words, e.g. glasses: black glasses, men's glasses, etc.
    - low versatility → co-occurs with few unique words, e.g. francisco: san francisco
  - formula:

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

the amount of probability mass that has been discounted for context w_{i-1}

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

    - Intuitively the numerator of P_{cont} counts the number of unique w_{i-1} that co-occurs with

w_i
    - High continuation counts for glasses
    - Low continuation counts for Franciso

- **Interpolation**

  - A better way to combine different orders of n-gram models
  - Interpolated trigram model:
    $P_{IN}(w_i|w_{i-1}, w_{i-2}) = \lambda_3 P_3(w_i|w_{i-2}, w_{i-1}) + \lambda_2 P_2(w_i|w_{i-1}) + \lambda_1 P_1(w_i)$
  - λ learned from held out data

Conclusion

- N-gram is a simple but effective way to capture the predictability of language
- Can be trained in an unsupervised fashion
- Scalable to large corpora
- Require smoothing to be effective
- Can be used to fix spelling errors: build a word-level LM and use left/right context words to find the most appropriate substituting word

Lecture 4 Text Classification

Classification

- Input
  - a document *d*
  - a fixed output set of classes $C = \{c_1, c_2, \ldots c_k\}$

- Output

  - a predicted class *c*

- Text classification tasks

  - topic classification
  - sentiment analysis
  - native-language identification
  - natural language inference
  - automatic fact-checking
  - paraphrase

- Building a text classifier

  1. identify a task of interest
  2. collect an appropriate corpus
  3. carry out annotation
  4. select features

5. choose a machine learning algorithm
6. train model and tue hyper-parameters using held-out development data
7. repeat earlier steps as needed
8. train final model
9. evaluate model on held-out test data

Algorithms for Classification

- Choosing a classification algorithm
  - bias vs. variance
    - bias: assumptions made in the model
    - variance: sensitivity to training set
  - underlying assumptions: feature independence, etc.
  - complexity
  - speed
- Naïve Bayes
  - $P(C|F) \propto P(F|C)P(C)$
  - assume feature independence: $p(c_n|f_1 \ldots f_m) = \prod_{i=1}^{m} p(f_i|c_n)p(c_n)$
  - pros:
    - **fast** to train and classify
    - robust, low-variance → good for **low data** situations
    - optimal classifier if independence assumption is correct
    - extremely simple to implement
  - cons:
    - Independence assumption rarely holds
    - low accuracy compared to similar methods in most situations
    - **smoothing required** for unseen class/feature combinations(probability=0)
- 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(L2) or sparse(L1) weights
  - pros:
    - Unlike Naïve Bayes not confounded(打败) by diverse, correlated → better performance
  - cons:
    - slow to train
    - feature scaling needed
    - requires a lot of data to work well
    - Choosing regularisation strategy is important since **overfitting** is a big problem
- SVM

    - Finds hyperplane which separates the training data with maximum margin
    - 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 needed
        - deals poorly with class imbalance
        - interpretability
    - prior to deep learning, SVM very popular in NLP:
        - Non-linear kernel trick works well for text
        - NLP problems often involve large feature sets
- KNN
    - Classify based on majority class of k-nearest training examples in feature space
    - distance metric: euclidean distance, cosine distance, etc.
    - pros:
        - simple but effective
        - no training required
        - inherently multiclass
        - optimal classfier with infinite data
    - cons:
        - have to select *k*
        - issues with class imbalance
        - often slow(in finding neighbours)
        - features must be selected carefully
- Decision tree
    - Construct a tree where nodes correspond to tests on individual features
    - Leaves are final lcass decisions
    - based on greedy maximization of mutual information
    - pros:
        - fast to build and test
        - no need for feature scaling
        - good for small feature sets
        - handles non-linearly-sparable problems
    - cons:
        - in practice, not interpretable
        - highly redundant sub-trees
        - not compatitive 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
        - Great classifier for medium feature sets
        - Training easily parallelised
    - cons:
        - Interpretability
        - slow with large feature sets
- Neural networks
    - An interconnected set of nodes typically arranged in layers
    - Input layer (features), output layer (class probabilities), and one or more hidden layers
    - Each node performs a linear weighting of its inputs from previous layer, passes result through activation function to nodes in next layer
    - pros:
        - Extremely powerful, dominant method in NLP and vision
        - little feature engineering
    - cons:
        - not an off-the-shelf classifier
        - many hyper-parameters, difficult to optimize
        - slow to train
        - prone to overfitting

Evaluation

- accuracy
- precision & recall
    - precision = $\frac{tp}{tp+fp}$
    - recall = $\frac{tp}{tp+fn}$

	Classified As	
Class	A	B
A	79	13
B	8	10

False Positives (fp)

True Positives (tp)

False Negatives (fn)

B as “positive class”

Precision = correct classifications of B (tp)
/ total classifications as B (tp + fp)
= 10/(10 + 13) = 0.43

Recall = correct classifications of B (tp)
/ total instances of B (tp + fn)
= 10/(10 + 8) = 0.56

- F1 score
  - harmonic mean of precision and recall: $F1 = \frac{2 \times precision \times recall}{precision + recall}$
  - defined relative to a specific positive class
  - can be used as a heneral multiclass metric:
    - Macro-average: average f1 score across classes
    - Micro-average: calculate f1 score using sum of counts(=accuracy for multiclass problems)

Conclusion

- lots of algorithms
- *well-annotated, plentiful datasets and appropriate features* often more important than the specific algorithm used

Lecture 5 Part of Speech Tagging

Part of Speech

- word classes, morphological classes, syntactic categories
- Nouns, verbs, adjective, etc.
- information 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

- POS open classes vs. closed classes

- how readily do POS categories take on new words
- **open class**(quite often):
  - Nouns
    - proper (Australia) vs. common (wombat)
    - mass (rice) vs. count (bowls)
  - Verbs
    - Rich inflection (go/goes/going/gone/went)
    - Auxiliary verbs (be, have, and do in English)
    - Transitivity (wait versus hit versus give) — number of arguments
  - Adjectives
    - Gradable (happy) versus non-gradable (computational)
  - Adverbs
    - Manner (slowly)
    - Locative (here)
    - Degree (really)
    - Temporal (today)
- **closed classes**(not very often):
  - Prepositions: *in, on, with, for, of, over,...*
  - Particles: *off*
  - Determiners
    - articles: *a, an, the*
    - demonstratives: *this, that, these, those*
    - quantifiers: *each, every, some, two,...*
  - Pronouns
    - personal: *I, she, me*
    - possessivve: *my, his, her*
    - interrogative or *Wh*: *who, what, where...*
  - Conjunctions
    - coordinating: *and, or, but*
    - subordinating: *if, althougn, that*
  - Modal verbs
    - ability: *can, could*
    - permission: *can, may*
    - possibility: *may, might, could, will*
    - necessity: *must*
  - and more: negatives, politeness markers, etc.

- ambiguity
  - word types bolong to multiple classes

- POS depends on context
  - *Time flies like an arrow*
  - *Fruit flies like a banana*

Time	flies	like	an	arrow
noun	verb	preposition	determiner	noun

Fruit	flies	like	a	banana
noun	noun	verb	determiner	noun

Tagsets

- a compact representation of POS information
  - Usually ≤ 4 capitalized characters (e.g. NN = noun)
  - Often includes inflectional distinctions

- some major English tagsets
  - Brown(87 tags)
  - Penn Treebank(45 tags)
  - CLAWS/BNC(61 tags)
  - "Universal"(12 tags)

- at least one tagset for all major languages
- major Penn Treebank tags

NN	noun	VB	verb
JJ	adjective	RB	adverb
DT	determiner	CD	cardinal number
IN	preposition	PRP	personal pronoun
MD	modal	CC	coordinating conjunction
RP	particle	WH	wh-pronoun
TO	*to*		

  - open class
    - NN (noun singular, wombat)

        - NNS(plural, wombats)
        - NNP(proper, Australia)
        - NNPS(proper plural, Australians)

    - VB(verb infinitive(不定式，动词原形), eat)
        - VBP (1st /2nd person present, eat)
        - VBZ (3rd person singular, eats)
        - VBD (past tense, ate)
        - VBG (gerund, eating)
        - VBN (past participle, eaten)

    - JJ(adjective, nice)
        - JJR (comparative, nicer)
        - JJS (superlative, nicest)

    - RB(adverb, fast)
        - RBR (comparative, faster)
        - RBS (superlative, fatest)

  - closed class
    - PRP (pronoun personal, I)
        - PRP$ (possessive, my)

    - WP (Wh-pronoun, what)
        - WP$ (possessive, whose)
        - WDT(wh-determiner, which)
        - WRB (wh-adverb, where)

  - tagged test example:

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

Automatic Tagging

- 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 attribution (fiction vs. non-fiction)
    - POS tags can offer word sense disambiguation ‣ E.g. cross/NN vs cross/VB cross/JJ
    - Can use them to create larger structures(parsing)

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, e.g. verb subcategorisation (its arguments)
- Apply rules to narrow down to a single tag
    - E.g. If DT comes before word, then eliminate VB
    - Relies on some unambiguous contexts

- Large systems have 1000s of constraints

Statistical 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
        - Correctly resolves about 75% of ambiguity

    - Often considered the baseline for more complex approaches

- classifier-based tagging
    - use a standard discriminative classifier(logistic regression, neural network, etc.)
    - features
        - targetword
        - lexical context around the word
        - classified tags in the sentence

    - issues: suffer from error propagation
        - wropredictions from previous steps affect the next ones

- Hidden Markov Models
    - A basic sequential (or structured) model
    - Like sequential classifiers, use both previous tag and lexical evidence
    - Unlike classifiers, considers all possibilities of previous tag
    - Unlike classifiers, treat previous tag evidence and lexical evidence as independent from each other
        - Less sparsity

        - Fast algorithms for sequential prediction, i.e. finding the best tagging of entire word sequence

- problem: unkown words
    - big problem in morphologically rich languages(e.g. Turkish)
    - Can use things we've seen only once (hapax legomena) to best guess for things we've never seen before
        - Tend to be nouns, followed by verbs
        - Unlikely to be determiners

    - Can use sub-word representations to capture morphology (look for common affixes)

Conclusion

- POS is a fundamental intersection between linguistics and automatic text analysis
- provide usefull information for NLP applications
- methods applied are typical of language tasks in general, e.g. probabilistic, sequential machine learning

Lecture 6 Squence Tagging: Hidden Markov Models

- sequence labelling(structured prediction)
    - decompose process into individual word level steps
    - consider the whole sequence when learning and predicting(no error propagation)

Hidden Markov Model(HMM)

- Goal: obtain best tag sequence **t** from sentence **w**

    - $\hat{t} = argmax_t P(t|w)$
    - $\hat{t} = argmax_t \frac{P(w|t)P(t)}{P(w)} = argmax_t P(w|t)P(t)$ [Bayes]

- decompose
    - $P(w|t) = \prod_{i=1}^{n} P(t_i|t_{i-1})$ [prob. of a word depends only on the tag]

        - output independence: An observed event (word) depends only on the hidden state(tag)

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

        - Markov assumption: the current state(tag) depends only on previous state

- training
    - parameters(individual probabilities)
        - $P(w_i|t_i)$ = **emission** (O) probabilities
        - $P(t_i|t_{i-1})$ = **transition** (A) probabilities

    - training use Maximum Likelihood Estimation(MLE) → done by counting word frequencies

- $P(like|VB) = \frac{count(VB,like)}{count(VB)}$
- $P(NN|DT) = \frac{count(DT,NN)}{count(DT)}$
- using <s> to represent the start of a sentence
- for unseen (word,tag) or (tag,previous_tag) → smoothing
- **transition matrix**

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

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

- **emission (observation) matrix**

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

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

- prediction(decoding)
  - $\hat{t} = argmax_t \frac{P(w|t)P(t)}{P(w)} = argmax_t P(w|t)P(t) = argmax \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$
  - simple idea: a greedy fashion from left to right (wrong!) → a local classifier: error propagation
  - correct way: consider all possible tag combinations, evaluate them, take the max → complexity:T^N

The Viterbi Algorithm

- dynamic programming
- keep track of **scores for each tag** for previous words and check with different tags of the current word
- example:

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

Consider all possible tags, and take the maximum

max(P(will | NNP) * P(NNP | NNP) * s(NNP | Janet),
P(will | NNP) * P(NNP | MD) * s(MD | Janet),
…
P(will | NNP) * P(NNP | DT) * s(DT | Janet))

= P(will | NNP) * P(NNP | NNP) * 8.8544e-06

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

Going Backwards To Get The Best Tag Sequence

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

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

- Complexity: $O(T^2N)$, where T is the size of the tagset and N is the length of the sequence.
  - T * N matrix, each cell performs T operations
- why it works?
  - **independence assumptions** decomposes the problem
- use **log** probabilities to prevent underflow and convert multiplications to sums
- Vectorisation (use matrix-vector operations)

HMMs In Practice

- example above based on **bigrams** (first order HMM)

- state-of-the-art use trigrams (second order HMM)
  - $P(t) = \prod_{i=1}^{n} P(t_i|t_{i-1}, t_{i-2}) \rightarrow$ now complexity: $O(T^3N)$
- need to deal with sparsity: some tag trigram sequences might not be present in training data
  - Interpolation: $P(t_i|t_{i-1}, t_{i-2}) = \lambda_3 P(t_i|t_{i-1}, t_{i-2}) + \lambda_2 P(t_i|t_{i-1}) + \lambda_1 P(t_i)$
  - $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- highly effective for POS tagging
  - trigram HMM gets 96.5% accuracy
  - related models are state of the art
    - MEMMs 97%
    - CRFs 97.6%
    - Deep CRF 97.9%

Generative vs. Discriminative Taggers

- HMM is **generative**
  - $\hat{T} = argmax_T P(T|W) = argmax_T P(W|T)P(T) = argmax_T \prod_{i=1}^{N} P(w_i|t_i)P(t_i|t_{i-1})$
  - trained HMM can generate data(sentences)
  - allows for unsupervised HMMs: learn model without any tagged data!
- **discriminative** models
  - model $P(T|W)$ directly
    - $\hat{T} = argmax_T P(T|W) = argmax_T \prod_{i=1}^{N} P(t_i|w_i, t_{i-1})$
  - supports richer feature set
  - generally better accuracy when trained over large supervised datasets
  - E.g., Maximum Entropy Markov Model (MEMM), Conditional random field (CRF)
  - most deep learning models of sequences are discriminative

Conclusion

- HMMS are a simple but effective way to perform sequence labelling
- competitive and fase. Natural baseline for other sequence labelling models
- Main drawback: not very flexible in terms of feature representation, compared to MEMMs and CRFs

Lecture 7 Deep Learning for NLP: Feedforward Networks

Feedforward Neural Networks Basics

- feedforward NN

- multilayer perceptrons
- Each arrow carries a weight, reflecting its importance
- certain layers have non-linear activation functions

- neuron
  - each neuron is a function
  - given input x, computes real-value (scalar) h: $h = tanh(\sum_j w_j x_j + b)$
  - scales input (with weights, w) and adds offset (bias, b)
  - applies non-linear function, such as logistic sigmoid, hyperbolic sigmoid (tanh), or rectified linear unit → to achieve non-linear decision boundary
  - *w* and *b* are parameters of the model

- Matrix Vector Notation
  - $\vec{h_i} = tanh(W\vec{x} + \vec{b})$
  - Where *W* is a matrix comprising the weight vectors, and $\vec{b}$ is a vector of all bias terms
  - non-linear function applied element-wise

- output layer

    - binary classification → sigmoid activation function
    - multi-class classification → softmax function

- regularisation
    - many parameters, overfits easily
    - low bias, high variance
    - regularisation is very important in NNs
    - L1-norm: sum of absolute values of all parameters
    - L2-norm: sum of squares
    - dropout: randomly zero-out some neurons of a layer

- dropout
    - dropout rate: percentage of neuron to randomly zero-out
    - mostly used in hidden layers
    - why work?
        - It prevents the model from being over-reliant on certain neurons
        - It penalises large parameter weights
        - It introduces noise into the network

Applications in NLP

- topic classification
    - given a document, classify it into a predifined set of topics
    - input: bag-of-words
    - possible improvements:
        - add 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

- NNs as lanuage model
    - word embeddings
        - Maps discrete word symbols to continuous vectors in a relatively low dimensional space
        - allow the model to capture similarity between words

    - concatenate input words embeddings as input to NNs
    - NNs output give probability distribution over all words in the vocabulary
    - architecture:

- advantages
  - count-based n-gram models
    - cheap to train (just collect counts)
    - problems with sparsity and scaling to larger contexts
    - don't adequately capture properties of words (grammatical and semantic similarity), e.g., film vs movie
  - FFNN n-gram models
    - automatically capture word properties, leading to more robust estimates
- limitations
  - slow to train
  - capture only limited context
  - unable to handle unseen words(no embeddings!)

Convolutional Networks

- commonly used in CV
- identify indicative local predictors
- combine them to produce a fixed-size representation
- in NLP:
  - use sliding widow over a sequence
  - W = convolution filter
  - max-pool to produce a fixed-size representation

Conclusion

- pros:
  - excellent performance
  - less hand-enginerring of features
  - flexible - customised architecture for different tasks

- cons
  - much slower than classical ML models, needs GPU
  - lost of parameters due to vocabulary size
  - data hungry, not good on tiny data sets
    - pre-training on big corpora helps

Lecture 8 Deep Learning for NLP: Recurrent Networks

Problem in n-gram language models: limited context(only a few words)

Recurrent Neural Networks(RNN)

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

recurring function

$$s_i = f(s_{i-1}, x_i)$$

new state previous state input

$$s_{i+1} = f(s_i, x_{i+1})$$

same recurring function

$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$

new state previous state input

- RNN unrolled

$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$
$$y_i = \sigma(W_y s_i)$$

← "Simple RNN"

same parameters W_s, W_x, b, W_y are used across all time steps

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

- Vocabulary: [*a*, *cow*, *eats*, *grass*]
- Training Example: *a cow eats grass*

$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$
$$y_i = \mathrm{softmax}(W_y s_i)$$

$$L_1 = -\log P(0.30)$$
$$L_2 = -\log P(0.50)$$
$$L_3 = -\log P(0.20)$$

$$L_{\text{total}} = L_1 + L_2 + L_3$$

- RNN as language model

- potential problems with the generation approach
  - Mismatch between training and decoding: model only trained with fluent sentences
  - error propagation: unable to recover from errors in intermediate steps → each output character is conditioned on the previous true character. However during testing, the model needs to rely on its own previous predictions
  - tends to generate 'bland' or 'generic' language: large amount of generic words in training data

LSTM

RNN seems to have the capability to model infinite context. In practice it can't capture long-range dependencies due to **vanishing gradients**: Gradients in later steps diminish quickly during backpropagation, earlier inputs do not get much update.

The repeating module in a standard RNN contains a single layer.

The repeating module in an LSTM contains four interacting layers.

- LSTM is introduced to solve vanishing gradients, only with a few more parameters and produce pretty good word embeddings
- Core idea: have “memory cells” that preserve gradients across time
- Access to the memory cells is controlled by “gates”
- For each input, a gate decides:
  - how much the new input should be written to the memory cell
  - and how much content of the current memory cell should be forgotten

- gating vector
  - A gate ***g*** is a vector
    - each element has values between 0 to 1
  - **g** is multiplied component-wise with vector **v**, to determine how much information to keep for **v**
  - Use sigmoid function to produce **g**: values between 0 to 1

- forget gate

  - controls how much information to forget in the memory cell C_{t-1}

- input gate

  - controls how much new information to put into memory cell

- update memory cell

- use the forget gate and input gate to update memory cell

- output gate

- controls how much to distill(提取，蒸馏) the content of the memory cell to create the next state h_t

- summary

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

- disadvantage

- still unable to caputer very long range dependencies
- much slower than simple RNNs due to a lot more computation steps

Applications

- Shakespeare generator
- Wikipedia generator
- code generator
- text classification: particularly suited for tasks where order of words matter
- sequence labelling
- varaints
  - Peephole connections: allow gates to look at cell state: $f_t = \sigma(W_f[C_{t-1}, h_{t-1}, x_t] + b_f)$
  - Gated Recurrent Unit(GRU): simplified vatiant with only 2 gates and no memory cell(less computation but similar performance)

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- multi-layer LSTM

- bidirectional LSTM

$$y_i = \text{softmax}(W_s[s_i, u_i])$$

use information from both
forward and backward LSTM

Conclusion

- pros
  - Has the ability to capture long range contexts
  - Just like feedforward networks: flexible

- cons
  - slower than FFNN due to sequential processing
  - In practice doesn't capture long range dependency very well
  - In practice also doesn't stack well (multi-layer LSTM)
  - less popular due to more advanced architectures: Transformer

Lecture 9 Lexical Semantics

Sentiment Analysis

- bag of words, KNN classifier
- training data:
  - 'this is a good movie' →↑
  - 'this is a great movie' →↑
  - 'this is a terrible film' →↓

- testing: 'this is a wonderful film' →↓
- problems

- the model does not know 'movie' and 'film' are synonyms.
- 'film' only appears in negative examples, learnt as negative word
- 'wonderful' is not in the vocabulary

- comparing words directly does not work, how to compare **word meanings** instead?
- solution: add this information explicitly though a **lexical database**

Lexical Database

- lexical semantics
  - How the meanings of words connect to one another.
  - Manually constructed resources: lexical database

- what is meaning
  - dictionary definition
    - but necessarily circular
    - only useful if meaning is alread understood

      red *n.* the color of blood or a ruby.
      blood *n.* the red liquid that circulates in the heart, arteries and veins of animals.

  - relationships with other words

- definitions
  - word sense: one aspect of the meaning of a word, if a word has multiple senses, it is **polysemous**(多义的)

    mouse^1 : .... a *mouse* controlling a computer system in 1968.
    mouse^2 : .... a quiet animal like a *mouse*

  - meaning through dictionary
    - gloss: textual definition of a sense, given by dictionary

  - meaning through relations

        - Synonymy: near identical meaning
            - vomit vs. throw up
            - big vs. large
        - Antonymy: opposite meaning
            - long vs. short
            - big vs. little
        - Hypernymy: is-a relation
            - cat is an animal
            - mango is a fruit
        - Meronymy: part-whole relation
            - leg is part of a chair
            - wheel is part of a car

- WordNet
    - A database of lexical relations
    - English WordNet includes ~120,000 nouns, ~12,000 verbs, ~21,000 adjectives, ~4,000 adverbs
    - On average: noun has 1.23 senses; verbs 2.16
    - WordNets available in most major languages (www.globalwordnet.org, https://babelnet.org/)
    - English version freely available (accessible via NLTK)
    - synsets (words group by senses)
        - Nodes of WordNet are not words or lemmas, but **senses**
        - They are represented by sets of synonyms, or **synsets**
        - Bass synsets:
            - {bass1, deep6}
            - {bass6, bass voice1, basso2}
        - Another synset:
            - {chump1, fool2, gull1, mark9, patsy1, fall guy1, sucker1, soft touch1, mug2}
            - Gloss: a person who is gullible and easy to take advantage of
    - hypernymy chain example

```
bass³, basso (an adult male singer with the lowest voice)
=> singer, vocalist, vocalizer, vocaliser
   => musician, instrumentalist, player
      => performer, performing artist
         => entertainer
            => person, individual, someone...
               => organism, being
                  => living thing, animate thing,
                     => whole, unit
                        => object, physical object
                           => physical entity
                              => entity
```

```
bass⁷ (member with the lowest range of a family of instruments)
=> musical instrument, instrument
   => device
      => instrumentality, instrumentation
         => artifact, artefact
            => whole, unit
               => object, physical object
                  => physical entity
                     => entity
```

Word Similarity

can use lexical database to estimate word similarity

- paths
  - based on path length
  - pathlen(c_1, c_2) = 1 + edge length in the shortest path between sense c_1, c_2 (a node in the Wordnet graph is a synset(sense), not a word)
  - similarity between two synsets: $simpath(c_1, c_2) = \frac{1}{pathlen(c_1, c_2)}$
  - similarity between two words: $wordsim(w_1, w_2) = max_{c_1 \in senses(w_1), c_2 \in sensesw_2} simpath(c_1, c_2)$

Examples

Each node is a synset!
For simplicity we use just the representative word

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

simpath(*nickel,coin*) = 1/2 = 0.5

simpath(*nickel,currency*)
= 1/4 = 0.25

simpath(*nickel,money*)
= 1/6 = 0.17

simpath(*nickel,Richter scale*)
= 1/8 = 0.13

- problem
  - edges vary widely in actual semantic distance: much bigger jumps near top of hierarchy
- solution: include depth information

- simwup
  - Use path to find lowest common subsumer (LCS)
  - Compare using depths
  - $simwup(c_1, c_2) = \frac{2 \times depth(LCS(c_1, c_2))}{depth(c_1) + depth(c_2)}$ (high simwup when parent is deep, senses are shallow)

simwup(*nickel,money*) =

2*2 / (6+3) = 0.44

simwup(*dime,Richter scale*) =

2*1 / (6+3) = 0.22

  - still poor semantic distance metric
  - Nodes high in the hierarchy is very abstract or general (need to penalise similarity)

- concept probability of a node
  - Intuition:
    - general node → high concept probability (e.g. object)
    - narrow node → low concept probability (e.g. vocalist)

- Find all the children ∈ node, and sum up their unigram probabilities: $P(c) = \frac{\sum_{s \in child(c)} count(s)}{N}$
- child(c): synsets that are children of c
- child(*geological-formation*) = {*hill, ridge, grotto, coast, natural elevation, cave, shore*}
- child(*natural elevation*) = {*hill, ridge*}

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

- similarity with information content
  - $IC = -logP(c)$ (general concept = small values, narrow concept = large values)
  - $simlin(c_1, c_2) = \frac{2 \times IC(LCS(c_1,c_2))}{IC(c_1)+IC(c_2)}$ (high simlin when: concept of parent is narrow, concept of senses are general)
  - example

$$\text{simlin}(\text{hill}, \text{coast}) = \frac{2 \times -\log P(\text{geological-formation})}{-\log P(\text{hill}) - \log P(\text{coast})} = \frac{-2 \log 0.00176}{-\log 0.0000189 - \log 0.0000216}$$

If LCS is *entity*, -2 log(0.395)!

Word Sense Disambiguation

- Task: selects the correct sense for words in a sentence
- Baseline: Assume the most popular sense
- Good WSD potentially useful for many tasks
  - Knowing which sense of *mouse* is used in a sentence is important!
  - Less popular nowadays; because sense information is implicitly captured by contextual representations

- 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? (in practice small)

  - Requires sense-tagged corpora
    - E.g. SENSEVAL, SEMCOR (available in NLTK)
    - Very time consuming to create!

- unsupervised: Lesk
  - Lesk: Choose sense whose WordNet gloss overlaps most with the context
  - The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

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

Conclusion

- Creation of lexical database involves expert curation (linguists)
- Modern methods attempt to derive semantic information directly from corpora, without human intervention

Lecture 10 Distributional Semantics

Lexical Databases Problems

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

Distributional Hypothesis

- learn word meanings by its neighbouring context
- Document co-occurrence indicative of topic (document as context)
  - E.g. voting and politics
  - if we observe documents, we can find that these two words often co-occur in a document.
  - differnet words' co-occurrence document reflect relations between words and topics
- Local context reflects its meaning (word window as context)
  - E.g. eat a pizza vs. eat a burger
  - *pizza* and *burger* have the same context *eat*
  - two words both a meaning that relates to *eat*

Guessing Meaning from Context

- learn unknown word from its usage
  - *tezgüino*

(14.1) A bottle of ____ is on the table.

(14.2) Everybody likes ____.

(14.3) Don't have ____ before you drive.

(14.4) We make ____ out of corn.

- look at words that share similar contexts

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

Word Vectors

- in the above table, each row can be thought of a word vector (if a word appears in a sentence)
- It describes the distributional properties of a word
  - i.e. encodes information about its context words

- Capture all sorts of semantic relationships (synonymy, analogy, etc)

Word Embeddings

other ways to produce word vectors: count-based and neural methods.

Count-based Methods

- **document as context**: the Vector Sapce Model (VSM)

  - Core idea: represent word meaning as a vector
  - Consider documents as context
  - One matrix, two viewpoints (row: document vector, column: word vector)

- Documents represented by their words
- Words represented by their documents

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

- manipulating the values
  - Weighting the values (not only raw count)
  - Creating low-dimensional dense vectors
    - for 1 million documents, the dimension of word vector is 1 million. word vectors are sparse since a lot of 0s.

- Tf-idf (Term Frequency(词频)-Inverse Document Frequency(逆向文件频率))
  - Standard weighting scheme for information retrieval
  - Discounts common words (contain less useful information)
  - $idf_w = log\frac{|D|}{df_w}$ ($|D|$: total #docs, df_w: #docs that has w)

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

tf matrix

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

tf-idf matrix

- dimensionality reduction
  - Term-document matrices are very sparse (a lot of 0s)
  - Dimensionality reduction: create shorter, denser vectors
  - More practical (less features)
  - Remove noise (less overfitting)
  - Singular Value Decomposition (SVD)
    - $A = U \sum V^T$

  - truncating - Latent Semantic Analysis (LSA)
    - Truncating U, $\sum$, and V to *k* dimensions produces best possible *k* rank approximation of original matrix
    - U_k is a new low dimensional representation of words
    - Typical values for *k* are 100-5000

- **word as context**
  - how often words appear with other words
    - In some predefined context (e.g. a window of N words)
  - The obvious problem with raw frequency:
    - dominated by common words ('the')
    - can't use tf-idf since no documents involved, instead use PMI

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

- Pointwise Mutual Information (PMI)
  - For two events x and y, PMI computes the discrepancy(差异) between:
    - Their joint distribution = $P(x, y)$
    - Their individual distributions (assuming independence) = $P(x)P(y)$
    - $PMI(x, y) = log_2 \frac{P(x,y)}{P(x)P(y)}$
    - example:

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

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

$$P(x, y) = \frac{\mathrm{count}(x, y)}{\Sigma}$$

$$P(x) = \frac{\Sigma_x}{\Sigma}$$

$$P(y) = \frac{\Sigma_y}{\Sigma}$$

$x =$ state, $y =$ country

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

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

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

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

  - PMI matrix
    - PMI does a better job of capturing semantics
      - E.g. heaven and hell
    - very biased towards rare word pairs ($P(x)P(y)$ very small)
    - doesn't handle unseen pairs well ($log_2 0 = -\infty$)
  - PMI tricks

        - Zero all negative values (Positive PMI)
            - Avoid –inf and unreliable negative values
        - Counter bias towards rare events
            - normalised PMI: $\frac{PMI(x,y)}{-logP(x,y)}$
    - can still apply SVD
        - Regardless of whether we use document or word as context, SVD can be applied to create dense vectors

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

tf-idf matrix

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

PPMI matrix

- word vs. document as context
    - 如果我们采用 TF-IDF，我们捕获到的语义学信息会更加宽泛，通常和某种主题关联；如果我们采用 PMI/PPMI，我们捕获到的词向量更多是关于局部单词上下文的语义学信息。

Neural Methods

In lecture 7 and 8, neural networks are designed for other tasks (classification, language modelling), word embeddings are just a **by-product**.
Desiderata(理想模型): unsupervised(no labelling cost) and efficient(can be used in various downstream tasks)

- Word2Vec
    - Core idea
        - *You shall know a word by the company it keeps*
        - Predict a word using context words
    - Framed as learning a classifier
        - **Skip-gram**: predict surrounding words of target word

- **CBOW**: predict target word using surrounding words

Two methods are inverse to each other

- use surrounding words with L position, L=2 above
- skip-gram model
  - predicts each neighbouring word given target word
  - total probability: $\prod_{l\in -L,\ldots,-1,1,\ldots,L} P(w_{t+l}|w_t)$
  - using a logistic regression model: $P(life|rests) = \frac{exp(W_{rests}\cdot C_{life})}{\sum_{u\in V} exp(W_{rests}\cdot C_u)}$ (W_{rests}: word embedding of rests, C_{life} : word embedding of life, $\cdot$: dot product)
  - embedding parameterisation
    - two word embedding matrices (*W* and *C*): target word embedding, context word embedding

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

  - structure:
    - 输入层是目标单词 w_t 的 one-hot 向量，其维度为 $1\times|V|$，所以其中只会有一个值被激活，即目标单词对应的位置索引（例如：x_j）。然后，经过目标单词嵌入矩阵 $W_{|V|\times d}$ 将其投影为一个 $1\times d$ 的嵌入向量。之后，我们将该嵌入向量和上下文单词嵌入矩阵 $C_{d\times|V|}$ 相乘，从而得到上下文单词 w_{t-1} 的一个 $1\times|V|$ 的概率向量，其中每个元素代表词汇表中相应位置的单词在这里出现的概率。我们可以对上下文单词 w_{t+1} 进行相同的操作。然后，我们可以通过计算交叉熵来得到模型的损失。

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

$$P(\text{life} \mid \text{rests}) = \frac{\exp(W_{\text{rests}} \cdot C_{\text{life}})}{\sum_{u \in V} \exp(W_{\text{rests}} \cdot C_u)}$$

  - Reduce problem to binary classification
    - (life, rests) → real context word
    - (aardvark, rests) → non-context word
    - How to draw non-context word or negative samples?
    - Randomly from V

  - negative sampling
    - 假设我们有以上例句，并且上下文窗口为目标单词前后各 2 个相邻单词。这里，我们的目标单词为 “*apricot*”，上下文单词为 “*tablespoon*”、“*of*”、“*jam*” 和 “*a*”。
    - 正样本非常简单，就是由窗口中的目标单词和上下文单词构成的单词对。负样本的获取也非常简单，我们只是通过简单地将目标单词与从词汇表中随机抽取得到的单词进行配对，例如，这里我们进行 8 次随机抽样，从而得到 8 个负样本。这里我们的目标单词是 “*apricot*”，我们也可以对其他目标单词进行相同的操作。
    - 我们将词嵌入 t 和 c 的点积 $t \cdot c$ 作为 logistic 函数的输入（即 x）。如果二者的点积 $t \cdot c$ 非常大（例如：“*apricot*” 和 “*jam*”），从上面的函数图像可以看到，我们得到的概率将非常接近 1，因为我们希望 最大化 目标单词和真实上下文单词之间的相似度。对于负样本，我们计算时用 1 减去 logistic 函数，目的是 最小化 目标单词和非上下文单词之间的相似度。

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    t     c3       c4
```

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

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

$$\frac{1}{1+e^{-x}}$$

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$ ← **maximise** similarity between target word and real context words

$$P(-|t,c) = 1 - \frac{1}{1+e^{-t\cdot c}}$$ ← **minimise** similarity between target word and non-context words

32

- skip-gram loss
  - $L(\theta) = -[\sum_{(t,c)\in+} \log P(+ \mid t,c) + \sum_{(t,c)\in-} \log P(- \mid t,c)]$
  - in practice, use *k* negative examples:
    $L(\theta) = -[logP(+|t,c) + \sum_{i=1}^{k} logP(-|t,n_i)]$
- Desiderata
  - Unsupervised
    - Unlabelled corpus
  - Efficient
    - Negative sampling (avoid softmax over full vocabulary)
    - Scales to very very large corpus

Problems with word vectors/embeddings (count and neural methods)?

- Difficult to quantify the quality of word vectors
- Don't capture polysemous(多义) words, since one embedding for one word

Evaluation

- word similarity
  - Measure similarity of two words using cosine similarity
  - Compare predicted similarity with human intuition
  - Datasets
    - WordSim-353 are pairs of nouns with judged relatedness
    - SimLex-999 also covers verbs and adjectives
- word analogy(单词类比)
  - Man is to King as Woman is to ???
  - v(Man) - v(King) = v(Woman) - v(???)
  - v(???) = v(Woman) - v(Man) + v(King)
  - Find word whose embedding is closest to v(Woman) - v(Man) + v(King)

- Word2Vec embedding space
  - Word2Vec embeddings show interesting geometry
  - Explains why they are **good in word analogy task**

- downstream tasks
  - use pre-trained embedding in other downstream tasks and observe performance
  - best way to evaluate

- general findings
  - neural better than count-based
  - **contextual word representation** work even better
  - Dynamic word vectors that change depending on context

Software

- Word2Vec
  - C implementation of Skip-gram and CBOW: https://code.google.com/archive/p/word2vec/

- GenSim
  - Python library with many methods include LSI, topic models and Skip-gram/CBOW: https://radimrehurek.com/gensim/index.html

- GLOVE
  - http://nlp.stanford.edu/projects/glove/

Lecture 11 Contextual Representation

word embeddings from previous lectures:

- each word type has exactly one representation e.g. word2vec
- the same representation regardless of the context of the word
- does not capture multiple senses of words

Contextual representation represents words based on context. Pre-trained contextual representation works really well for downstream applications.
RNN language model: only captures context to the left

use bidirectional RNN

Embeddings from Language Models(ELMo)

- a bidirectional, multi-layer LSTM language model over 1B word corpus
- Combine hidden states from multiple layers of LSTM for downstream tasks
  - Prior studies use only top layer information

- Improves task performance significantly
- ELMo structure
  - Number of LSTM layers = 2
  - LSTM hidden dimension = 4096
  - character convolutional networks (CNN) to create word embeddings → no unknown words

- learn embedding for every character and use padding to ensure same length word embedding, then feed to a max pooling layer to create a representation of a word based on character embeddings

- extract contextual representation
  - get embedding of 'stick' in 'let's stick to'
  - feed words in the sentence to pre-trained ELMo (in forward and backward language model)

  - concatenate hidden layers from LSTM, multiply each with a weight, them sum them to get a ELMo embedding

- downstream task: POS tagging

$$s_i = \tanh(W_s s_{i-1} + (W_x x_i \oplus e_i) + b)$$

  - concatenate ELMp embedding with input embedding in downstream task
  - train with the downstream RNN
  - 通常我们不会对 ELMo 中的语言模型进行训练，我们固定其隐藏状态，即双向 LSTM 中的参数，我们要学习的只是 ELMo 中的最后一步加权求和时的权重参数（即之前提到的s_1）。在我们的词性标注任务开始时，我们将这些权重设置为一些随机值，然后我们根据任务的表现来更新这些权重

- ELMo performance

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

  - SQuAD：一个非常著名的问答数据集（QA
  - SNLI：文本蕴含数据集（textual entailment）
  - SRL：语义角色标注数据集（semantic role labelling）
  - Coref：共指消解数据集（coreference resolution）
  - NER：命名实体识别数据集（named entity recognition）
  - SST-5：情感分析数据集（sentiment analysis)

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

- contextual vs. non-contextual

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

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

  - Non-contextual only find similar words
  - Contextual find similar sentences, where the target word has similar meaning
  - contextual embedding doog in word sense disambiguation

- disadvantages of contextual embeddings
  - Difficult to do intrinsic evaluation (e.g. word similarity, analogy)
  - Interpretability
  - Computationally expensive to train large-scale contextual embeddings

Bidirectional Encoder Representations from Transformers (BERT)

- disadvantages of RNNs

- Sequential processing: difficult to scale to very large corpus or models
- RNN language models run left to right (capturesonly one side of context)
- Bidirectional RNNs help, but they only capture surface bidirectional representations (no interaction between forward and backward language model, simple concatenation)

23

- BERT
  - Uses self-attention networks (aka Transformers) to capture dependencies between words
    - No sequential processing (easier to scale to large corpus)

  - Masked language model objective to capture deep bidirectional representations (forward and backward not independent)
  - Loses the ability to generate language
  - Not an issue if the goal is to learn contextual representations
  - BERT vs. ELMo
    - In ELMo, no interaction between forward and backward model
    - In BERT, Transformer sees all context words of the target word, then calculate an aggregate representation(集合表示).
    - 由于这里我们使用单个模型来捕获两个方向的上下文信息，模型可以捕获到不同的两侧上下文单词之间更深层次的关系。

ELMo

BERT

?? We'll come back to describe Transformers in the last part of the lecture

- masked language model
  - 'Mask' out k% of tokens at random
  - Objective: predict the masked words
  - use both left and right context to predict

lecture *and*

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

- 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)
  - labelling not needed. use neighbouring sentences from corpus as positive samples, and randomly sample sentences as negative samples.

Sentence A: *Today we have a lecture on NLP.*
Sentence B: *It is an interesting lecture.*
Label: IsNextSentence

Sentence A: *Today we have a lecture on NLP.*
Sentence B: *Polar bears are white.*
Label: NotNextSentence

- training
  - WordPiece (subword) Tokenisation (like BPE, but WordPiece based on probability rather than frequency)
  - Multiple layers of transformers to learn contextual representations
  - BERT is pretrained on Wikipedia+BookCorpus
  - Training takes multiple GPUs over several days

- how to use
  - given a pre-trained BERT, continue training (fine-tune) it on downstream tasks
  - Add a classification layer on top of the contextual representations

  - during fine-tuning, parameters of the whole network are updated
  - use [CLS] representation for downstream tasks
  - example: spam detection

- 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
- BERT performance

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

Transformers

Use **attention** instead of using RNNs (or CNNs) to capture dependencies between words. An example of using **attention** to capture dependencies between words:

假设现在我们有一个句子 “*I made her duck*”，我们希望计算其中单词 “*made*” 的上下文嵌入。我们需要先分别单独观察单词 “*made*” 周围的单词，例如，对于单词 “*I*”，我们将结合单词 “*made*” 和 “*I*”，组合得到一个向量（第二层中的亮紫色向量）。这个过程称为 注意力（attention），因为我们这里将注意力放在一个特定单词（“*I*”）上。然后，对于第二个单词 “*her*” 进行相同的操作，得到一个表征了单词 “*made*” 和 “*her*” 的复合向量（第二层中的亮黄色向量）。同理，对于最后一个单词 “*duck*” 也一样，我们得到一个复合向量（第二层中的桃红色向量）。这三个复合向量分别捕获了单词 “*made*” 周围的三个上下文单词。然后，我们对这三个复合向量进行加权求和，得到单词 “*made*” 的上下文表示。

这就是注意力机制的核心思想。所以，可以看到，这里并没有涉及序列处理，因为假如现在我们要计算单词 “*duck*” 的上下文表示，我们可以直接计算其周围单词的注意力组合，而无需先计算出其前一个单词 “*her*” 的上下文表示。实际上，第二层的三个复合向量可以采用**并行方式计算**，因此，这里不涉及序列处理，这也是和基于 RNN 这种序列模型的 ELMo 相比，BERT 能够很好地扩展到大型语料库上的原因。

- self-attention via Query, Key, Value
  - Input:
    - query(查询向量) q (e.g. made) → target word vector
    - key(键向量) k and value(值向量) v (e.g. her) → context word vector
  - Query, key and value are all vectors
    - inear projections from embeddings
  - Comparison between **query vector** of target word (made) and **key vectors** of context words to compute weights
  - Contextual representation of **target word** = weighted sum of **value vectors** of **context**

words and target word

- $c_{made} = 0.1v_I + 0.5v_{made} + 0.2v_{her} + 0.3v_{duck}$

- $A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} \times v_i$
  - $\frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}}$: softmax (weights of value vectors)
  - $e^{q \cdot k_i}$: query and key comparisons
- if multiple queries, stack them in a matrix to speed up calculation:
  $A(Q, K, V) = softmax(QK^T)V$
- Uses scaled dot-product to prevent values from growing too large
  - $A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$, where d_k is the dimension of query and key vectors

- Multi-Head Attention

- Only one attention for each word pair
- Uses multi-head attention to allow multiple interactions
- W_i : linear projection of Q/K/V for each head
- 此前我们看到的是，**每一个单词对只有一个注意力（attention）**，例如：对于单词 “*made*”，我们在计算注意力时，每次只考虑一个上下文单词（例如：“*I*”）。但是，论文作者还提出了一种 多头注意力机制（multi-head attention），就是简单地将这个步骤重复多次以允许多次交互。

- 这里，我们有查询、键、值向量 V, K, Q，然后我们有一个之前提过的经过缩放的点积自注意力层。现在我们所做的就是将这个过程重复多次，这个过程可以并行化。然后，我们将得到的结果进行连接。这意味着我们将允许模型捕获这些单词对之间的不同类别的交互。具体公式如上所示，对于每一个头 head_i，我们进行一次注意力计算 $\mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$。因为有多个头，我们可以多次线性投影所有的查询、键、值向量，所以每一个头 head_i 都有一个自己的投影矩阵 W_i。将计算得到的所有 head_i 进行连接，然后再进行一次线性投影计算得到多头注意力 $\mathrm{MultiHead}(Q, K, V)$。

- Transformer block

- 可以看到，完整的 Transformer 块中还包含看一些其他操作。给定一个输入，我们先对其进行嵌入计算，然后计算查询、键、值向量，并在此基础上计算多头注意力的向量。然后，我们**将之前查询、键、值向量的结果再加入到多头注意力的结果中进行归一化处理**（normalization）。这意味着，我们从注意力网络中得到第一个单词的上下文表示 T_1 后，我们再回过头来将原单词的嵌入 E_1 加入到 T_1 的结果中，这种做法被称为**残差连接（residual connection），其理由是我们不希望模型忽略原单词的嵌入信息**。然后，我们将通过残差连接以及归一化得到的结果喂给一个前馈网络，并再次进行归一化操作。这就是在一个 Transformer 模型中所发生的全部事情。
- 在输入 Transformer 块之前的位置编码（Positional Encoding）又指的是什么呢？回忆一下，当我们计算注意力（attention）时，例如：我们要计算单词 “*made*” 的上下文表示时，我们每次分别独立地将注意力放在上下文单词 “*I*”、“*her*”、“*duck*” 上，**但是对于模型而言，它并不知道单词的位置信息**：即单词 “*her*” 是和目标单词 “*made*” 相邻的下一个单词，单词 “*duck*” 和目标单词 “*made*” 的距离要更远一些等等。所以，**位置编码实际上是一个向量，它编码了文本中单词的一些位置信息**，例如：单词 “*duck*” 出现在文本中的第 4 个位置，单词 “*her*” 出现在第 3 个位置等等。因此，位置编码为模型提供了一些词序相关的信息。

Conclusion

- Contextual representations are very useful
- Pre-trained on very large corpus
  - Learned some knowledge about language
  - Uses unsupervised objectives

- When we use them for downstream tasks, we are no longer starting from “scratch”

Lecture 12 Discourse

So far, most tasks/models we learned operate at word or sentence level. But NLP often deals with documents

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

Discourse segmentation

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

- 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*
    - $depth(gap_i) = (sim_{i-1} - sim_i) + (sim_{i+1} - sim_i)$
    - example:

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

$$depth(gap_i) = (sim_{i-1} - sim_i) + (sim_{i+1} - sim_i)$$

9

- supervised approaches
  - get labelled data from easy souces
    - scientific publications
    - Wikipedia articles

Abstract

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

1 Introduction

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

Separately, language models have long been a foundational component of any NLP task involving generation or textual normalisation of a noisy input (including speech, OCR and the processing of social media text). The primary purpose of a language model is to predict the probability of a span of text, traditionally at the sentence level, under the assumption that sentences are independent of one another, although recent work has started using broader local context such as the preceding sentences (Wang and Cho, 2016; Ji et al., 2016).

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

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

2 Related Work

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

- 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): 语篇标记在这里要更加重要一些，因为它们通常对于语篇分段前后的差异具有放大效应。

Discourse Parsing

- Identify discourse units(语篇单元), and the relations that hold between them
- Rhetorical(修辞) Structure Theory (RST) is a framework to do hierarchical analysis of discourse structure in documents. 在这个文档中，我们一共有 3 个语篇单元，RST 试图在给定这些语篇单元的情况下，发现它们之间所维系的关系。例如：第 2 个从句和第 3 个从句之间存在 让步（Concession）关系，而这两个语篇单元作为整体又和第一个主要句子之间存在 详述（Elaboration）关系。一旦我们构建完成这样一个层级树形结构，我们就可以知道根结点代表的核心语篇单元以及用于支持它的其他语篇单元。

- discourse units
  - Typically clauses(子句) of a sentence
  - DUs do not cross sentence boundary
  - *[It does have beautiful scenery,] [some of the best since Lord of the Rings.]*
  - 2 merged DUs = another composite DU

- discourse relations
  - Relations between discourse units

- 连接 (conjuction)，论证 (justify)，让步 (concession)，详述 (elaboration), etc.

[It does have beautiful scenery,]

↑ *(elaboration)*

[some of the best since Lord of the Rings.]

- nucleus(核心) vs. satellite(伴随体)
  - Within a discourse relation, one argument is the nucleus (the primary argument)
  - The supporting argument is the satellite

[It does have beautiful scenery,]$_{nucleus}$

↑ *(elaboration)*

[some of the best since Lord of the Rings.]$_{satellite}$

  - Some relations are equal (e.g. conjunction), and so both arguments are nuclei(nucleus复数)

[He was a likable chap,]$_{nucleus}$

↑ *(conjunction)*

[and I hated to see him die.]$_{nucleus}$

- RST tree
  - an RST relation conbines two or more DUs into composite DUs
  - process of combining DUs is repeated, creating an RST tree

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

17

- RST parsing(解析)
  - given a document, recover the RST tree
    - rule-based parsing
    - bottom-up approach
    - top-down approach
  - using discourse markers
    - Some discourse markers(语篇标记) (cue phrases(提示语)) explicitly indicate relations
      - although, but, for example, in other words, so, because, in conclusion,…
    - Can be used to build a simple rule-based parser
    - However
      - Many relations are not marked by discourse marker

        - DUs have some relation, but no explicit discourse markers

    - Many discourse markers ambiguous (e.g. and)
        - sometimes not a discourse marker: "*John and I went fishing*" (connect two nouns, not two DUs)
        - '*and*' can represent *conjunction, reason, justify*

- using machine learning
    - RST Discourse Treebank
        - 300+ documents annotated with RST trees
        - 创建这些文档的成本非常高并且非常耗时，语言学专家需要现将这些文档分成单个的语篇单元，然后缓慢地迭代合并这些语篇单元来构建一个能够描述整篇文档的完整 RST 树。(time-consuming to construct)

    - Basic idea:
        - Segment document into DUs
        - Combine adjacent DUs into composite DUs iteratively to create the full RST tree (bottom-up parsing)
        - 这里我们有一个非常简单的文档，它包含 4 个句子。首先，我们构建一个 EDU 分段器，将该文档划分为 4 个 EDUs。然后，我们再构建一个分类器，迭代地对这些 EDUs 进行合并。例如，我们可以自底向上，先对 *EDU-2*和 *EDU-3*进行合并，同时我们还需要预测出两者之间的关系*elab*。然后，我们按照同样的方式，将两者的复合 DU 与*EDU-1*进行合并，并给出新的关系。以此类推，我们可以通过自底向上的方式构建出完整的 RST 树。

EDU-1: Roy E. Parrott, the company's president and chief operating officer since Sept. 1, was named to its board.
EDU-2: The appointment increased the number of directors to 10,
EDU-3: three of whom are company employees.
EDU-4: Simpson is an auto parts maker.

    - bottom-up parsing
        - Transition-based parsing (lecture 16):
            - Greedy, uses shift-reduce algorithm (local optimum)
            - 基于转移的解析可能是语篇解析任务中最常用的方法。该方法是一种贪婪方法，类似于局部分类器，这里每个决策点之间都是独立的，它仅仅选择当前最可能的 EDU 对进行合并，而不会不考虑过去和未来的情况。

- CYK/chart parsing algorithm (lecture 14)
  - Global, but some constraints prevent CYK from finding globally optimal tree for discourse parsing
  - 该方法要比基于转移的解析方法慢得多，因此，通常在语篇解析任务中，主要采用的方法还是基于转移的解析。

- top-down parsing
  1. Segment document into DUs
  2. Decide a boundary to split into 2 segments
  3. For each segment, repeat step 2

2 → 1A: [It could have been a great movie.]
4 → 1B: [It does have beautiful scenery,]
3 → 1C: [some of the best since Lord of the Rings.]
5 → 1D: [The acting is well done,]
6 → 1E: [and I really liked the son of the leader of the Samurai.]
7 → 1F: [He was a likable chap,]
1 → 1G: [and I hated to see him die.]
1H: [But, other than all that, this movie is nothing more than hidden rip-offs.]

- features
  - bag of words

        - 单词对于语篇解析非常重要。

    - discourse markers
        - 语篇标记也非常重要，像 “however” 这类单词通常显式地表明了语篇单元之间的关系。

    - starting/ending n-grams
        - 通常，我们倾向于在句子开头和结尾使用一些特定的单词短语，它们也包含了一些句子之间逻辑上的转变。

    - location in the text
        - 文本中的句子之间的语篇关系转变有时和句子在文本中的位置有关。

    - syntax features
        - 通常，像词性（part-of-speech）等句法特征通常也包含了一些有用信息。

    - lexical and distributional similarities
        - 单词含义及上下文信息通常也非常有用。

- applications of discourse parsing

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

- Summarisation
    - 当语篇解析刚被提出的时候，主要是用于文本总结任务的：给定一段文本，我们希望找出这段文本想要表达的主要观点。假如我们知道了这边文档的 RST 树，我们就可以知道其

中哪些语篇单元是核心，哪些是伴随体，以及它们之间的关系，这样，我们就可以很容易地对文章内容进行总结归纳，因为我们知道作为核心的语篇单元（例如1B）的重要程度总是要比作为伴随体的语篇单元（例如1C）更高。

- Sentiment analysis
  - 语篇解析的另一个应用场景是情感分析。这里，我们来看一下 RST 树是如何决定这篇文档的情感倾向的。可以看到，我们用绿色标出了这段文本中所有表示正面情感的单词，然后用棕色标出了所有表示负面情感的单词。如果仅仅观察正负情感单词的出现频率，我们可以会认为这段影评表达的是一种正面情感，因为其中出现的大部分都是表示正面情感的单词。但是，如果我们观察这段文本的 RST 树，我们会发现位于根结点的最核心的语篇单元是1H，它对应文本中的最后一个句子，这表明我们在对这段文本进行情感分析时，最后一个句子表达的意思才是整段文本的重点。然后，我们会发现这段文本的情感倾向实际上是负面的。因此，语篇解析中的 RST 树可以帮助我们发现文本中的中心句。
- Argumentation mining
  - 语篇解析还可以帮助我们进行论据挖掘任务。RST 树可以帮助我们发现文本中用于支持一些观点的论据，例如：RST 树中的*Justify*关系。
- Authorship attribution
- Essay scoring

Anaphora Resolution(指代消解)

- anaphor(照应语)
  - linguistic expressions that refer back to earlier elements in the text
  - Anaphors have a antecedent(先行词) in the discourse, often but not always a noun phrase
    - Yesterday, Ted was late for work. It all started when his car wouldn't start.
      - '*it*' → '*Ted was late for work*'
      - '*his*' → '*Ted's*'
  - Pronouns(代词) are the most common anaphor
  - but there are various others
    - Demonstratives(指示代词) (that problem)
- motivations
  - Essential for deep semantic analysis
  - Very useful for QA, e.g., reading comprehension

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

Whose car is borrowed?

- antecedent restrictions
  - Pronouns must agree in **number** with their antecedents

        - *His coworkers were leaving for lunch when Ted arrived. They invited him, but he said no.*
        - 代词 “*They*” 是复数形式，因此，其所指代的对象必须是前面句子开头的 “*His coworkers*” 而非 “*Ted*”。

    - Pronouns must agree in **gender** with their antecedents

        - *Sue was leaving for lunch when Ted arrived. She invited him, but he said no.*

    - Pronouns whose antecedents are the subject of the same syntactic clause must be **reflexive**(反身的) (…self)

        - *Ted was angry at him. [him ≠ Ted]*
        - *Ted was angry at himself. [himself = Ted]*

- antecedent preferences
    - The antecedents of pronouns should be recent(距离近)
        - *He waited for another 20 minutes, but the tram didn't come. So he walked home and got his bike out of the garage. He started riding it to work.*
        - 上面的句子中，代词 “*it*” 存在歧义性，它可能指代前面的 “*the tram*”、“*his bike*” 或者 “*the garage*”。但是，结合上下文单词 “*riding*”，我们会发现这里的 “*it*” 指代的显然不可能是 “*the garage*”。然后对于剩下的 “*the tram*” 和 “*his bike*”，我们可以认为代词 “*it*” 更有可能指代的是 “*his bike*”，因为二者的距离更近一些。

    - The antecedent should be salient(突出的位置), as determined by grammatical position
        - Subject(主语) > object(宾语) > argument of preposition(介词主题)
        - *Ted usually rode to work with Bill. He was never late.*
        - 在上面的句子中，第二句中的 “*He*” 更有可能指代的是前一句的主语 “*Ted*” 而非宾语 “*Bill*”，因为在主语在一个句子中的重要程度要高于宾语。

- entities and reference

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

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

    - Discourse 16.1 (left) more coherent(连贯)
    - because pronouns all refer to John consistently, the protagonist(主人公)

- centering theory
    - A unified account of relationship between discourse structure and entity reference(语篇结构与实体参照之间关系的统一描述。)
        - RST 为我们描述了如何表示这些语篇结构，而中心理论要更进一步，它告诉我们如何将语篇结构中的这些实体和参照联系起来。并且，中心理论为我们提供了一种方式来解释为什么之前例子中的 16.1 的语篇要比 16.2 的语篇更具连贯性。

- Every utterance(表述) in the discourse is characterised by a set of **entities**, known as **centers**
- Explains preference of certain entities for ambiguous pronouns
- For an utterance U_n

(16.1) a. John went to his favorite music store to buy a piano.
b. He had frequented the store for many years.

  - **Forward-looking centers**:
    - All entities in Un: $C_f(U_n) = [e1, e2, \ldots]$
    - $C_f(16.1a) = [John, musicstore, piano]$
    - Ordered by syntactic prominence(句法突出性): subjects > objects
  - **Backward-looking center**:
    - **Highest ranked** forward-looking center in previous utterance $(C_f(U_{n-1}))$ that is also in current utterance (U_n)
    - Candidate entities in 16.1b = [John, music store]
    - $C_b(16.1b) = [John]$
    - Not *music store* because *John* has a higher rank in previous utterance's forward-looking centers $C_f(U_{n-1})$

- centering algorithm
  - When resolving entity for anaphora resolution, choose the entity such that the **top foward-looking center** matches with the **backward-looking center**
  - why? becausethe text reads more fluent when this condition is satisfied
  - example:

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

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

Text is coherent because the top forward-looking center matches the backward-looking center for each utterance:

top forward-looking center = *John*
backward-looking center = *John*

Not quite the case here.

C_f(16.2b) = [*music store*, *John*]
C_b(16.2b) = [*John*]

C_f(16.2d) = [*music store*, *John*]
C_b(16.2d) = [*John*]

- 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
  - also easy to include features that are potentially helpful
    - words around anaphor/antecedent

Conlucsion

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

Lecture 13 Formal Language Theory & Finite State Automata

We learn n-gram language model, hidden Markov model, RNNs, but none of these is fundamentally linguistic.(没有涉及到语言的本质)

Formal Language Theory

- Studies **classes of languages** and their computational properties
  - Regular language (this lecture)

- Context free language (next lecture)

- A language = set of strings
- A string = sequence of elements from a finite alphabet (AKA vocabulary)

- Goal: solve the **membership problem** by defining its **grammar** → whether a string is in a language

- some examples of language
  - Binary strings that start with 0 and end with 1
    - { 01, 001, 011, 0001, … } ✓
    - { 1, 0, 00, 11, 100, … } ✗
  - Even-length sequences from alphabet {a, b}
    - { aa, ab, ba, bb, aaaa, … } ✓
    - { aaa, aba, bbb, … } ✗
  - English sentences that start with wh-word and end in ?
    - { what ?, where my pants ?, … } ✓
    - { hello how are you?, why is the dog so cute! } ✗
- beyond membership problem
  - Membership
    - Is the string part of the language? Y/N
  - Scoring
    - Graded membership
    - “How acceptable is a string?” (language models!)
  - Transduction(转导,转换)
    - “Translate” one string into another (stemming!)

Regular Language

- The simplest class of languages
- Any regular expression is a regular language
  - Describes what strings are part of the language (e.g. ‘0(0|1)*1’)

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

- examlpes of regular languages
  - Binary strings that start with 0 and end with 1
    - $0(0|1)^*1$

  - Even-length sequences from alphabet {a, b}
    - $((aa)|(bb)|(ab)|(ba))^*$

  - English sentences that start with wh-word and end in ?
    - $((what)|(where)|(why)|(which)|(whose)|(whom))\Sigma^*?$

- properties of regular languages
  - **Closure**: if we take regular languages L1 and L2 and merge them, is the resulting language regular? if it is, then this is a closed operation(封闭运算)
  - RLs are closed under the following:
    - **concatenation and union**
    - **intersection**: strings that are valid in both L1 and L2
    - **negation**: strings that are not in L

  - Extremely versatile(通用，全能)! Can have RLs for different properties of language(不同的正则语言来描述语言不同的性质), and use them together

Finite State Acceptor

Regualr expression defines a regular language, but does not give an algorithm to check whether a string belongs to the language. $Finitestateacceptors(FSA)$ describes the computation involved for membership checking.

- FSA consists
  - alphabet of input symbols: Σ
  - set of state: Q
  - start state: $q_0 \in Q$
  - final states: $F \subseteq Q$
  - transition function: $(q_i, symbol) \rightarrow q_j$
- example

  - regular expression defined by this FSA: $a^* bb^*$
- derivational morphology
  - Use of affixes to change word to another grammatical category (* means invalid)
    - $grace \rightarrow graceful \rightarrow gracefully$
    - $grace \rightarrow disgrace \rightarrow disgracefully$
    - $allure \rightarrow alluring \rightarrow alluringly$
    - $allure \rightarrow^* allureful$
    - $allure \rightarrow^* disallure$

- FSA for morphology
  - fairly consisten process, and we want to:
    - accept valid forms ($grace \rightarrow graceful$)
    - reject invalid ones ($allure$\righarrow$^{*}allureful$)
    - generalise to other words e.g. nouns behave like *grace* or *allure*

    - collapse or merge states to make FSA to have less states

- weighted FSA
  - Some words are more plausible(似乎合理的) than others
    - *fishful* vs. *disgracelyful*, *fishful* more plausible
    - *musicky* vs. *writey*, *musicky* more plausible

  - Graded measure of acceptability — weighted FSA changes the following:
    - start state weight function: $\lambda : Q \rightarrow \mathbb{R}$
    - final state weight function: $\rho : Q \rightarrow \mathbb{R}$
    - transition function: $\delta : (Q, \Sigma, Q) \rightarrow \mathbb{R}$

  - WFSA shortest-path
    - total score of a path $\pi = t_1, \ldots, t_N$

      - $\lambda(t_0) + \sum_{i=1}^{N} \delta(t_i) + \rho(t_N)$, where t is an edge

    - use **shortest-path algorithm** to find π with minimum cost

      - $O(VlogV + E)$

Finite State Transducer

Often don't want to just accept or score strings, want to translate them into another string

Finite state **acceptor**: allure + ing = allureing
Finite state **transducer**: allure + ing = alluring

- FST add **string output capability**(字符串输出能力) to FSA
  - includes an **output alphabet**
  - transitions now take input symbol and **emit(发出) output symbol** (Q, Σ, Σ, Q)
- Can be weighted = WFST
  - Graded scores for transition
- E.g., edit distance(编辑距离) as WFST
  - distance to transform one string to another
- edit distance automata

- FST for inflectional morphorlogy
  - Verb inflections in Spanish must match the subject in person & number
  - Goal of morphological analysis:
    - canto → cantar+VERB+present+1P+singular

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

canto →

- *canto*+Noun+Masc+Sing
- *cantar*+Verb+PresInd+1P+Sing

canta → *cantar*+VERB+PresInd+3P+Sing

- is natural language regualr? → no! only sometimes:

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

  - Length is unbounded, but structure is local
    - (Det Noun Prep Verb)*
    - Can describe with FSA

- non-regular languages
  - Arithmetic expressions with balanced parentheses
    - $(a + (b \times (c/d)))$
    - Can have arbitrarily many opening parentheses
    - Need to remember how many open parentheses, to produce the same number of closed parentheses
    - Can't be done with finite number of states
  - $a^n b^n$

- center embedding
  - 在自然语言中，还有一种被称为 中心嵌入（Center Embedding）的现象：相关的从句是以中心方式嵌入的
  - Center embedding of relative clauses

The cat loves Mozart

The cat the dog chased loves Mozart

The cat the dog the rat bit chased loves Mozart

The cat the dog the rat the elephant admired bit chased loves Mozart

- Need to remember the *n* subject nouns, to ensure n verbs follow (and that they agree etc)
- cannot be captured by regular expressions!($S^n V^n$)
- **Requires (at least) context-free grammar**

Conclusion

- Concept of a language
- Regular languages
- Finite state automata: acceptors, transducers
- Weighted variants
- Application to edit distance, morphology

Lecture 14 Context-Free Grammar

- basics of context-free grammars
  - **Symbols**
    - **Terminal**: word such as book
    - **Non-terminal**: syntactic label(句法标签) such as NP or VP
    - convention: lowercase for terminals, uppercase for non-terminals
  - **Productions** (rules)
    - W → X Y Z
    - **Exactly one non-terminal** on left-hand side (LHS)
    - An **ordered** list of symbols on right-hand side (RHS); can be **terminals or non-terminals**
  - Start symbol: **S**
- why is it context-free
  - Production rule depends only on the LHS (and not on ancestors, neighbours)
    - Analogous(相似) to Markov chain
    - Behaviour at each step depends only on current state
- context-free vs. regular
  - context-free more genral and allows recursive(non-terminals can produce themselves) nesting

- CFG parsing
  - given production rules
    - $S \rightarrow aSb$
    - $S \rightarrow ab$
  - and a string
    - $aaabbb$
  - produce a valid parse tree

- if English can be represented with CFG
  - first develop the production rules
  - then build a 'parser' to automatically judge whether a sentence is grammatical
- but natural language is not quite context-free: cross-serial dependencies ($a^m b^n c^m d^n$)

  Swiss-German:

  ...de Karl d'Maria em Peter de Hans laat hälfe lärne schwüme

  English:

  ...Charles lets Mary help Peter to teach John to Swim
- however, CFG strike a good balance
  - covers most syntactic patterns
  - CFG parsing is computational efficient
- can use CFG to describe a core fragment of English syntax

Constituents

- syntactic constituents
  - Sentences are broken into constituents
    - word sequence that function as a **coherent unit** for linguistic analysis

        - helps build CFG production rules

    - Constituents have certain key properties:
        - movement
            - Constituents can be moved around sentences
                - *Abigail gave [her brother] [a fish]*
                - *Abigail gave [a fish] to [her brother]*

            - Contrast: [gave her], [brother a]

        - substitution
            - Constituents can be substituted by other phrases of the same type
                - *Max thanked [his older sister]*
                - *Max thanked [her]*

            - Contrast: [Max thanked], [thanked his]

        - coordination
            - Constituents can be conjoined(结合，连接) with coordinators like and and or
                - [Abigail] and [her young brother] brought a fish
                - Abigail [bought a fish] and [gave it to Max]
                - Abigail [bought] and [greedily ate] a fish

- phrase
    - Once we identify constituents, we use **phrases** to describe them
    - Phrases are determined by their **head word**:

        - noun phrase: her younger **brother**
        - verb phrase: greedily **ate** it

    - can use CFG to formalise these intuitions

- given *'He gave a lecture and away a pie'*, which of the following is a constituent

    - *a lecture* (yes, check movement: '*he gave away a pie and a lecture*')
    - *gave a lecture* (no, chek movement: '*he away a pie and give a lecture*')
    - *a pie* (no, check movement: '*he gave a pie and away a lecture*')
    - *away a pie* (yes, check movement: '*he gave away a pie and a lecture*')

- a simple CFG for English
    - **Terminal symbols**: rat, the, ate, cheese
    - **Non-terminal symbols**: S, NP, VP, DT, VBD, NN
    - **Productions**:

        - S → NP VP
        - NP → DT NN
        - VP → VBD NP
        - DT → the
        - NN → rat

        - NN → cheese
        - VBD → ate

    - generating sentence with the CFG
        - Always start with S (the sentence/start symbol)
            - S

        - Apply a rule with S on LHS (S → NP VP), i.e substitute RHS
            - NP VP

        - Apply a rule with NP on LHS (NP → DT NN)
            - DT NN VP

        - Apply rule with DT on LHS (DT → the)
            - the NN VP

        - Apply rule with NN on LHS (NN → rat)
            - the rat VP

        - Apply rule with VP on LHS (VP → VBD NP)
            - the rat VBD NP

        - Apply rule with VBD on LHS (VBD → ate)
            - the rat ate NP

        - Apply rule with NP on LHS (NP → DT NN)
            - the rat ate DT NN

        - Apply rule with DT on LHS (DT → the)
            - the rat ate the NN

        - Apply rule with NN on LHS (NN → cheese)
            - the rat ate the cheese ← no non-terminals left! done!

- CFG tree
    - Generation corresponds to a syntactic tree
    - Non-terminals are internal nodes
    - Terminals are leaves
    - CFG parsing is the **reverse** process (sentence → tree)

- a CFG for arithmetic expressions

$$
\begin{aligned}
\text{S} &\rightarrow \text{S Op S} \mid \text{Num} \\
\text{Op} &\rightarrow + \mid - \mid \times \mid \div \\
\text{Num} &\rightarrow \text{Num Digit} \mid \text{Digit} \\
\text{Digit} &\rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9
\end{aligned}
$$

  - S = starting symbol
  - I = operator OR
  - recursive, NUM and S can priduce themselves
  - parsing example: '1+2-3'

CYK Algorithm

- use bottom-up parsing to test whether a string is valid given a CFG, without enumerating all possible parses.
- Core idea: form small constituents first, and merge them into larger constituents
- Requirement: CFGs must be in **Chomsky Normal Forms**
- Chomsky Normal Form
  - all rules of form:
    - A → BC
    - A → a
  - convert rules of form A → Bc into:
    - A → BX
    - X → c
  - Convert rules A → B C D into:
    - A → B Y
    - Y → C D
    - E.g. VP → VP NP NP for ditransitive(双宾语) cases, “sold [her] [the book]”
  - X, Y are new symbols we have introduced
  - 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
      - A → B, B → cat, B → dog
      - A → cat, A → dog
- CYK parsing algorithm

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

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

fill the diagonals (NP → we)

bottom to top

going through the 'splits'

create the links if they are in the production rules

Figure 12.5 The CKY algorithm.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- however, two different results(parse ambiguity). which one is better?
  - the second one is better → '*eat sushi*' and '*with chopsticks*' more sensible
  - the first one → '*eat*' and '*sushi with chopsticks*' not sensible

Representing English with CFGs

For toy grammars (like the example above), we only need handful of production rules good for demonstration or extremely limited domains. However, for real texts, real grammars are needed. Many thousands of production rules!

- key constituents in Penn Treebank
  - Sentence (S)
  - Noun phrase (NP)
  - Verb phrase (VP)
  - Prepositional phrase (PP)
  - Adjective phrase (AdjP)
  - Adverbial phrase (AdvP)
  - Subordinate clause (SBAR)

- PTB/0001 example
- basic English sentence structures
  - Declarative sentences(陈述句) (S → NP VP)
    - The rat ate the cheese

  - Imperative sentences(祈使句，命令句) (S → VP)
    - Eat the cheese!

    - Yes/no questions (S → VB NP VP)
        - Did the rat eat the cheese?
    - Wh-subject-questions (S → WH VP)
        - Who ate the cheese?
    - Wh-object-questions (S → WH VB NP VP)
        - What did the rat eat?
- English noun phrases
    - Pre-modifiers
        - DT, CD, ADJP, NNP, NN
        - E.g. *the two very best Philly cheese steaks*
    - Post-modifiers
        - PP, VP, SBAR
        - *A delivery from Bob coming today that I don't want to miss*
    - NP → DT? CD? ADJP? (**NNINNP**)+ PP* VP? SBAR?
- English verb phrases
    - Auxiliaries
        - MD, AdvP, VB, TO
        - E.g *should really have tried to wait*
        - VP → (MDIVBITO) AdvP? **VP**
    - Arguments and adjuncts
        - NP, PP, SBAR, VP, AdvP
        - E.g *told him yesterday that I was ready*
        - E.g. *gave John a gift for his birthday to make amends*
        - VP → VB NP? NP? PP* AdvP* VP? SBAR?
- other constituents

Prepositional phrase

- PP → IN NP — *in the house*

Adjective phrase

- AdjP → (AdvP) JJ — *really nice*

Adverb phrase

- AdvP → (AdvP) RB — *not too well*

Subordinate clause

- SBAR → (IN) S — *since I came here*

Coordination

- NP → NP CC NP; VP → VP CC VP; etc. — *Jack and Jill*

Complex sentences

- S → S SBAR; S → SBAR S; etc. — *if he goes, I'll go*

Conclusion

- Context-free grammars can represent most linguistic structures of natural languages
- There are relatively fast dynamic programming algorithms (CYK) to retrieve this structure

Lecture 15 Probabilisitc Context-Free Grammar

use probabilistic CFG(PCFG) to solve parse ambiguity

Basics of PCFGs

- Same symbol set:
  - Terminals: words such as book
  - Non-terminal: syntactic labels such as NP or NN
- Same productions (rules)
  - LHS non-terminal → ordered list of RHS symbols
- In addition, store a probability with each production
  - NP → DT NN [p = 0.45]
  - NN → cat [p = 0.02]
  - NN → leprechaun [p = 0.00001]
  - one way to generate probability: use CFGs to parse training data, probabilities based on frequencies of production rules
- Probability values denote **conditional**

    - P(LHS → RHS)
    - P(RHS | LHS)

- Consequently they:
    - must be positive values, between 0 and 1
    - must sum to one for given LHS

- E.g.,
    - NN → aadvark [p = 0.0003]
    - NN → cat [p = 0.02]
    - NN → leprechaun [p = 0.0001]
    - $\sum_x P(NN \rightarrow x) = 1$

- stochastic generation with PCFGs

    - Almost the same as for CFG, with one twist:

    1. Start with S, the sentence symbol
    2. Choose a rule with S as the LHS
        - **Randomly** select a RHS **according to P(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

    - Gives us a tree, as before, with a sentence as the yield

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

- how likely is a tree
  - Given a tree, we can compute its probability
    - Decomposes into probability of each production
  - example

P(tree) =

P(S → VP) ×
P(VP → Verb NP) ×
P(Verb → *Book*) ×
P(NP → Det Nominal) ×
P(Det → *the*) ×
P(Nominal → Nominal Noun) ×
P(Nominal → Noun) ×
P(Noun → *dinner*) ×
P(Noun → *flight*)

P(tree)
= P(S → VP) × P(VP → Verb NP) × P(Verb → Book) ×
P(NP → Det Nominal) × P(Det → the) × P(Nominal → Nominal Noun) ×
P(Nominal → Noun) × P(Noun → dinner) × P(Noun → flight)
= 0.05 × 0.20 × 0.30 ×
0.20 × 0.60 × 0.20 ×
0.75 × 0.10 × 0.40
= 2.2 × 10-6

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

- resolving parse ambiguity
  - select between different trees based on P(T)
  - the left tree with higher probability is a better tree

PCFG Parsing

CYK for unweighted grammars(CFGs) finds **all possible trees**, but there are often 1000s, many competely nonsensical. how to solve for **most probable tree**?

- CYK for PCFGs
  - CYK finds all trees for a sentence; we want best tree
  - Prob. CYK follows similar process to standard CYK
  - Convert grammar to Chomsky Normal Form (CNF)
    - VP → Verb NP NP [0.10]

- VP → Verb NP+NP [0.10] ⇒ NP+NP → NP NP [1.0]
- where NP+NP is a new symbol

- example:

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

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

	we	**eat**	**sushi**	**with**	**chopsticks**
	NP 1/4 [0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		V 1 [1,2]	[1,3]	[1,4]	[1,5]
			NP 1/8 [2,3]	[2,4]	[2,5]
				IN 1 [3,4]	[3,5]
					NP 1/2 [4,5]

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

	we	eat	sushi	with	chopsticks
	NP 1/4 [0,1]	Ø [0,2]	[0,3]	[0,4]	[0,5]
		V 1 [1,2]	[1,3]	[1,4]	[1,5]
			NP 1/8 [2,3]	[2,4]	[2,5]
				IN 1 [3,4]	[3,5]
					NP 1/2 [4,5]

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

	we	eat	sushi	with	chopsticks
	NP 1/4 [0,1]	Ø [0,2]	[0,3]	[0,4]	[0,5]
		V 1 [1,2]	**VP** 1/16 (1/2 * 1 * 1/8) [1,3]	[1,4]	[1,5]
			NP 1/8 [2,3]	[2,4]	[2,5]
				IN 1 [3,4]	[3,5]
					NP 1/2 [4,5]

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

we	eat	sushi	with	chopsticks
NP 1/4 [0,1]	Ø [0,2]	**S** 1/64 (1 * 1/4 * 1/16) [0,3]	[0,4]	[0,5]
	V 1 [1,2]	**VP** 1/16 [1,3]	[1,4]	[1,5]
		NP 1/8 [2,3]	[2,4]	[2,5]
			IN 1 [3,4]	[3,5]
				NP 1/2 [4,5]

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

we	eat	sushi	with	chopsticks
NP 1/4 [0,1]	Ø [0,2]	**S** 1/64 [0,3]	Ø [0,4]	[0,5]
	V 1 [1,2]	**VP** 1/16 [1,3]	Ø [1,4]	[1,5]
		NP 1/8 [2,3]	Ø [2,4]	[2,5]
			IN 1 [3,4]	[3,5]
				NP 1/2 [4,5]

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

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

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

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

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

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

S	→ NP VP	1
NP	→ NP PP	⅛
	→ we	¼
	→ sushi	⅛
	→ chopsticks	½
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

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

S	→ NP VP	1
NP	→ NP PP	⅛
	→ we	¼
	→ sushi	⅛
	→ chopsticks	½
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

S	→	NP VP	1
NP	→	NP PP	⅛
	→	we	¼
	→	sushi	⅛
	→	chopsticks	½
PP	→	IN NP	1
IN	→	with	1
VP	→	V NP	½
	→	VP PP	¼
	→	MD V	¼
V	→	eat	1

- 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(Chomsky Normal Form) by removing new non-terminals

- prob. CYK vs. CYK

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

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

CYK

Both CYK and prob. CYK store all possible NTs

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

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                  and its probability
  for j ← from 1 to LENGTH(words) do
      for all { A | A → words[j] ∈ grammar}
          table[j − 1, j, A] ← P(A → words[j])
      for i ← from j − 2 downto 0 do
          for k ← i + 1 to j − 1 do
              for all { A | A → BC ∈ grammar,
                          and table[i,k,B] > 0 and table[k, j,C] > 0 }
                  if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                      table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                      back[i,j,A] ← {k,B,C}
      return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Probabilistic CYK

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

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

Time complexity in terms of sentence length n?

  - Both CYK and prob. CYK store all possible NTs
  - validity test now looks to see that the child chart cells have non-zero probability
  - Instead of storing set of symbols, store the probability of best scoring tree fragment covering span [i,j] with root symbol A. Overwrite lower scoring analysis if this one is better, and record the best production
  - time complexity in terms of sentence length n: $O(n^3)$ (note: checks with probabilities of multiple productions of LHS is only $O(1)$)

Limitations of CFG

poor independence assumptions

- Rewrite decisions made independently, whereas inter-dependence is often needed to capture global structure.
  - NP → DT NN [0.28] (the cat)
  - NP → PRP [0.25] (pronoun)
  - Probability of a rule independent of rest of tree
  - No way to represent this contextual differences in PCFG probabilities

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

- NP → PRP should go up to 0.91 as a subject
- NP → DT NN should be 0.66 as an object
- Solution: add a condition to denote whether NP is a subject or object
- solution: parent conditioning

  - Make non-terminals more explicit by incorporating parent symbol into each symbol
  - NP^S represents subject position (left)
  - NP^VP denotes object position (right)
  - now change probabilities of production rules

Lack of Lexical Conditioning

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

"into a bin" describes the resulting location of the sacks

sacks to be dumped are the ones which are "into a bin"

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

- solution: head lexicalisation
  - Record head word with parent symbols
    - the most salient child of a constituent, usually the noun in a NP, verb in a VP etc
    - VP → VBD NP PP ⇒ VP(dumped) → VBD(dumped) NP(sacks) PP(into)

  - Incorporate head words into productions, to capture the most important links between words
    - Captures correlations between head words of phrases
    - PP(into): VP(dumped) vs. NP(sacks)

  - problem: grammar symbol inventory **expands massively**!

    - Many of the productions too specific, rarely seen
    - Learning more involved to avoid sparsity problems (e.g., zero probabilities)

Conclusion

- 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!

Lecture 16 Dependency Grammar

CFGs assume a constituency tree which identifies the **phrases** in a sentence, these phrases are interchangable and maintain grammaticality.

Dependency Grammars

- Dependency grammar offers a **simpler** approach

    - describe **relations** between pairs of words
    - namely, between **heads** and **dependents** (directed arc!)
    - e.g. (*prefer*, *dobj*, *flight*)

- why dependency grammars
    - 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 (very rigid)

    - Head-dependent relations similar to semantic relations between words
        - More useful for applications: coreference resolution, information extraction, etc

Basics of Dependency Grammar

- dependency relations
    - Captures the grammatical relation between two words:
        - **Head** = central word
        - **Dependent** = supporting word

    - Grammatical relation = subject, direct object, etc
    - Many dependency theories and taxonomies(分类学，分类法) proposed for different

languages(different languages have different syntatic relations)

- **Universal Dependency**: a framework to create a set of dependency relations that are computationally useful and **cross-lingual**(跨语言的)(but a bit coarse-grained)

  - example

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

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

  - missing preposition relations, because it is specific to English. expressed in CASE relation.

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

- information extraction
  - "Brasilia, the Brazilian capital, was founded in 1960."
    - → capital(Brazil, Brasilia)
    - → founded(Brasilia, 1960)
  - Dependency tree captures relations succinctly

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

- dependency vs. constituency
  - Dependency tree

        - each node is a word token
        - one node is chosen as the root(typically the main verb in the sentence)
        - 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

    - Both use part-of-speech

- 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 should be **projective**

- projectivity
    - An arc is projective if there is a path from head to every word that lies between the head and the dependent

    - Dependency tree is projective if all arcs are projective
    - In other words, a dependency tree is projective if it can be drawn with **no crossing edges**

        - arc(*flight,nmod,was*) is not projective! (no path from *flight* to *this, morning*)

- Most sentences are projective, but **exceptions exist**
  - Common in languages with flexible word order

- Treebank conversion (convert CFG trees to dependency trees)
  - A few dependency treebanks (Czech, Arabic, Danish…)
  - Many constituency treebanks
  - Some can be converted into dependencies
  - Dependency trees generated from constituency trees are **always projective**
  - Main idea: identify head-dependent relations in constituency structure and the corresponding dependency relations
    - Use various heuristics, e.g., head-finding rules
    - Often with manual correction

- examples from treebanks
  - Danish DDT includes additional 'subject' link for verbs

http://www.buch-kromann.dk/matthias/ddt1.0/

  - METU-Sabancı Turkish treebank
    - edges between morphological units, not just words

Oflazer, Kemal, et al. "Building a Turkish treebank." *Treebanks*. Springer, 2003. 261-277.

Transition-based Parsing

- dependency parsing
  - Find the best structure for a given input sentence
  - Two main approaches:
    - **Transition-based**: bottom-up greedy method (not global optimal,but dominant approach for quick computation!)
    - **Graph-based**: encodes problem using nodes/edges and use graph theory methods to find optimal solutions(global optimal)

- caveat(警告，说明)
  - Transition-based parsers can only handle **projective** dependency trees!
  - Less applicable for languages where crossdependencies are common

- transition-based parsing: intuition
  - Processes word from left to right
  - Maintain two data structures
  - **Buffer**: input words yet to be processed
  - **Stack**: store words that are being processed

  - At each step, perform one of the 3 actions:

- **Shift**: move a word from buffer to stack
- **Left-Arc**: assign current word as head of the previous word in stack

- **Right-Arc**: assign previous word as head of current word in stack

- example:

Step	Stack	Buffer	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

- dependency labels
  - For simplicity, we omit labels on the dependency relations
  - In practice, we parameterise the left-arc and rightarc actions with dependency labels(a larger set of actions):
    - E.g. left-arc-nsubj or right-arc-dobj
  - Expands the list of actions to > 3 types
- the right action?
  - We assume an **oracle** that tells us the right action at every step
  - Given a dependency tree, the role of oracle is to generate a sequence of ground truth actions

Action
SHIFT
SHIFT
RIGHTARC
SHIFT
SHIFT
SHIFT
LEFTARC
LEFTARC
RIGHTARC
RIGHTARC
Done

- 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 (as 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)
  - example

Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled → United) (flights → morning) (flights → the)

- Input features:
  - $s_1.w$ = *flights*
  - $s_2.w$ = *cancelled*
  - $s_1.t$ = *NNS*
  - $s_2.t$ = *VBD*
  - $b_1.w$ = *to*
  - $b_1.t$ = *TO*
  - $s_1.t \circ s_2.t$ = *NNS_VBD*

Source	Feature templates		
One word	$s_1.w$	$s_1.t$	$s_1.wt$
	$s_2.w$	$s_2.t$	$s_2.wt$
	$b_1.w$	$b_1.w$	$b_0.wt$
Two word	$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
	$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
	$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

- Output label: *shift*

- classifiers
  - Traditionally SVM works best
  - Nowadays, deep learning models are state-of-the-art
  - Weakness: local classifier based on **greedy search** (not always give global optimal solution)
  - Solutions:
    - Beam search: keep track of top-N best actions
    - Dynamic oracle: during training, use predicted actions occasionally
    - Graph-based parser

Graph-based Parsing

- 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 (largest weight sum, Kruskal's algorithm)

- advantage
  - Can produce **non-projective** trees
    - Not a big deal for English(don't tend to see non-projective trees in English)
    - But important for many other languages
  - Score entire trees
    - Avoid making greedy local decisions like transition-based parsers

        - Captures long dependencies better

- example
    - Caveat: tree may contain cycles
    - Solution: need to do cleanup to remove cycles (Chu-Liu-Edmonds algorithm)

    - maximum spanning tree highlighted in blue

Conclusion

- Dependency parsing a compelling, alterative, formulation to constituency parsing
    - Edges encode word-word syntactic and semantic relations

- Transition-based parsing
- Graph-based parsing

Lecture 17 Machine Translation

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

虽然北风呼啸，但天空依然十分清澈

↓

However, the sky remained clear under the strong north wind

- why?
    - Removes language barrier
    - Makes information in any languages accessible to anyone
    - But translation is a classic “AI-hard” challenge
    - Difficult to preserve the **meaning and the fluency** of the text after translation

- MT is difficult
  - Not just simple word for word translation
  - Structural changes, e.g., syntax and semantics
  - Multiple word translations, idioms(习语，成语，方言)
  - Inflections for gender, case etc
  - Missing information (e.g., determiners)

虽然	北	风	呼啸	，	但	天空	依然	十分	清澈
although	*north*	*wind*	*howls*	*,*	*but*	*sky*	*still*	*extremely*	*limpid*

Statistical MT

- earliy MT
  - Started in early 1950s
  - Motivated by the Cold War to translate Russian to English
  - Rule-based system
    - Use bilingual dictionary to map Russian words to English words
  - Goal: translate 1-2 million words an hour within 5 years
- statistical MT
  - Given French sentence *f*, aim is to find the best English sentence *e*
    - $argmax_e P(e|f)$
  - Use Baye's rule to decompose into two components
    - $argmax_e P(f|e)P(e)$
  - language vs translation model
    - $argmax_e P(f|e)P(e)$
    - $P(e)$: **language model**
      - learn how to write fluent English text
    - $P(f|e)$: **translation model**
      - learns how to translate words and phrases from English to French
  - how to learn LM and TM
    - Language model:
      - Text statistics in large **monolingual(仅一种语言的) corpora** (n-gram models)
    - Translation model:
      - Word co-occurrences in **parallel corpora**
      - i.e. English-French sentence pairs
  - parallel corpora
    - One text in multiple languages
    - Produced by human translation

    - Bible, news articles, legal transcripts, literature, subtitles
    - Open parallel corpus: http://opus.nlpl.eu/

- models of translation
  - how to learn $P(f|e)$ from paralell text?
  - We only have sentence pairs; **words are not aligned** in the parallel text
  - I.e. we don't have word to word translation

- alignment
  - Idea: introduce **word alignment** as a latent variable into the model
    - $P(f, a|e)$
  - Use algorithms such as expectation maximisation (EM) to learn (e.g. GIZA++)

  - complexity
    - some words are dropped and have no alignment

    - One-to-many alignment

- many-to-one alignment

- many-to-many alignment

- summary
  - A very popular field of research in NLP prior to 2010s
  - Lots of feature engineering
  - State-of-the-art systems are very complex
    - Difficult to maintain
    - Significant effort needed for new language pairs

Neural MT

- introduction

- Neural machine translation is a new approach to do machine translation
- Use a single neural model to directly translate from source to target
- from model perspective, a lot simpler
- from achitecture perspective, easier to maintain
- Requires parallel text
- Architecture: encoder-decoder model
  - 1st RNN to encode the source sentence
  - 2nd RNN to decode the target sentence

$$h_i = RNN_1(h_{i-1}, x_i)$$
$$s_t = RNN_2(s_{t-1}, y_t)$$
$$s_1 = h_{|x|}$$

22

- neural MT
  - The decoder RNN can be interpreted as a **conditional language model**
    - Language model: predicts the next word given previous words in target sentence y
    - Conditional: prediction is also conditioned on the source sentence *x*
  - $P(y|x) = P(y_1|x)P(y_2|y_1, x)...P(y_t|y_1, ..., y_{t-1}, x)$
  - training
    - Requires parallel corpus just like statistical MT
    - Trains with next word prediction, just like a language model
    - loss

- 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**: take the word with the highest probability at every step
- exposure bias
  - Describes the discrepancy(差异) between training and testing
  - Training: always have the ground truth tokens at each step
  - Test: uses its own prediction at each step
  - Outcome: model is unable to recover from its own error(error propagation)

- greedy decoding
  - argmax decoding is also called greedy decoding

- Issue: does not guarantee optimal probability $P(y|x)$

- exhaustive search decoding
  - To find optimal $P(y|x)$, we need to consider every word at every step to compute the probability of all possible sequences
  - $O(V^n)$ where V = vocab size; n = sentence length
  - Far too expensive to be feasible

- beam search decoding
  - Instead of considering all possible words at every step, consider k best words
  - That is, we keep track of the top-k words that produce the best partial translations (hypotheses) thus far
  - k = beam width (typically 5 to 10)
  - k = 1 = greedy decoding
  - k = V = exhaustive search decoding
  - example:

Beam search decoding with k=2

Beam search decoding with k=2

Beam search decoding with k=2

Beam search decoding with k=2

Beam search decoding with k=2

Beam search decoding with k=2

Beam search decoding with k=2

- when to stop
  - When decoding, we stop when we generate 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)

- issues of NMT
  - Information of the whole source sentence is represented by a single vector
  - NMT can generate new details not in source sentence
  - NMT tend to generate not very fluent sentences (×, usually fluent, a strength)
  - Black-box model; difficult to explain when it doesn't work

- summary
  - Single end-to-end model
    - Statistical MT systems have multiple subcomponents
    - Less feature engineering
    - Can produce new details that are not in the source sentence (hallucination:错觉，幻觉)

Attention Mechanism

- With a long source sentence, the encoded vector is unlikely to capture all the information in the sentence
- This creates an **information bottleneck**(cannot capture all information in a long sentence in a single short vector)
- attention
    - For the decoder, at every time step allow it to **'attend'** to words in the source sentence

cow
weighted sum
attention
distribution
first time step attends
to “牛” to predict cow
softmax
dot product
RNN1
RNN1
RNN1
RNN2
牛
吃
草
<start>

cow
eats
weighted sum
attention distribution
second time step attends to "吃" to predict eats
softmax
dot product
RNN1
RNN1
RNN1
RNN2
RNN2
牛
吃
草
<start>
cow

- encoder-decoder with attention

- Encoder hidden states: $h_i = RNN_1(h_{i-1}, x_i)$
- Decoder hidden states: $s_t = RNN_2(s_{t-1}, y_t)$
- For each time step in the decoder, attend to each of the hidden states to produce the attention weights:
  - $e_t = [s_t^{\mathrm{T}} h_1, s_t^{\mathrm{T}} h_2, \ldots, s_t^{\mathrm{T}} h_{|x|}]$
- Apply softmax to the attention weights to get a valid probability distribution:
  - $\alpha_t = \text{softmax}(e_t)$
- Compute weighted sum of the encoder hidden states:
  - $c_t = \sum_{i=1}^{|x|} \alpha_t^i h_i$
- Concatenate c_t and s_t to predict the next word

context vector

- variants
  - attention
    - dot product: $s_t^T h_i$
    - bilinear: $s_t^T W h_i$
    - additive: $v^{T}\tanh(W_s s_t + W_h h_i)$
  - c_t can be injected to the current state (s_t), or to the input word (y_t)
- summary
  - Solves the information bottleneck issue by allowing decoder to have access to the source sentence words directly(reduce hallucination a bit, direct access to source words, less likely to generate new words not related to source sentence)
  - Provides some form of interpretability (look at attention distribution to see what source word is attended to)
    - Attention weights can be seen as word alignments
  - Most state-of-the-art NMT systems use attention
    - Google Translate (https://slator.com/technology/google-facebook-amazonneural-machine-translation-just-had-its-busiest-month-ever/)

Evaluation

- MT evaluation
  - BLEU: compute n-gram overlap between “reference” translation(ground truth) and generated translation
  - Typically computed for 1 to 4-gram
    - $BLEU = BP \times exp(\frac{1}{N} \sum_n^N logp_n)$, where BP → "Brevity Penalty" to penalise short outputs
    - $p_n = \frac{\#\ correct\ n-grams}{\#\ predicted\ n-grams}$
    - $BP = min(1, \frac{output\ length}{reference\ length})$

Conclusion

- Statistical MT
- Neural MT
  - Nowadays use Transformers rather than RNNs

- Encoder-decoder with attention architecture is a general architecture that can be used for other tasks
  - Summarisation (lecture 21)
  - Other generation tasks such as dialogue generation

Lecture 18 Information Extraction

- information extraction
  - Given this: *"Brasilia, the Brazilian capital, was founded in 1960."*
  - Obtain this:
    - capital(Brazil, Brasilia)
    - founded(Brasilia, 1960)

  - Main goal: turn **text** into **structured data**

- applications
  - Stock analysis
    - Gather information from news and social media
    - Summarise texts into a structured format
    - Decide whether to buy/sell at current stock price

  - Medical research
    - Obtain information from articles about diseases and treatments
    - Decide which treatment to apply for new patient

- how
  - 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")

- machine learning in IE
  - Named Entity Recognition (NER): sequence models such as RNNs, HMMs or CRFs.
  - Relation Extraction: mostly classifiers, either binary or multi-class.
  - This lecture: how to frame these two tasks in order to apply sequence labellers and classifiers.

Named Entity Recognition

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

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

- typical entity tags (types of tags to use depend on domains)
  - **PER**(people): people, characters
  - **ORG**(organisation): companies, sports teams
  - **LOC**(natural location): regions, mountains, seas
  - **GPE**(man-made locations): countries, states, provinces (in some tagset this is labelled as **LOC**)
  - **FAC**(facility): bridges, buildings, airports
  - **VEH**(vehcle): planes, trains, cars
  - Tag-set is application-dependent: some domains deal with specific entities e.g. proteins and genes

- NER as sequnce labelling
  - NE tags can be ambiguous:
    - “Washington” can be a person, location or political entity

- Similar problem when doing POS tagging
  - possible solution: Incorporate(包含) context

- Can we use a sequence tagger for this (e.g. HMM)?
  - No, as entities can span multiple tokens(multiple words)
  - Solution: modify the tag set

- IO(inside,outside) tagging
  - [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.
  - 'I-ORG' represents a token that is inside an entity (ORG in this case).
  - All tokens which are not entities get the 'O' token (for **outside**).
  - Cannot differentiate between:
    - a single entity with multiple tokens
    - multiple entities with single tokens

- IOB(beginning) tagging
  - [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.
  - B-ORG represents the *beginning* of an ORG entity.
  - If the entity has more than one token, subsequent tags are represented as I-ORG.

Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

- example: annotate the following sentence with NER tags(IOB)
  - *Steves Jobs founded Apple Inc. in 1976*, Tageset: PER, ORG, LOC, TIME
    - [B-PER **Steves**] [I-PER **Jobs**] [O founded] [B-ORG **Apple**] [I-ORG **Inc.**] [O **in**] [B-Time **1976**]

- NER as sequence labelling
  - Given such tagging scheme, we can train any sequence labelling model
  - In theory, HMMs can be used but **discriminative** models such as CRFs are preferred (HMMs cannot incorperate new features)

- NER
  - features
    - Example: *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.

Word	POS	Chunk	Short shape	Label
American	NNP	B-NP	Xx	B-ORG
Airlines	NNPS	I-NP	Xx	I-ORG
,	,	O	,	O
a	DT	B-NP	x	O
unit	NN	I-NP	x	O
of	IN	B-PP	x	O
AMR	NNP	B-NP	X	B-ORG
Corp.	NNP	I-NP	Xx.	I-ORG
,	,	O	,	O
immediately	RB	B-ADVP	x	O
matched	VBD	B-VP	x	O
the	DT	B-NP	x	O
move	NN	I-NP	x	O
,	,	O	,	O
spokesman	NN	B-NP	x	O
Tim	NNP	I-NP	Xx	B-PER
Wagner	NNP	I-NP	Xx	I-PER
said	VBD	B-VP	x	O
.	,	O	.	O

Figure 18.6 Word-by-word feature encoding for NER.

- classifier

Figure 18.7 Named entity recognition as sequence labeling. The features available to the classifier during training and classification are those in the boxed area.

- deep learning for NER
    - State of the art approach uses LSTMs with character and word embeddings (Lample et al. 2016)

Relation Extraction

- relation extraction
  - [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.
  - Traditionally framed as triple(a relation and two entities) extraction:
    - unit(American Airlines, AMR Corp.)
    - spokesman(Tim Wagner, American Airlines)

  - Key question: do we know all the possible relations?
    - map relations to a closed set of relations
    - unit(American Airlines, AMR Corp.) → subsidiary
    - spokesman(Tim Wagner, American Airlines) → employment

Figure 18.9 The 17 relations used in the ACE relation extraction task.

- methods

  - If we have access to a fixed relation database:
    - Rule-based
    - Supervised

    - Semi-supervised
    - Distant supervision

- If no restrictions on relations:
    - Unsupervised
    - Sometimes referred as “OpenIE”

- rule-based relation extraction
    - *“Agar is a substance prepared from a mixture of red algae such as Gelidium, for laboratory or industrial use.”*
    - identify linguitics patterns in sentence
    - **[NP red algae]** such as **[NP Gelidium]**
    - NP0 such as NP1 → hyponym(NP1, NP0)
    - hyponym(Gelidium, red algae)
    - Lexico-syntactic patterns: high precision, low recall(unlikely to recover all patterns, so many linguistic patterns!), manual effort required
    - more rules

NP {, NP}* {,} (and|or) other NP_H	temples, treasuries, and other important civic buildings
NP_H such as {NP,}* {(or|and)} NP	red algae such as Gelidium
such NP_H as {NP,}* {(or|and)} NP	such authors as Herrick, Goldsmith, and Shakespeare
NP_H {,} including {NP,}* {(or|and)} NP	common-law countries, including Canada and England
NP_H {,} especially {NP}* {(or|and)} NP	European countries, especially France, England, and Spain

Figure 18.12 Hand-built lexico-syntactic patterns for finding hypernyms, using {} to mark optionality (Hearst 1992a, Hearst 1998).

- supervised relation extraction
    - Assume a corpus with annotated relations
    - Two steps (if only one step, class imbalance problem: most entities have no relations!)
        - 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 multiclass classifier (e.g. SVM) to obtain the relation
        - example
            - [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.
            - First:
                - (American Airlines, AMR Corp.) → positive
                - (American Airlines, Tim Wagner) → positive
                - (AMR Corp., Tim Wagner) → negative

            - Second:
                - (American Airlines, AMR Corp.) → subsidiary
                - (American Airlines, Tim Wagner) → employment

    - features

- [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.
- (American Airlines, Tim Wagner) → employment

M1 headword	*airlines* (as a word token or an embedding)
M2 headword	*Wagner*
Word(s) before M1	NONE
Word(s) after M2	*said*
Bag of words between	{*a, unit, of, AMR, Inc., immediately, matched, the, move, spokesman* }
M1 type	ORG
M2 type	PERS
Concatenated types	ORG-PERS
Constituent path	$NP \uparrow NP \uparrow S \uparrow S \downarrow NP$
Typed-dependency path	$Airlines \leftarrow_{subj} matched \leftarrow_{comp} said \rightarrow_{subj} Wagner$

- semi-supervised relation extraction
  - Annotated corpora is very expensive to create
  - Use seed tuples to bootstrap a classifier (use seed to find more training data)
  - steps:
    1. Given seed tuple: hub(Ryanair, Charleroi)
    2. Find sentences containing terms in seed tuples
       - *Budget airline **Ryanair**, which uses **Charleroi** as a hub, scrapped all weekend flights out of the airport*
    3. Extract general patterns
       - [ORG], which uses [LOC] as a hub
    4. Find new tuples with these patterns
       - hub(Jetstar, Avalon)
    5. Add these new tuples to existing tuples and repeat step 2
  - issues
    - Extracted tuples deviate from original relation over time
      - semantic drift(deviate from original relation)
        - Pattern: [NP] has a {NP}* hub at [LOC]
        - *Sydney has a ferry hub at Circular Quay*
          - hub(Sydney, Circular Quay)
        - More erroneous(错误的) patterns extracted from this tuple…
        - Should only accept patterns with high confidences
    - Difficult to evaluate(no labels for new extracted tuples)
    - Extracted general patterns tend to be very noisy
- distant supervision

        - Semi-supervised methods assume the existence of seed tuples to mine new tuples
        - Can we mine new tuples directly?
        - Distant supervision obtain new tuples from a range of sources:
            - DBpedia

            - Freebase

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

    - unsupervised relation extraction

        - No fixed or closed set of relations
        - **Relations are sub-sentences**; usually has a verb
        - "United has a hub in Chicago, which is the headquarters of United Continental Holdings."
            - "has a hub in"(United, Chicago)
            - "is the headquarters of"(Chicago, United Continental Holdings)

        - Main problem: so many relation forms! mapping 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 samples)
            - Can only obtain (approximate) precision, not recall(too many possible relations!)

Other IE Tasks

- temporal expression extraction

  "[TIME **July 2, 2007**]: A fare increase initiated [TIME **last week**] by UAL Corp's United Airlines was matched by competitors over [TIME **the weekend**], marking the second successful fare increase in [TIME **two weeks**]."

    - **Anchoring**: when is "last week"?
        - "last week" → 2007–W26
    - **Normalisation**: mapping expressions to canonical forms.
        - July 2, 2007 → 2007-07-02
    - Mostly **rule-based** approaches
- event extraction
    - "American Airlines, a unit of AMR Corp., immediately [EVENT **matched**] [EVENT **the move**], spokesman Tim Wagner [EVENT **said**]."
    - Very similar to NER but different tags, including annotation and learning methods.
    - **Event ordering**: detect how a set of events happened in a timeline.
        - Involves both event extraction and temporal expression extraction.

Conclusion

- Information Extraction is a vast field with many different tasks and applications
    - Named Entity Recognition
    - Relation Extraction
    - Event Extraction
- Machine learning methods involve classifiers and sequence labelling models.

Lecture 19 Question Answering

introduction

- Definition: question answering ("QA") is the task of automatically determining the answer for a natural language question
- Mostly focus on "factoid" questions
- factoid question(not ambiguious)
    - Factoid questions, have short precise answers:
        - What war involved the battle of Chapultepec?
        - What is the date of Boxing Day?
        - What are some fragrant white climbing roses?
        - What are tannins?
- non-factoid question
    - General non-factoid questions require a longer answer, critical analysis, summary, calculation and more:
        - Why is the date of Australia Day contentious?
        - What is the angle 60 degrees in radians?

- why focus on factoid questions
  - They are easier
  - They have an objective answer
  - Current NLP technologies cannot handle non-factoid answers(under developing)

- 2 key approaches
  - Information retrieval-based QA
    - Given a query, search relevant documents
    - extract answers within these relevant documents

  - Knowledge-based QA
    - Builds semantic representation of the query
    - Query database of facts to find answers

IR-based QA (dominant approach)

- IR-based factoid QA: TREC-QA

  1. Use question to make query for IR engine
     - query formulation: extract key terms to search for documents in the database
     - answer type detection: guess what type of answer is the question looking for

  2. Find document, and passage(段，章) within document(find most relevant passages)
  3. Extract short answer string(use relevant passages and answer type information)

- question processing
  - Find key parts of question that will help retrieval
    - Discard non-content words/symbols (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 (a closed set of answer types)
  - given question, predict answer type
  - key feature is question headword
  - *What are the animals on the Australian coat of arms*?
  - Generally not a difficult task

Tag	Example
ABBREVIATION	
abb	What's the abbreviation for limited partnership?
exp	What does the "c" stand for in the equation E=mc2?
DESCRIPTION	
definition	What are tannins?
description	What are the words to the Canadian National anthem?
manner	How can you get rust stains out of clothing?
reason	What caused the Titanic to sink?
ENTITY	
animal	What are the names of Odin's ravens?
body	What part of your body contains the corpus callosum?
color	What colors make up a rainbow?
creative	In what book can I find the story of Aladdin?
currency	What currency is used in China?
disease/medicine	What does Salk vaccine prevent?
event	What war involved the battle of Chapultepec?
food	What kind of nuts are used in marzipan?
instrument	What instrument does Max Roach play?
lang	What's the official language of Algeria?
letter	What letter appears on the cold-water tap in Spain?
other	What is the name of King Arthur's sword?
plant	What are some fragrant white climbing roses?
product	What is the fastest computer?
religion	What religion has the most members?
sport	What was the name of the ball game played by the Mayans?
substance	What fuel do airplanes use?
symbol	What is the chemical symbol for nitrogen?
technique	What is the best way to remove wallpaper?
term	How do you say " Grandma" in Irish?
vehicle	What was the name of Captain Bligh's ship?
word	What's the singular of dice?
HUMAN	
description	Who was Confucius?
group	What are the major companies that are part of Dow Jones?
ind	Who was the first Russian astronaut to do a spacewalk?
title	What was Queen Victoria's title regarding India?
LOCATION	
city	What's the oldest capital city in the Americas?
country	What country borders the most others?
mountain	What is the highest peak in Africa?
other	What river runs through Liverpool?
state	What states do not have state income tax?
NUMERIC	
code	What is the telephone number for the University of Colorado?
count	About how many soldiers died in World War II?
date	What is the date of Boxing Day?
distance	How long was Mao's 1930s Long March?
money	How much did a McDonald's hamburger cost in 1963?
order	Where does Shanghai rank among world cities in population?
other	What is the population of Mexico?
period	What was the average life expectancy during the Stone Age?
percent	What fraction of a beaver's life is spent swimming?
temp	How hot should the oven be when making Peachy Oat Muffins?
speed	How fast must a spacecraft travel to escape Earth's gravity?
size	What is the size of Argentina?
weight	How many pounds are there in a stone?

- retrieval
  - Find top *n* documents matching query (standard IR)
  - Next find passages (paragraphs or sentences) in these documents (also driven by IR)
  - a good passage should contain:
    - many instances of the question keywords
    - several named entities of the answer type
    - close proximity(靠近) of these terms in the passage
    - high ranking by IR engine

    - 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 passage
        - "Who is the federal MP for Melbourne?"
        - *The Division of Melbourne is an Australian Electoral Division in Victoria, represented since the 2010 election by Adam Bandt, a member of the Greens.*
        - "How many Australian PMs have there been since 2013?"
        - *Australia has had five prime ministers in five years. No wonder Merkel needed a cheat sheet at the G-20.*

    - how?
        - Use a neural network to extract answer
        - AKA **reading comprehension** task(assuming query and evidence passage are given, and find the span)
        - But deep learning models require lots of data
        - Do we have enough data to train comprehension models?

    - dataset
        - MCTest(a dataset)
            - Crowdworkers write fictional stories, questions and answers
            - 500 stories, 2000 questions
            - Multiple choice questions

James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.

One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.

After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

1) What is the name of the trouble making turtle?
A) Fries
B) Pudding
C) James
D) Jane

- SQuAD
  - Use Wikipedia passages(easier to create than MCTest)
  - First set of crowdworkers create questions (given passage)
  - Second set of crowdworkers label the answer
  - 150K questions (!)
  - Second version includes unanswerable questions(no answer in the passage)

Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Q: "In what city and state did Beyoncé grow up?" A: "Houston, Texas"
Q: "What areas did Beyoncé compete in when she was growing up?" A: "singing and dancing"
Q: "When did Beyoncé release Dangerously in Love?" A: "2003"

- reading comprehension
  - Given a question and context passage, **predict where the answer span starts and end in passage**?
  - Compute:
    - $P_{start}(i)$: prob. of token i is the starting token
    - $P_{end}(i)$: prob. of token i is the ending token

Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Q: "In what city and state did Beyoncé grow up?" A: "Houston, Texas"

  - LSTM-based model
    - Feed question tokens to a bidirectional LSTM
    - Aggregate LSTM outputs via weighted sum to produce *q*, the final **question embedding**

- Process passage in a similar way, using **another** bidirectional LSTM
- More than just word embeddings as input
  - A feature to denote whether the word matches a question word
  - POS feature
  - Weighted question embedding: produced by attending to each question words

- $\{p_1, \ldots, p_m\}$: one vector for each passage token from bidirectional LSTM
- To compute start and end probability for each token
  - $p_{start}(i) \propto exp(p_i W_s q)$
  - $P_{end}(i) \propto exp(p_i W_e q)$

- BERT-based model
  - Fine-tune BERT to predict answer span
    - $p_{start}(i) \propto exp(S^T T_i')$
    - $p_{end}(i) \propto exp(E^T T_i')$

  - why BERT works better than LSTM
    - It's pre-trained and so already "knows" language before it's adapted to the task
    - Self-attention architecture allows fine-grained analysis between words in question and context paragraph

Knowledge-based QA

- QA over structured KB
  - Many large knowledge bases
    - Freebase, DBpedia, Yago, …
  - Can we support natural language queries?

- E.g.
  - 'When was Ada Lovalace born?' → birth-year (Ada Lovelace, ?x)
  - 'What is the capital of England' → capital-city(?x, England)
- Link “Ada Lovelace” with the correct entity in the KB to find triple (*Ada Lovelace, birth-year, 1815*)

- but
  - Converting natural language sentence into triple is not trivial
    - 'When was Ada Lovelace born' → birth-year (Ada Lovelace, ?x)
  - Entity linking also an important component
    - Ambiguity: “When was **Lovelace** born?”
  - Can we simplify this two-step process?

- semantic parsing
  - Convert questions into logical forms to query KB directly
    - Predicate calculus
    - Programming query (e.g. SQL)

Question	Logical form
What states border Texas?	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$
What is the largest state?	$\text{argmax}(\lambda x.\text{state}(x), \lambda x.\text{size}(x))$
I’d like to book a flight from San Diego to Toronto	SELECT DISTINCT f1.flight_id FROM flight f1, airport_service a1, city c1, airport_service a2, city c2 WHERE f1.from_airport=a1.airport_code AND a1.city_code=c1.city_code AND c1.city_name= ’san diego’ AND f1.to_airport=a2.airport_code AND a2.city_code=c2.city_code AND c2.city_name= ’toronto’
How many yards longer was Johnson’s longest touchdown compared to his shortest touchdown of the first quarter?	ARITHMETIC diff(SELECT num(ARGMAX(SELECT)) SELECT num(ARGMIN(FILTER(SELECT))))

    - how to build a semantic parser
      - Text-to-text problem:
        - Input = natural language sentence
        - Output = string in logical form
      - Encoder-decoder model(but do we have enough data?)

Hybrid QA

- hybrid methods
  - Why not use both text-based and knowledgebased resources for QA?
  - IBM's Watson which won the game show Jeopardy! uses a wide variety of resources to answer questions
    - (question)THEATRE: A new play based on this Sir Arthur Conan Doyle canine classic opened on the London stage in 2007.
    - (answer)The Hound Of The Baskervilles

- core idea of Watson
  - Generate lots of candidate answers from textbased and knowledge-based sources
  - Use a rich variety of evidence to score them
  - Many components in the system, most trained separately

- QA evaluation
  - IR: Mean Reciprocal Rank for systems returning matching passages or answer strings
    - E.g. system returns 4 passages for a query, first correct passage is the 3rd passage
    - MRR = 1/3
  - MCTest: Accuracy
  - SQuAD: Exact match of string against gold answer

Conclusion

- IR-based QA: search textual resources to answer questions
  - Reading comprehension: assumes question+passage
- Knowledge-based QA: search structured resources to answer questions
- Hot area: many new approaches & evaluation datasets being created all the time (narratives, QA, commonsense reasoning, etc)

Lecture 20 Topic Modelling

Topic Modelling

- makeingsense of text
  - English Wikipedia: 6M articles
  - Twitter: 500M tweets per day
  - New York Times: 15M articles
  - arXiv: 1M articles
  - What can we do if we want to learn something about these document collections?

- questions
  - What are the less popular topics on Wikipedia?
  - What are the big trends on Twitter in the past month?
  - How do the themes/topics evolve over time in New York Times from 1900s to 2000s?
  - What are some influential research areas?

- topic models to the rescue
  - Topic models learn common, overlapping themes in a document collection
  - Unsupervised model
    - No labels; input is just the documents!

  - What's the output of a topic model?
    - Topics: each topic associated with a list of words
    - Topic assignments: each document associated with a list of topics

- what do topics look like
  - A list of words
  - Collectively describes a concept or subject
  - Words of a topic typically appear in the same set of documents in the corpus

- Wikipedia topics

- Twitter topics

- New York Times topics

music band songs rock album jazz pop song singer night	book life novel story books man stories love children family	art museum show exhibition artist artists paintings painting century works	game knicks nets points team season play games night coach	show film television movie series says life man character know
theater play production show stage street broadway director musical directed	clinton bush campaign gore political republican dole presidential senator house	stock market percent fund investors funds companies stocks investment trading	restaurant sauce menu food dishes street dining dinner chicken served	budget tax governor county mayor billion taxes plan legislature fiscal

- applications of topic models
  - Personalised advertising
  - Search engine
  - Discover senses of polysemous words
  - Part-of-speech tagging

A Brief History of Topic Models

- latent semantic analysis

- LSA: truncate

- issues
  - Positive and negative values in the U and V^T
  - Difficult to interpret

- probabilistic LSA
  - based on a probabilistic model

  - issues
    - No more negative values!
    - PLSA can learn topics and topic assignment 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(LDA)
  - 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

LDA

- LDA
  - Core idea: assume each document contains a mix of topics
  - But the topic structure is hidden (latent)
  - LDA infers the topic structure given the observed words and documents
  - LDA produces soft clusters of documents (based on topic overlap), rather than hard clusters
  - Given a trained LDA model, it can infer topics on new documents (not part of train data)

- input
  - A collection of documents
  - Bag-of-words
  - Good preprocessing practice:
    - Remove stopwords
    - Remove low and high frequency word types
    - Lemmatisation

- output
  - Topics: distribution over words in each topic

- Topic assignment: distribution over topics in each document

- learning
  - How do we learn the latent topics?
  - 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: $P(t_i|w,d) \propto P(t_i|w)P(t_i|d)$
    4. Go to step 2 and repeat until convergence
    - when to stop
      - Train until convergence
      - Convergence = model probability of training set becomes stable
      - How to compute model probability?
        - $logP(w_1, w_2, \ldots, w_m) = log \sum_{j=0}^{T} P(w_1|t_j)P(t_j|d_{w_1}) + \ldots + log \sum_{j=0}^{T} P(w_m|t_j)P(t_j|d_{w_m})$
        - m = #word tokens
        - $P(w_1|t_j) \rightarrow$ based on the topic-word co-occurrence matrix
        - $P(t_j|d_{w_1}) \rightarrow$ based on the document-topic co-occurrence matrix
    - infer topics for new documents
      1. Randomly assign topics to all tokens in new/test documents
      2. Update document-topic matrix based on the assignments; but use the trained topic-word matrix (kept fixed)
      3. Go through every word in the test documents and sample topics: $P(t_i|w,d) \propto P(t_i|w)P(t_i|d)$
      4. Go to step 2 and repeat until convergence

- hyper-parameters
  - T: number of topic

Low T (<10): broad topics

High T (100+): fine-grained, specific topics

  - β: prior on the topic-word distribution
  - α: prior on the document-topic distribution
  - Analogous to *k* in add-*k* smoothing in *N*-gram LM
  - Pseudo counts to initialise co-occurrence matrix:
    - $p(w \mid t)$: β
    - $p(t \mid d)$: α

	mouse	cat	scroll	tv	...
t_1	0.01	0.01	0.01	0.01	
t_2	0.01	0.01	0.01	0.01	
t_3	0.01	0.01	0.01	0.01	

	t_1	t_2	t_3
d_1	0.1	0.1	0.1
d_2	0.1	0.1	0.1
...			

  - High prior values → flatter distribution

    - a very very large value would lead to a uniform distribution
  - Low prior values → peaky distribution

- β: generally small (< 0.01)
    - Large vocabulary, but we want each topic to focus on specific themes
- α: generally larger (> 0.1)
    - Multiple topics within a document

Evaluation

- how to evaluate topic models
    - Unsupervised learning → no labels
    - Intrinsic evaluation:
        - model logprob / perplexity on test documents
        - $logL = \sum_W \sum_T logP(w|t)P(t|d_w)$
        - $ppl = exp^{\frac{-logL}{W}}$
- issues with perlexity
    - 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
    - {farmers, farm, food, rice, agriculture} → {farmers, farm, food, rice, cat, agriculture}
  - Ask users to guess which is the intruder word
  - Correct guess → topic is coherent
  - Try guess the intruder word in:
    - {choice, count, village, i.e., simply, unionist}
  - Manual effort; does not scale

- PMI ≈ coherence?
  - High PMI for a pair of words → words are correlated
    - PMI(farm, rice) ↑
    - PMI(choice, village) ↓
  - If all word pairs in a topic has high PMI → topic is coherent
  - If most topics have high PMI → good topic model
  - Where to get word co-occurrence statistics for PMI?
    - Can use same corpus for topic model
    - A better way is to use an external corpus (e.g. Wikipedia)

- PMI
  - Compute pairwise PMI of top-N words in a topic
    - $PMI(t) = \sum_{j=2}^{N} \sum_{i=1}^{j-1} log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$
  - Given topic: {farmers, farm, food, rice, agriculture}
  - Coherence = sum PMI for all word pairs:
    - PMI(farmers, farm) + PMI(farmers, food) + … + PMI(rice, agriculture)
  - variants
    - Normalised PMI
      - $NPMI(t) = \sum_{j=2}^{N} \sum_{i=1}^{j-1} \frac{log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-logP(w_i, w_j)}$
    - conditional probability

        - $LCP(t) = \sum_{j=2}^{N} \sum_{i=1}^{j-1} log \frac{P(w_i, w_j)}{P(w_i)}$

    - example

Topic	PMI	NPMI
cell hormone insulin muscle receptor	0.61	0.59
electron laser magnetic voltage wavelength	0.54	0.52
magnetic neutrino particle quantum universe	0.55	0.55
album band music release song	0.37	0.56
college education school student university	0.38	0.57
city county district population town	0.34	0.52

Conclusion

- Topic model: an unsupervised model for learning latent concepts in a document collection
- LDA: a popular topic model
    - Learning
    - Hyper-parameters

- How to evaluate topic models?
    - Topic coherence

Lecture 21 Summarisation

- summarisation
    - Distill the most important information from a text to produce shortened or abridged version
    - Examples
        - outlines of a document
        - abstracts of a scientific article
        - headlines of a news article
        - snippets of search result

    - what to summarise
        - Single-document summarisation
            - Input: a single document
            - Output: summary that characterise the content

        - Multi-document summarisation
            - Input: multiple documents
            - Output: summary that captures the gist of all documents
            - E.g. summarise a news event from multiple sources or perspectives

- how to summarise
  - Extractive summarisation
    - Summarise by selecting representative sentences from documents
  - Abstractive summarisation
    - Summarise the content in your own words
    - Summaries will often be paraphrases of the original content
- goal of summarisation
  - Generic summarisation
    - Summary gives important information in the document(s)
  - Query-focused summarisation
    - Summary responds to a user query
    - "Non-factoid" QA
    - Answer is much longer than factoid QA

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

- We will focus on content selection
- For single-document summarisation, information ordering not necessary
  - present extracted sentences in original order

- Sentence realisation also not necessary if they are presented in dot points

- content selection
  - Not much data with ground truth extractive sentences
  - Mostly unsupervised methods
  - Goal: Find sentences that are important or salient
  - method1: TF-IDF
    - Frequent words in a doc → salient
    - But some generic words are very frequent but uninformative
      - function words
      - stop words

    - Weigh each word w in document d by its inverse document frequency:

      - $weight(w) = tf_{d,w} \times idf_w$

  - method 2: Log Likelihood Ratio
    - Intuition: a word is salient if its probability in the input corpus is very different to a background corpus
    - $f(n) = \begin{cases} 1, & if - 2log\lambda(w) > 10 \\ 0, & otherwise \end{cases}$
    - $\lambda(w)$ is the ratio between:

      - P(observing w in I) → $\begin{pmatrix} N_I \\ x \end{pmatrix} p^x(1-p)^{N_I - x}$ and P(observing w in B) → $\begin{pmatrix} N_B \\ y \end{pmatrix} p^y(1-p)^{N_B - y}$, assuming $P(w|I) = P(w|B) = p \rightarrow \frac{x+y}{N_I + N_B}$
      - P(observing w in I) → $\begin{pmatrix} N_I \\ x \end{pmatrix} p_I^{x_I}(1-p_I)^{N_I - x}$ and P(observing w in B) → $\begin{pmatrix} N_B \\ y \end{pmatrix} p_B^y(1-p_B)^{N_B - y}$, assuming $P(w|I) = p_I \rightarrow \frac{x}{N_I}$ and $P(w|B) = p_B \rightarrow \frac{y}{N_B}$

    - saliency of a sentence
      - $weight(s) = \frac{1}{|S|} \sum_{w \in S} weight(w)$
      - only consider non-stop words in S

  - 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 BOW to represent sentence
    - Use cosine similarity to measure distance

- $centrality(s) = \frac{1}{\#sent} \sum_{s'} cos_{tfidf}(s, s')$
- final extracted summary
  - Use top-ranked sentences as extracted summary
    - Saliency (tf-idf or log likelihood ratio)
    - Centrality

- method 4: RST parsing

With its distant orbit – 50 percent farther from the sun than Earth – and slim atmospheric blanket, Mars experiences frigid weather conditions. Surface temperatures typically average about -70 degrees Fahrenheit at the equator, and can dip to -123 degrees C near the poles. Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion, but any liquid water formed in this way would evaporate almost instantly because of the low atmospheric pressure. Although the atmosphere holds a small amount of water, and water-ice clouds sometimes develop, most Martian weather involves blowing dust or carbon dioxide.

- 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

- which sentence is the best summary sentence?

Extractive: Multi-Doc

- summarisation system
  - Similar to single-document extractive summarisation system
  - Challenges:
    - Redundancy in terms of information
    - Sentence ordering

  - content selection
    - We can use the same unsupervised content selection methods (tf-idf, log likelihood ratio, centrality) to select salient sentences
    - But ignore sentences that are redundant

  - Maximum Marginal Relevance
    - Iteratively select the best sentence to add to summary
    - Sentences to be added must be novel
    - Penalise a candidate sentence if it's similar to extracted sentences:
      - $MMR - penalty(s) = \lambda max_{s_i \in S} sim(s, s_i)$

    - Stop when a desired number of sentences are added

  - 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

Original summary:
Presidential advisers do not blame **O'Neill**, but they've long recognized that a shakeup of the economic team would help indicate **Bush** was doing everything he could to improve matters. **U.S. President George W. Bush** pushed out **Treasury Secretary Paul O'Neill** and top economic adviser Lawrence Lindsey on Friday, launching the first shake - up of his administration to tackle the ailing economy before the 2004 election campaign.
Rewritten summary:
Presidential advisers do not blame **Treasury Secretary Paul O'Neill**, but they've long recognized that a shakeup of the economic team would help indicate **U.S. President George W. Bush** was doing everything he could to improve matters. **Bush** pushed out **O'Neill** and White House economic adviser Lawrence Lindsey on Friday, launching the first shake-up of his administration to tackle the ailing economy before the 2004 election campaign.

Abstractive: Single-Doc

- example

*a detained **iranian-american academic** accused of acting against national security has been **released** from a **tehran** prison after a hefty **bail** was posted, a top judiciary official said tuesday*

↓

iranian-american academic** held in **tehran released** on **bail

- Paraphrase
- A very difficult task
- Can we train a neural network to generate summary?

- Encoder-Decoder?

- What if we treat:
  - Source sentence = “document”
  - Target sentence = “summary”

*a detained **iranian-american academic** accused of acting against national security has been **released** from a **tehran** prison after a hefty **bail** was posted, a top judiciary official said tuesday*

↓

iranian-american academic** held in **tehran released** on **bail

- data
  - News headlines
  - Document: First sentence of article
  - Summary: News headline/title
  - Technically more like a “headline generation task”
  - and it kind of works..

    I(1): a detained iranian-american academic accused of acting against national security has been released from a tehran prison after a hefty bail was posted , a to p judiciary official said tuesday .
    G: iranian-american academic held in tehran released on bail
    A: detained iranian-american academic released from jail after posting bail
    A+: detained iranian-american academic released from prison after hefty bail

    I(2): ministers from the european union and its mediterranean neighbors gathered here under heavy security on monday for an unprecedented conference on economic and political cooperation .
    G: european mediterranean ministers gather for landmark conference by julie bradford
    A: mediterranean neighbors gather for unprecedented conference on heavy security
    A+: mediterranean neighbors gather under heavy security for unprecedented conference
  - More Summarisation Data
    - But headline generation isn’t really exciting…
    - Other summarisation data:
      - CNN/Dailymail: 300K articles, summary in bullets
      - Newsroom: 1.3M articles, summary by authors
        - Diverse; 38 major publications
      - XSum: 200K BBC articles
        - Summary is more abstractive than other datasets
- improvements
  - Attention mechanism
  - Richer word features: POS tags, NER tags, tf-idf
  - Hierarchical encoders
    - One LSTM for words
    - Another LSTM for sentences

- Potential issues of an attention encoderdecoder summarisation system?
  - Has the potential to generate new details not in the source document
  - Unable to handle unseen words in the source document
  - Information bottleneck: a vector is used to represent the source document
  - Can only generate one summary

Encoder-decoder with Attention

Encoder-decoder with Attention + Copying

- 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

- latest development
  - State-of-the-art models use transformers instead of RNNs
  - Lots of pre-training
  - Note: BERT not directly applicable because we need a unidirectional decoder (BERT is only an encoder)

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 separately (e.g. from 1 to 3)
  - ROUGE-2: calculates the percentage of bigrams from the reference that are in the generated summary

- ROUGE-2: example

$$\text{ROUGE-2} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{bigram \in S} \text{Count}_{\text{match}}(bigram)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{bigram \in S} \text{Count}(bigram)}$$

- **Ref 1:** *Water spinach is a green leafy vegetable grown in the tropics.*
- **Ref 2:** *Water spinach is a commonly eaten leaf vegetable of Asia.*
- **Generated summary:** *Water spinach is a leaf vegetable commonly eaten in tropical areas of Asia.*
- *ROUGE-2* $= \frac{3+6}{10+9}$

Conclusion

- Research focus on single-document abstractive summarisation
- Mostly news data
- But many types of data for summarisation:
  - Images, videos
  - Graphs
  - Structured data: e.g. patient records, tables
  - Multi-document abstractive summarisation

Lecture 22 Ethics

How we ought to live — Socrates
- what is ethics
- What is the right thing to do?
- Why?
- Why Should We Care?
- AI technology is increasingly being deployed to real-world applications
- Have real and tangible impact to people
- Whose responsibility is it when things go bad?
- Why Is Ethics Hard?
- Often no objective truth, unlike sciences
- A new philosophy student may ask whether fundamental ethical theories such as utilitarianism is right
- But unlikely a new physics student would question the laws of thermodynamics
- In examining a problem, we need to think from different perspectives to justify our reasons
- Learning Outcomes
- Think more about the application you build
- Not just its performance

- Its social context
- Its impact to other people
- Unintended harms
- Be a socially-responsible scientist or engineer

Arguments Against Ethical Checks in NLP

- Should We Censor Science?
  - A common argument when ethical checks or processes are introduced:
    - Should there be limits to scientific research? Is it right to censor research?
  - Ethical procedures are common in other fields: medicine, biology, psychology, anthropology, etc
  - In the past, this isn't common in computer science
  - But this doesn't mean this shouldn't change
  - Technology are increasingly being integrated into society; the research we do nowadays are likely to be more deployed than 20 years ago
- H5N1
  - Ron Fouchier, a Dutch virologist, discovered how to make bird flu potentially more harmful in 2011
  - Dutch government objected to publishing the research
  - Raised a lot of discussions and concerns
  - National policies enacted
- Isn't Transparency Always Better?
  - Is it always better to publish sensitive research publicly?
  - Argument: worse if they are done underground
  - If goal is to raise awareness, scientific publication isn't the only way
    - Could work with media to raise awareness
    - Doesn't require exposing the technique
- AI vs. Cybersecurity
  - Exposing vulnerability publicly is desirable in cyber-security applications
    - Easy for developer to fix the problem
  - But the same logic doesn't always apply for AI
    - Not easy to fix, once the technology is out

Core NLP Ethics Concepts

- Bias
  - Two definitions:
    - Value-neutral meaning in ML
    - Normative meaning in socio-cultural studies
  - Ethics research in NLP: harmful prejudices in models

    - A biased model is one that performs unfavourably against certain groups of users
        - typically based on demographic features such as gender or ethnicity
    - Bias isn't necessarily bad
        - Guide the model to make informed decisions in the absence of more information
        - Truly unbiased system = system that makes random decisions
        - Bad when overwhelms evidence, or perpetuates harmful stereotypes
    - Bias can arise from data, annotations, representations, models, or research design
- Bias in Word Embeddings
    - Word Analogy (lecture 10):
        - v(man) - v(woman) = v(king) - v(queen)
    - But!
        - v(man) - v(woman) = v(programmer) - v(homemaker)
        - v(father) - v(mother) = v(doctor) - v(nurse)
    - Word embeddings reflect and amplify gender stereotypes in society
    - Lots of work done to reduce bias in word embeddings
- Dual Use
    - Every technology has a primary use, and unintended secondary consequences
        - uclear power, knives, electricity
        - could be abused for things they are not originally designed to do.
    - Since we do not know how people will use it, we need to be aware of this duality
- OpenAI GPT-2
    - OpenAI developed GPT-2, a large language model trained on massive web data
    - Kickstarted the pretrained model paradigm in NLP
        - Fine-tune pretrained models on downstream tasks (BERT lecture 11)
    - GPT-2 also has amazing generation capability
        - Can be easily fine-tuned to generate fake news, create propaganda
    - Pretrained GPT-2 models released in stages over 9 months, starting with smaller models
    - Collaborated with various organisations to study social implications of very large language models over this time
    - OpenAI's effort is commendable, but this is voluntary
    - Further raises questions about self-regulation
- Privacy
    - Often conflated with anonymity
    - Privacy means nobody know I am doing something
    - Anonymity means everyone know what I am doing, but not that it is me
- GDPR
    - Regulation on data privacy in EU

    - Also addresses transfer of personal data
    - Aim to give individuals control over their personal data
    - Organisations that process EU citizen's personal data are subjected to it
    - Organisations need to anonymise data so that people cannot be identified
    - But we have technology to de-identify author attributes

- AOL Search Data Leak
    - In 2006, AOL released anonymised search logs of users
    - Log contained sufficient information to de-identify individuals
        - Through cross-referencing with phonebook listing an individual was identified

    - Lawsuit filed against AOL

Group Discussion

- Prompts
    - Primary use: does it promote harm or social good?
    - Bias?
    - Dual use concerns?
    - Privacy concerns? What sorts of data does it use?
    - Other questions to consider:
        - Can it be weaponised against populations (e.g. facial recognition, location tracking)?
        - Does it fit people into simple categories (e.g. gender and sexual orientation)?
        - Does it create alternate sets of reality (e.g. fake news)?

- Automatic Prison Term Prediction
    - A model that predicts the prison sentence of an individual based on court documents

- Automatic CV Processing
    - A model that processes CV/resumes for a job to automatically filter candidates for interview

- Language Community Classification
    - A text classification tool that distinguishes LGBTQ from heterosexual language
    - Motivation: to understand how language used in the LGBTQ community differs from heterosexual community

- Take Away
    - Think about the applications you build
    - Be open-minded: ask questions, discuss with others
    - NLP tasks aren't always just technical problems
    - Remember that the application we build could change someone else's life
    - We should strive to be a socially responsible engineer/scientist