Natural Language Processing

Michael Noukhovitch

Fall 2020, McGill

Notes written from Jackie Cheung's lectures

Contents

1 Introduction

1.1 Overview

language is a form of communication

- *arbitrary* pairing between form and meaning
- very expressive and productive
- nearly universal
- uniquely human*

computational linguistics modelling natural language with computational models

- acoustic signals
- NL understanding (comprehension)
- NL generation (production)

goals of the field

- practical technologies (NLP)
- understanding how language works (CL)

models and techniques

- gathering data
- evaluation
- statistical methods (ML)
- rule-based systems

some example problems

- is language an instinct? (Chomsky)
- language processing to understand meaning of sentence
- can we learn mathematical properties of language

types of language

- **text** an idealization of spoken language
  - luckily English is similar between writing and speaking, and there is lots of data on it
  - older work used "clean" language but recent work ventures into messy data (e.g. Twitter)
- **speech** is much messier
  - automatic speech recognition (ASR)
  - text-to-speech generation (TTS)

1.2 Domains of Language

phonetics study of speech sounds

- articulation, transmission
- how each sound is made in the mouth

phonology rules that govern sound patterns

- how the sounds are organized
- "p" in peach and speach are the same phoneme but phonetically distinct (aspiration)

morphology word formation and meaning

- anti-dis-establish-ment-arian-ism

syntax structure of language

- "I a woman saw park in the" is **ungrammatical**
- **ambiguity** different possible meaning for the same phrase

semantics meaning of language

- "Ross wants to marry **a** Swedish woman"

pragmatics meaning of language in context

- different from literal meaning
- **deixis** interpretation that relies on extra-linguistic context
- "dessert would be delicious"

discourse structure of larger spans of language

- do large spans of text form a coherent story

1.3 Technology

combination of hand-crafted knowledge and ML on data

- rule-based systems
- machine learning
- knowledge representation

2 Text Classification

2.1 Basics

text classification assign a label or category to a piece of text

- sentiment analysis
- spam detection
- language identification
- authorship attribution

supervised output data is labelled

- learn a function, minimize θ with loss on data
- e.g. spam classificaiton, predict POS
- **regression** y is continuous
- **classification** y is discrete

unsupervised output data is unlabelled

- learn a density
- e.g. grammar induction, word-relatedness (word2vec)

2.2 Building a text classifier

- define problem, collect data
- extract feats
- train a classifier on train data
- apply classifier to test data

problem definition

- problem
- input
- output categories
- how to annotate

2.3 Feature Extraction

feature extraction get "important" properties of documents

- convert text into numerical format
- e.g. word counts as features *unigram counts*

lemma remove affixes get dictionary word "flies → fly" **stemming** remove affix get stem "airliner → airlin"

- rule-based e.g. (Porter, 1980) "ies → i"

n-grams sequences of adjacent words

- presence or absence
- counts
- proportion of total document
- scaled version (tf-idf)

POS tags crudely capture syntactic pattern (PTB dataset) **stop-word removal** remove common uninformative words
sentence representation

- vector addition $v_{hello} + v_{world}$
- vector multiplication $v_{hello} \cdot v_{world}$
- **max pooling** choose the max of vectors

2.4 Models

training select parameters θ^* according to some objective
types of models

- **generative** models joint distribution $P(x, y)$
  - less flexible features as they need to be consistent with each other
- **discriminative** models conditional $P(y|x)$
  - can be more flexible in terms of features

2.4.1 Naive Bayes

Naive Bayes probabilistic classifier that uses Bayes' Rule $P(y|x) = \frac{P(y)P(x|y)}{P(x)}$

- generative
- assumes data x is generated independently conditioned on class $P(x_i|y)$
- graphical assumption $P(x, y) = P(y) \prod_i P(x_i|y)$

In NLP, we can assume NB over a *categorical* distribution and train

- loss $L = \prod_{(x,y)\in D} P(y) \prod_i P(x_i|y)$
- learn $P(Y = y)$ proportion of samples with class y
- learn $P(X_i = x|Y = y)$ proportion of samples with feature x given class y

Inference time we want $P(y|x)$

$$P(y|x) = P(x, y)|P(x) \tag{1}$$
$$= P(y) \prod_i P(x_i|y)/P(x) \tag{2}$$

where $P(x)$ is the marginalized over all classes

2.4.2 Logistic Regression

logistic regression linear regression with a logit activation

- $P(y|x) = \frac{1}{Z} \exp(\sum_i a_i x_i)$
- squash output between $(0, 1)$

train log-likelihood with *gradient descent*

$$\log L(\theta) = \prod_{(x,y)\in D} \log P(y|x; \theta) \tag{3}$$
$$= \prod_{(x,y)\in D} (\sum_i a_i x_i - \log Z) \tag{4}$$

2.4.3 Support Vector Machines

SVM learns linear decision to maximize margin to nearest sample in each of two classes

- can be non-linear using *kernels*

2.4.4 Neural Network

Perceptron logistic regression with Perceptron learning rule $f(x) = \begin{cases} 1 & \text{if } wx + b > 0 \\ 0 & \text{else} \end{cases}$

Stacked Perceptron stacks perceptron neurons
Artificial Neural Network stacked neurons with non-linear activation functions

- can learn complex functions
- need lots of data and computational power
- given enough neurons can compute any function
- highly flexible, generic architecture
- **multi-task learning** train model to solve multiple tasks simultaneously

- **transfer learning** use a network on a task different from its training

Feed-forward network

$$h_1 = g_1(W_1 x + b_1) \tag{5}$$
$$h_2 = g_2(W_2 h_1 + b_2) \tag{6}$$
$$y = W_2 h_2 \tag{7}$$

Activation function on the output of a neuron

- sigmoid
- tanh
- ReLU
- softmax $\frac{exp(x_i)}{\sum_j \exp(x_j)}$

Training neural network done with **gradient descent** on loss function

- **backpropogation** (Rumelhart et al, 1986) uses chain rule to pass gradient
- **SGD** apply gradient iteratively over mini-batches of data

2.5 Model Selection

How to choose preprocessing, model, etc.. evaluate on unseen data!
Data split

- **training** learning the model, 60-90%
- **dev/validation** evaluating while learning the model
- **testing** evaluate once at the end to see how well you do

k-fold cross-validation split training data into k folds, train on $k-1$ fold and test on the last
Evaluation measures

- **accuracy** correct / total samples
- **precision** true positives / all predicted positives
- **recall** true positives / true number of positives
- **F1** $\frac{2*P*R}{P+R}$, can average with P, R, F1

You can average evaluation method across classes

- **macro-average** takes the average after computing P, R, F1 for each class
  - $\frac{P_{spam}+P_{not-spam}}{2}$
  - weighs each *class* equally
- **micro-average** sum of counts first then compute P, R, F

  - $P_{micro} = \frac{TP_{spam}+TP_{not-spam}}{AP_{spam}+AP_{not-spam}}$
  - weighs each *sample* equally

key issues

- which eval measure to use
- stastical significance of test
- do these tests matter?

3 Language Modelling

3.1 Words

word smallest unit that can appear in isolation

- philosophically not so clear e.g. "peanut butter" vs "football"
- practically: *spaces delimit words*

word features

- **type** identity of a word (count each word once)
- **token** instance of a word (count number of occurences)

counting words

- **term frequency** $TF(w, S)$ =number of w in corpus S
- **relative frequency** $RF(w, S) = \frac{TF(w,S)}{|S|}$

corpus of text to use, e.g.

- Brown
- British National Corpus
- WSJC

Zipf's law on the "long tail" $f \propto \frac{1}{r}$

- frequency of word is inversely proportional to its rank (by frequency in the corpus)
- 100th most common word appears 100x less than most common word
- **Zipf-Mandelbrot** $f = \frac{P}{(r+\rho)^B}$
- proportion differs by language

3.2 N-gram Language Models

language modelling predicting the next word in a context $P(W = w|C)$

- can break down with chain rule $P(w_1, 2) = P(w_2|w_1)P(w_1)$
- should assign higher probability to grammatical sentence
- shouldn't be used to predict grammaticality
- captures linguistic knowledge and maybe facts about world

n-grams feature using $N - 1$ previous words as context using MLE

- **unigram** 1 word, $P(cats) = \frac{\text{count(cats)}}{\text{count(all words}}$
- **bigram** 2 word $P(cats|the) = \frac{(counts(thecats)}{\text{count(the)}}$
- **trigram** $N = 3$ word context

evaluation measure

- **likelihood** of generating the test corpus
- **cross-entropy** $H(p, q) = -\sum_i p_i \log_2 q_i$ with model distribtuion q
  - information $I(x) = \log_2 \frac{1}{P(x)}$
  - entropy $H(p) = -\sum_i p_i \log_2 p_i$
  - approx cross entropy with $H(p, q) = -\frac{1}{N} \log_2 q(w_1, \ldots, w_N)$
- **perplexity** $2^{H(p,q)}$ to make differences larger

3.2.1 Learning

maximum likelihood estimation choose parameters θ to maximize likelihood training corpus X

- i.i.d assumption of words in corpus $P(C; \theta) = \prod_n P(x_n; \theta)$
- for a Bernoulli $P(C; \theta) = \theta^{N_1}(1 - \theta)^{N_0} \ldots \theta_{MLE} = \frac{N_1}{N_0 + N_1}$

two issues with learning

- **overfitting** with high train accuracy, low test data accuracy
- **underfitting** low train accuracy

model complexity tradeoff

- highly expressive models overfit easier
- choose a model class and regularization/smoothing technique

3.2.2 Smoothing

out-of-vocabulary (OOV) at test time is a real problem
simple solution

- replace all words less than freq threshold with `<UNK>`
- treat `<UNK>` as a vocabulary item

smoothing probability distribution, move mass to unseen cases

- don't trust frequency counts entirely
- no longer doing MLE but can use θ_{MLE} prior
- $\theta_{smooth} = \max_\theta P(X;\theta)P(\theta)$

add-δ smoothing add frequency to each word (*pseudocount*(

- for unigram $P(w) = \frac{count(w)+\delta}{N+\delta*|Lexicon|}$
- with $\delta = 1$ called **laplace discounting**

interpolation lower N in N-gram to alleviate data sparsity

- combine $\hat{P} = \lambda_1 P^{MLE}_{unigram} + \lambda_2 P^{MLE}_{bigram} + \lambda_3 P^{MLE}_{trigram}$

Good-Turing Smoothing assume probability based on zipf's law, assuming all UNK appear once and readjust others accordingly

- build a histogram of event frequency f_c (e.g. 1000 words appear once $f_1 = 1000$, 600 words appear twice $f_2 = 600$)
- total number of observed event-tokens $N = \sum_i f_i * i$
- w_c is event that occurs c times
- choose probability for all unseen $P(UNK) = \frac{f_1}{N}$
- readjust other mass $c^* = \frac{(c+1)f_{c+1}}{f_c}, P(w_c) = \frac{c^*}{N}$

GT-refinement estimate linear regression $f^{LR}_c = a \log c + b$ for higher values of c

- Good-Turing fails for higher c where $f_{c+1} = 0$
- use linear regression estimate for $c >$ threshold
- use regular f_c for $c <$ threshold

3.3 Hidden Markov Models

3.3.1 POS Tagging

part of speech syntactic category that tells you grammatical properties of a word

- nouns
- verbs
- adjectives
- prepositions
- adverbs
- determiners “the, a, an”

other POS

- modals and auxiliary verbs “*did* you see him?”
- conjunctions “and, or, but, yet”
- particles “look *up* and *down*”

classes of POS

- open class: where new words (**neologism**) can be added to the language
  - e.g. nouns like ”Kleenex”, adjectives like ”sick”
  - tend to be content words
- closed class: new words *tend* not to be added
  - function words that convey grammatical info

corpora

- Penn Treebank (45 tags)
- Brown corpus (87 tags)

language differences - japanese doesn’t differ between nouns and pronouns but verbs are closed class - wolof doesn’t conjugate verbs for person and tense but pronouns are - Salishan languages may not distinguish between nouns and verbs
POS tagging is a **sequence labelling** problem since it uses context

3.3.2 Markov Chains

We assume a **markov process**

- N states
- weighted transitions between states
- transition only dependent on current state

Model POS as **hidden variable** aka state, words are visible **emissions**

- independence assumption allows you to factorize
- $P(O,Q) = P(Q_1)\prod_t P(Q_{t+1}|Q_t)\prod_t P(O_t|Q_t)$
- $P(thetext) = P(DT)P(the|DT)P(NN|DT)P(text|NN)$

HMM for N tags, W words, parameters θ

- initial prob Q_1
- transition probs $a_{t,t+1} : Q_t \to Q_{t+1}$
- emission probs $b_t(O_t) : Q_t \to O_t$

Choosing the most likely POS tag for each output is naive!

- $P(O|\theta)$ computing likelihood of a sequence (forward/backward algorithm)
- $\arg\max_Q P(Q,O|\theta)$ what state sequence best explains observations (Viterbi)
- Given an observation sequence, what is the best model? (Forward-Backward, Baum-Welch, EM)

3.3.3 Forward Algorithm

$P(O|\theta) = \sum_Q P(O,Q|\theta)$
since there are N^T possible paths so use DP to avoid recalculations (see Trellis in Figure 1)

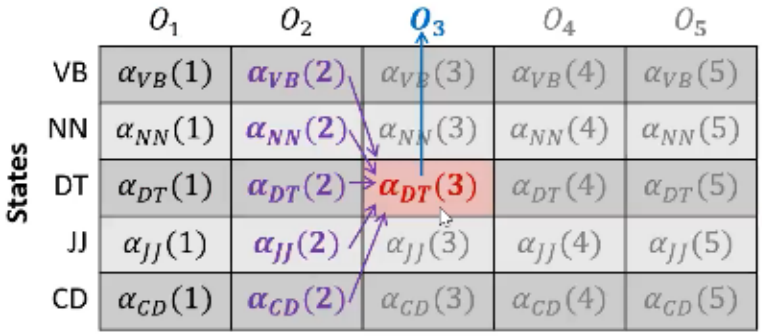

Figure 1: Trellis for Forward Algorithm

Current tag,previous words given previous tags $\alpha_i(t) = P(O_{1:t}, Q_t = i|\theta)$

- initial prob $\alpha_j(1) = \pi_j b_j(O_1)$
- recurrent sum over prev $\alpha_j(t) = \sum_i \alpha_i(t-1) a_{ij} b_j(O_t)$
- final $P(O|\theta) = \sum_j \alpha_j(T)$
- runtime $O(N^2 T)$

3.3.4 Backward Algorithm

Subsequent words given current tag $\beta_(t) = P(O_{t+1:T}|Q_t = i, \theta)$

- excludes the current word unlike α
- initial prob $\beta_j(T) = 1$
- recurrent sum over prev $\beta_i(t) = \sum_j a_{ij} b_j(O_{t+1}) \beta_j(t+1)$
- final $P(O|\theta) = \sum_i \pi_i b_i(O_i) \beta_i(1)$
- runtime $O(N^2 T)$

3.3.5 Forward-Backward

Double-check forward using backward

- $\alpha_i(t)\beta_i(t) = P(O, Q_t = i|\theta)$
- therefore $P(O|\theta) = \sum_i \alpha_i(t)\beta_i(t)$ for any $t \in [1 \dots T]$

Work in log probs for numerical stability

- $\prod p_i \to \sum \log p_i$
- $\sum p_i \to \log \sum p_i = b + \log \sum e^{a_i - b}$ where $b = \max \log p_i$

3.3.6 Viterbi

Find most likely state sequence $Q* = \arg\max_Q P(Q, O|\theta)$, like forward algorithm, but take the max

- initial $\delta_j(1) = \pi_j b_j(O_1) \forall j \in [1, N]$
- recurrence $\delta_j(t) = \max_i \delta_i(t-1) a_{ij} b_j(O_t)$
- final $\max_i \delta_i(T)$
- runtime $O(N^2 T)$

Recover best i for each T by keeping track of $\arg\max_i$ (Figure 2)

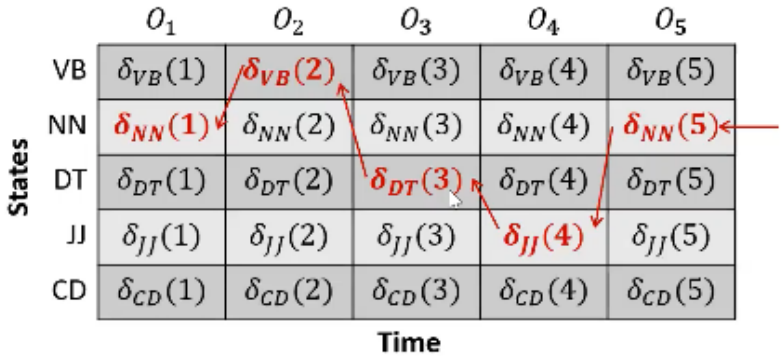

Figure 2: Viterbi backpointers

3.3.7 Baum-Welch

How to deal with unsupervised learning? **Hard EM** or Viterbi EM

1. initialize randomly
2. predict current state sequence using current model
3. update current parameters based on current predictions
4. repeat from 2

Baum-Welch or soft EM uses soft predictions

- expectation: get expected counts for hidden structures using θ_k
- maximization: find θ_{k+1} to maximize likelihood on expected counts
- finds a local optimum in $P(O|\theta)$

Expectation of **responsabilities**: prob distribution over tags

- state prob $\gamma_i(t) = P(Q_t = i|O, \theta^k) = \frac{\alpha_i(t)\beta_i(t)}{P(O|\theta^k)}$
- transition prob $\xi_{ij}(t) = P(Q_t = i, Q_{t+1} = j|O, \theta^k) = \frac{\alpha_i(t)a_{ij}b_j(O_{t+1})\beta_j(t+1)}{P(O|\theta^k)}$

Maximization of Soft MLE update

- $\pi_i^{k+1} = \gamma_i(1)$
- $a_{ij}^{k+1} = \frac{\sum_t \xi_{ij}(t)}{\sum_t \gamma_i(t)} \approx \frac{count(i,j)}{count(i)}$
- $b_i^{k+1}(w_k) = \frac{\sum_t \gamma_{ij}(t)|_{O_t=w_k}}{\sum_t \gamma_i(t)} \approx \frac{count(w_k,i)}{count(i)}$

Practical training

- stop iteration using performance on held-out set
- **random restarts** train different models from different inits
- biased initialization using some external supervised knowledge
- **semi-supervised** with small labelled data and large unlablled

3.3.8 Multiword Tasks

Similar HMM tasks can require multiple words per tag

- **Chunking** find syntactic chunks e.g. `[The chicken] [crossed] [the road]`
- **Named Entity Recognition** (NER) identify elements corresponding to high-level categories e.g. `[McGill University] is located in [Montreal, Canada]`

IOB tagging label whether word is at beginning, inside, or outside a span of tags e.g. McGill`[B-ORG]` University`[I-ORG]` is located in Montreal`[B-LOC]` Canada`[I-LOC]`

3.4 Linear-chain CRF

3.4.1 Discriminative

Shortcoming of HMMs

- adding a feature requires adding an emission e.g. word position, capitalization...

LC-CRF linear chain conditional random field

- HMM-like task-specific discriminative model $P(Y|X;\theta)$ using features, not probs
- $P(Y|X) = \frac{1}{Z(X)} \exp \sum_t \sum_k \theta_k f_k(y_t, y_{t-1}, x_t)$
- $Z(X) = \sum_y P(Y|X)$ is a normalization constant
- sum over all timesteps t, features k

Examples of features

- HMM probs e.g. emit "the" from DT $1(y_t = DT)1(x_t = "the")$
- capitalization $1(y_t =?)1(x_t \text{is capitalized})$

3.4.2 Inference

Forward algorithm computes $Z(X)$

- initial prob $\alpha_j(1) = \exp \sum_k \theta_k^{init} f_k^{init}(y_1 = j, x_1)$
- recurrent sum over prev $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp \sum_k \theta_k f_k(y_t = j, y_{t-1}, x_1)$
- final $Z(X) = \sum_j \alpha_j(T)$

Viterbi algorithm computes $\arg\max_Y P(Y|X,\theta)$

3.4.3 Training

Use gradient descent $\theta_{t+1} \leftarrow \theta_t + \alpha \nabla l(\theta)$

- no analytic solution
- $l(\theta)$ is concave
- can use conjugate gradient and L-BFGS

Gradient $\nabla l(\theta)$ = empirical count of features - expected feature count using current model

- minimized when model matches empirical distribution
- regularization $\sum_k \frac{\theta_k^2}{2\sigma^2}$
- SGD used in practice

3.5 Recurrent Neural Networks

3.5.1 Neural Networks

Feed-forward NN

- all computations flow forward

Time-delay NN

- feed context window around word into FF
- requires fixed horizon
- sequence interaction learned indirectly, not sequence model!

3.5.2 RNN

RNN: NN sequence model

- modelling long-range dependencies
- $RNN(s_0, x_{1:n}) = s_{1:n}, y_{1:n}$
- $s_i = R(s_{i-1}, x_i)$
- $y_i = O(s_i)$

comparison to LC-CRF

- LCCRF is linear, RNN is more expressive
- LCCRF uses feature engineering, RNN discovers features
- LCCRF assumes local independence, fast inference, RNN need approx (e.g. beam-search)

3.5.3 LSTM

vanishing and exploding gradient

- gradients wrt time 1 $\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial f^N} \frac{\partial f^{N-1}}{\partial f^{N-2}} \cdots \frac{\partial f^1}{\partial W^1}$
- if gradient norm < 1, it will vanish
- if gradient norm > 1, it will explode to infinity

Long Short-Term Memory (Hochreiter and Schmidhuber, 1997)

- explicitly model memory $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- forget $f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$
- input $i_t = \sigma(W_i \cdot [h_{t-1}, x_t]), \tilde{C} = \tanh(W_C \cdot [h_{t-1}, x_t])$
- output $o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$ $h_t = o_t * \tanh(C_t)$

BiLSTM, a layer for forward and backward

- concatenate $h_{forward}, h_{backward}$ to get output

LSTM-CRF, LCCRF layer on top of a BiLSTM

- using features: output scores of LSTM P, transition probabilities b/w tags (learned) A
- total score $s(X, y) = \sum_i A_{y_i, y_{i+1}} + \sum_i P_{i, y_i}$

training

1. forward pass for BiLSTM
2. forward CRF to get predictions
3. backward pass CRF to get loss
4. backprop loss through BiLSTM

3.6 Pretrained LMs

3.6.1 Transfer Learning

transfer learning using knowledge from one task to improve performance on another

- don't start task from scratch, transfer knowledge of words, syntax,...
- use language modelling as a source task

ELMo (Peters et al, 2018) large-scale LM pretrained BiLSTM

- generate contextual word embeddings
- $ELMo(w_k)$ = weighted sum of BiLSTM layers

3.6.2 Transformer

Attention is all you need (Vaswani et al, 2017)

- allow information flow between any pair of words
- Transformer architecture uses only attention, no recurrence
- $O(N^2)$ connections but better parallelism

attention, three views of a word

- query: word we want to compute in the next layer
- key: how important the word is to another word
- value: value associated with the key once attention is computed

3.6.3 Large Scale Pretrained Models

BERT - transformer encoder model

- pretraining MLM (masked language modelling) and next sentence prediction
- trained on 3B words
- 340M parameters

GPT-3 (OpenAI, 2020) - transformer decoder model

- pretraining on language modelling
- trained on 500B words
- up to 175B model parameters
- success in 0-shot and few-shot learning

Problems

- generated text can be incoherent or repetitive
- reasoning about physics + commonsense
- arguments about memorization vs understanding
- misuse of language models (spam, fake news)
- fairness, bias, cost!

4 Syntax

4.1 Form of Language

syntax words arranged to form a grammatical sentence

- generate all and exactly those sentences which are grammatical (**grammaticality**)
- **descriptive** not **prescriptive**

contituency group of words that behave as a unit, e.g for noun phrases

- can appear in similar syntactic envs
- can be replaced as a unit or rearranged
- can be used to answer a question

grammatical relations between contituents e.g.

- subject
- (direct) object e.g. he kicked *the ball*
- indirect object e.g. she gave *him* a good beating

subcategorization different number and type of args mandatory for a verb or adj

- (subj) relax
- (subj) steal (obj)
- (subj) want (obj / inf clause)
- different (from / than / to)

4.2 Formal Grammars

formal grammar rules that generate a set of strings to make up a language
finite state automata generates a regular language

- correspond to **regular grammars**
- used for stemming, lemmatization, ...

context-free grammars more powerful class of grammars for NL

- N set of **non-terminals**
- Σ of **terminals**
- R set of rules or productions
- $S \in N$ start symbol

generation

- **undergeneration** misses valid sentences
- **overgeneration** adds extra invalid sentences
- allows recursion

constituency grammars combine into bigger and bigger constituents (NP *to* Det, N)

- can be converted into dependency tree if you know the head of constituent

dependency grammar represent relations as directed edges

- each phrase has a **head** word e.g. student *studied* for the exam
- easier to extract relations
- can be converted to contituency trees if **projective** (dependecy edges don't cross, freer word order langs)

chomsky hierarchy

- regular
- context-free: can describe English/German but not others
- context-sensitive: technically Swiss German bc of cross-serial dependencies
- recursively enumerable

4.3 CYK Parsing Algorithm

4.3.1 CYK Parsing

parsing given input sentence and CFG, recover all possible parse trees

- top-down: start at S and expand (e.g. Earley)
- bottom-up: start from input words (e.g. CYK, shift-reduce)

CYK (Cocke-Younger-Kasami) bottom-up dynamic programming

- convert CFG into Chomsky Normal Form
- setup table of contituents
- fill in table
- use table

Chomsky Normal Form: $A \to B, C$ non-terminal, $A \to s$ terminal

- reduce non-terminals to 2 $A \to B, C, D$ converts $A \to X1, D$ and$X1 \to B, C$
- split non-terminal,terminal $A \to s, B$ converts $A \to X2, B$ and $X2 \to s$
- remove single non-terminals $A \to B, B \to \ldots$ converts $A \to \ldots$

constituent parse table (see Figure 3)

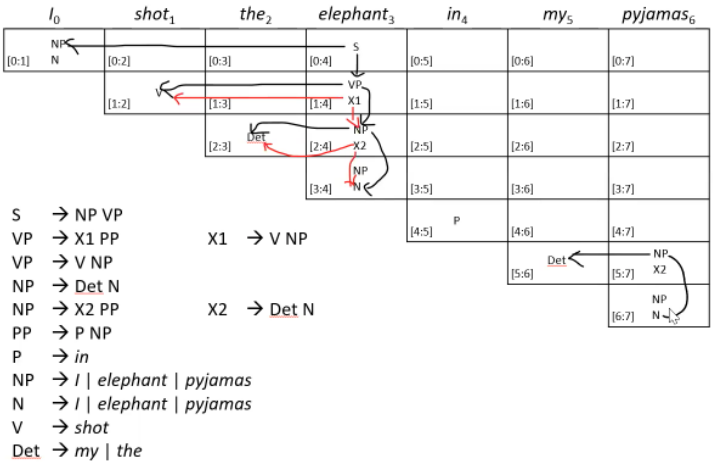

Figure 3: contituent parse table

- sentence $w[0], w[1] \dots w[N-1]$
- cell i, j corresponds to span $w[i : j+1]$
- cell lists non-terminals that can span those words

build parse table

- base: for each word i put in $c[i : i+1]$ non-terminal that generates the word
- recursive: check all break points $m \in [i, j]$ and check for rule $A[i, j] \rightarrow B[i : m], C[m : j]$
- fill in bottom-up, left-to-right

use table

- use cell $c[0 : N]$ and follow back-pointers

4.3.2 Probabilistic CYK Parsing

probabilistic CYK associate each rule with a probability, use product of probabilities for rules

- for each non-terminal $A, \sum_B Pr(A \rightarrow B) = 1$
- combining rules for tree $t, Pr(t) = \prod_i Pr(A_i \rightarrow B_i)$ with rules $A_i \rightarrow B_i$

probabilistic parsing recovers best parse $\arg\max_t Pr(t)$

- naive: run CYK and take argmax
- idea: keep track of probability in CYK and always choose highest probability parse

probabilistic CYK keeps track of probability and only most likely for each tree type

- table[2, 4, NP] = 0.215 (table[2,3,Det], table[3,4,N])
- table[3, 4, NP] = 0.022
- table[3, 4, N] = 0.04

Vanilla PCFG learn probabilities using MLE on corpus (e.g. WSJ)

- not enough context e.g. "I" "me" are NP but are object or subject
- rules are too sparse e.g $VP \to VBD$ and $VP \to VBD, PP$ are separate not factorizable
- strong *vertical* assumption, contituent independent of other parts of tree
- weak *horization* assumption, tries to model all combinations separately

4.3.3 Markovization

Improved PCFG by markovization: better context annotation and factorization

- **vertical** annotate with n-th parent NP^S likey subject, NP^{VP} likely object
- **horizontal** factorize large rules like n-gram $Pr(VP \to START, AdvP, VBD \ldots) = Pr(VP \to START, AdvP) * Pr(VP \to AdvP, VBD) \ldots$ and learn factors separately

WSJ results by Klein and Manning (2003) show large improvement 71.3 $\to$ 79.7

5 Semantics

5.1 Meaning of Language

semantics study of meaning in language

- **extensional** picks out word's **referents** in the real world
- **intensional** defines a word in terms of other words e.g. dictionary

Frege (1892) differentiates sense and reference

- **sense** of a term is its meaning, whereas
- **reference** is what it points to in the real world

5.2 Lexical Semantics

lexical semantics the meaning of words, with relations **hyponym / hypernym** hypo "is a" hyper

- class: monkey / mammal
- instance: Montreal / city

synonym same meaning **antonym** opposite meaning **homonym** same form, different unrelated meaning

- homophone: same sound
- homograph: same written form

polysem multiple related meanings **metonym** substituting one entity for related one (e.g. order a *dish* from the menu)

- synecdoche: whole-part relation (e.g. all *hands* on deck)

meronym / holonym mero "is part of" holo

- groups and members (student / class)
- whole and part (wheel / car)
- whole and substance (wood / chair)

WordNet (Miller et al, 1990) lexical resource organized by **synsets**

- hierarchies based on lexical relations

5.2.1 Word Sense Disambiguation

word sense disambiguation figure out which sense a word is expressing using contextual words
Lesk's algorithm (1986) heuristic approach

1. construct B bag-of-words rep for context
2. calculate $overlap(B, signature(s_i)$ for each candidate sense s_i
3. choose sense with highest overlap

Yarowsky's algorithm (1990) unsupervised (or minimal) based on bootstrapping

1. choose word to disambiguate (e.g. plant)
2. find two senses and label with heuristic (e.g. plant *life* vs *manufacturing*) creating *seed set*
3. repeat iteration
    - learn a supervised model on seed set
    - predict labels of whole dataset
    - keep highly confident labels as next seed set

5.2.2 Hearst Patterns

detect lexical semantic relationships in text since they tend to occur in certain **lexico-syntactic** patterns
Hearst (1992) patterns for hyper-hypo

- NP such as NP and|or NP
- NP , including NP,
- NP , especially NP,

Find new patterns by bootstrapping with known pairs → patterns → new pairs

5.3 Distributional Semantics

some semantic rels, e.g. synonyms, do not appear together but rather occur in the same context
"You shall know a word by the company it keeps" (Firth, 1957)

5.3.1 Word Vectors

Count-based word vectors

- for some word i, keep count of words that appear within a window of n words
- generate term-context matrix such as Figure 4
- compute cosine sim between word vectors to get similarity

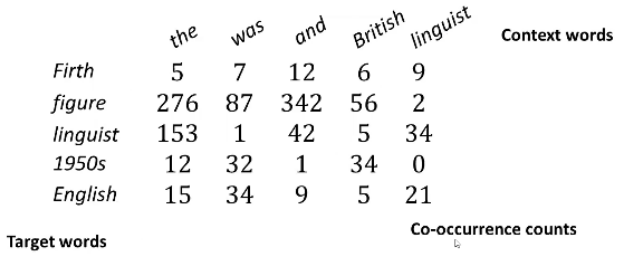

Figure 4: term-context matrix

vector similarity measures *relatedness*

- synonyms and antonyms are hard to distinguish by context
- **similarity** synonymy, hypernymy, hyponymy
- **relatedness** association, includes antonyms

evaluating word vectors is non-trivial

- compare to gold standard e.g. WS-353 (Finkelstein et al, 2002)

5.3.2 Better Word Vectors

instead of raw counts use **Pointwise MI**

- $\log \frac{P(w_1,w_2)}{P(w_1)P(w_2)}$
- normalizes chance co-occurence
- PMI can measure positive correlation > 0 or negative < 0
- **Positive PMI** sets $PMI < 0 \rightarrow PMI = 0$

5.3.3 Singular Value Decomposition

term-context matrices are sparse, compress into smaller dense
Singular Value Decomposition matrix factorization $X = W \times \Sigma \times C^T$

- new word vectors $W \in R^{|V| \times m}$
- Σ has singular values of X, arranged highest to lowest

truncated SVD take top k values in Σ

- use $W_k \times \Sigma_k$ as word vectors
- corresponds to finding smallest reconstruction error
- same as PCA, projecting to top k components

5.3.4 Word2Vec

word2vec (Mikolov et al, 2013) trains NNs to predict words in context

- vector representations known as **word embeddings**
- CBOW: use context words vec c_j to predict target word vec v_j
- Skip-gram: use target to predict context $P(w_k|w_j) = \frac{\exp c_k \cdot v_j}{\sum_{i \in |V|} \exp c_i \cdot v_j}$
- **negative sampling** to approximate denominator, separate true context from fake context

hyperparameters are key to performance - weighting important of context words - negative sampling $p^{3/4}(w)$ is better than $p(w)$ - similar changes improve other methods like SVD
widely used in NLP - cheap, easy, large pretrained - doesn't always work for task-specific

5.4 Compositional Semantics

5.4.1 Compositionality

compositionality (c) the meaning of a phrase depends on the meaning of its parts

- **non-c** e.g. idioms "kick the bucket"
- **co-c** meaning depends on composed words e.g. *red* hair vs *red* wine

c semantics derivate a good meaning of a representation from its parts

- assert a falsifiable proposition
- convey information about the world
- query about the world

5.4.2 Semantic Inference

Montague (1970) used logical formalism to represent meaning
semantic inference make explicit something that is implicit in language (Blackburn and Bos, 2003)
First-order logic

- **domain of discourse** a set of entities
- **variables** potential elements of domain x
- **predicates** maps elements to truth value $inCourse(x, y)$
- **functions** maps elements to elements $instructorOf(x) \rightarrow y$
- **logical connectives** $\neg, \vee, \wedge, \rightarrow \ldots$
- **quantifiers** $\exists, \forall$

interpretation or **model** of a FOL

- domain of discourse D
- mapping of functions to D
- mapping of predicates to True or False

5.4.3 Lambda Calculus

we need a computation procedure to convert from NL to formal logic
lambda calculus

- variable x
- $\lambda x.t$ where t is a lambda term
- ts where both are lambda terms
- **beta reduction** function application of $(\lambda x.t)s$
- note that it is **left associative** $abcd = ((ab)c)d$

construct contituents with lambda calculus

- "disdained" $\lambda x.\lambda y.disdained(y, x)$
- "disdained catnip" $\lambda y.disdained(y, catnip)$
- "Whiskers disdained catnip" $(\lambda y.disdained(y, catnip))Whiskers$

syntax-driven sem composition: augment CFG with lambda expression

- syntax $A \rightarrow a_1, \ldots a_n$ semantic attachment $\{f(\alpha_1.sem, \ldots \alpha_n.sem)\}$
- common noun $N \rightarrow student, \{\lambda x.Student(x)\}$
- proper noun $PN \rightarrow COMP550, \{\lambda x.x(COMP550)\}$ **type-raised** to be same as common
- intransitive verb $V \rightarrow rules, \{\lambda x.\exists e Rules(e) \wedge Ruler(e, x)\}$
  neo-davidsonian event semantics $Rules(x)$ uses reified event e
- composition is function application $S \rightarrow NP, VP, \{NP.sem(VP.sem)\}$

5.4.4 Quantification

quantifiers

- universal *all*, $\forall x \ldots \rightarrow \ldots$
- existential *a*, $\exists x \ldots \wedge \ldots$
- definite *the* (Russell, 1905)
  - existence $\exists x.Student(x)$
  - uniqueness (at most one) $\wedge \forall y.Student(y) \rightarrow x = y$
  - predicate $\wedge Smart(x)$

more semantic attachments

- universal $\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)$
- existential $\lambda P.\lambda Q.\exists x.P(x) \wedge Q(x)$
- adjectives $\lambda N.\lambda x.Smart(x) \wedge N(x)$

5.4.5 Quantifier Scope Ambiguity

scope ambiguity when multiple quantifier have multiple meanings e.g. "*Every* student took *a* course"

- different course $\forall x.Student(x) \rightarrow (\exists y.Course(y) \ldots)$
- same course $\exists y.Course(y) \wedge (\forall x.Student(x) \ldots)$

underspecification derive representation represents *all possible* meanings
Cooper storage (1988) associates store with each FOL

$$\begin{aligned} &\exists e.took(e) \wedge taker(e, s_1) \wedge takee(e, s_2) \\ &\qquad (\lambda Q.\forall x.Student(x) \rightarrow Q(x), 1), \\ &\qquad (\lambda Q.\exists y.Course(y) \wedge Q(y), 2) \end{aligned}$$

recover reading from cooper storage

1. select quantifier order (e.g. s_1 first)
2. do lambda abstraction for each quantifier and apply function (beta-reduce)

$$\begin{aligned} &\lambda Q.\forall x.Student(x) \rightarrow Q(x) \\ &\quad \lambda s_1 \exists e.took(e) \wedge taker(e, s_1) \wedge takee(e, s_2) \\ &\quad (\lambda Q.\exists y.Course(y) \wedge Q(y), 2) \end{aligned}$$

semantic attachment quantifier composition

- compose by putting in storage
- $NP \rightarrow Det, N$ gives $\{\lambda u.u(s_i), \text{storage:}(Det.sem(N.sem), i)\}$

6 Discourse

6.1 Language Communication

language is a *discourse*

- **monologue** one-directional communication
- **dialogue** multi-participant communication

sequential sentences must "make sense"

- **coherence** is about logical sense
- **cohesion** is about linguistic devices to flow better
  - **lexical chain** of semantically similar words e.g. The *government* proposes rules. *Citizens* are protesting.
  - **anaphoric devices** like **coreference chains** when one phrase meaning depends on another e.g. *The rules* are long. *These regulations* ...
  - **discourse markers** of cue words e.g. The dog is big. Dogs can *also* be small.

6.2 Coreference Resolution

6.2.1 Corefence and Anaphora

reference relating *mentions* to *referents* in the real world

- proper names *Montreal*
- pronouns *he, she*
- noun phrases
  - indefinite *a* deer
  - definite *the* cat
- demonstrative: points to something *this* hotdog

coreference when two mentions point to the same object

- **anaphor** points to previous Maru is a cat. *He* is grumpy
- **cataphor** points to following When *he* is grumpy, Maru doesn't eat

zero anaphora in other **pro-drop** languages allows omitting pronouns

- e.g. Spanish can figure out pronoun using verb *hablo* Español
- e.g. Japanese requires context *Ai shi* (love) te ru (prog pres) = I love you

noun phrase coreference

- **bridging** infer ref from previous mentions "I like my office. *The windows* are large"
- **pleonastic** are non-referential pronounce "*It* is raining"

- **clefting** puts focus on something (sorta referential) "*It* is my cat that is grumpy"

coreference beyond entities and noun phrases

- **event** that corefer or happen in same time / place
  "I *bought* my cat. The *adoption* was quick"
- **abstraction** shell noun (this) represents a whole thing
  "Grumpiest cats are the best. *This fact* is true"
- **cross document** alignment of events and entities from multiple sources

6.2.2 Hobbs Algorithm

cues for anaphora resolution

- number and gender
- recency
- syntactic info e.g. **binding theory** (Chomsky, 1981)
  - "the students taught *themselves*"
  - reflexives are bound by a subject in a certain way
  - personal pronouns are not

Hobbs (1978) traversal algorithm using constituent parse tree and morph analysis of number and gender

1. search the sentence right-to-left
2. use heuristics and check if NP found match in number and gender
3. if no antecedent found, search previous sentences left-to-right

6.2.3 Coreference ML

coreference subtasks

- **mention detection** decide which text are mentions and anaphoric
- **coreference resolution** determine coref links in passage

Soon et al (2001) defined 12 features and used decision tree

- string overlap
- pronoun
- wordnet class
- number agreement
- gender agreement
- distance (recency)

Ng and Cardie (2002) extended the feature set, Durrett and Klein (2013) incorporated even more features into log-linear model
Lee et al (2017) supervised end-to-end neural coref

- $s_m(i)$ mention score of i
- $s_a(i, j)$ score of j antecedent of i
- score all possible pairs, prune low-prob mentions

6.3 Coherence Modelling

theories of coherence

- discourse relations b/w text spans e.g. **rhetorical structure**
- **local coherence** links between text spans
- **global coherence** structure of entire discourse

6.3.1 Rhetorical Strucutre Theory

Mann and Thomson (1988) describe discourse structure as tree e.g. Figure 5

1. segement text into **EDU** elementary discourse units
2. relate spans according to **rhetorical relations**

rhetorical relations (nucleus-sattelite)

- elaboration
- attribution (gives source of information)
- contrast
- list (without explicit contrast or comparison)
- background info

relation components

- **nucleus-sattelite** for asymmetric
- **nucleus-nucleus** for symmetric

applications using RST Tree features

- essay grading
- automatic summarization

bootstrapping

- gather discourse cues "consequently" $\rightarrow RESULT$
- supervised learn the discourse value

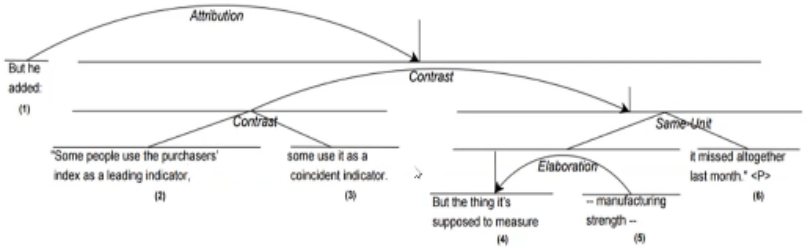

Figure 5: RST Tree from Joy et al. (2013)

6.3.2 Local Coherence Modelling

LCM (Barzilay and Lapata, 2005) looks at local cohesives in adjacent sentences
Generally, entity mentions follow patterns

- first mention is often subject "Maru is . . . "
- mentions are clustered together ". . . Maru is grumpy. He likes cat food. . . . (no more mentions)"

entity grid (Barzilay and Lapata, 2008)

- plots entity mentions and syntactic role (see Figure 6)
- extract features of document using relative freq of entity mention transition e.g. prob of subject → subject
- use to model ordering of a document, summary coherence, readability

	Department	Trial	Microsoft	Evidence	Competitors	Markets	Products	Brands	Case	Netscape	Software	Tactics	Government	Suit	Earnings	
1	s	o	s	x	o	–	–	–	–	–	–	–	–	–	–	1
2	–	–	o	–	–	x	s	o	–	–	–	–	–	–	–	2
3	–	–	s	o	–	–	–	–	s	o	o	–	–	–	–	3
4	–	–	s	–	–	–	–	–	–	–	–	s	–	–	–	4
5	–	–	–	–	–	–	–	–	–	–	–	–	s	o	–	5
6	–	x	s	–	–	–	–	–	–	–	–	–	–	–	o	6

Figure 6: Entity grid across 6 sentences (Bazilay and Lapata, 2008) with grammatical roles subject (s), object (o), or neither (x)

entity grid extensions

- combining global and local coherence (Elsner et al, 2007)
- other languages (Cheung and Penn, 2010)

- neural coherence (Nguyen and Joty, 2017)
- self-supervised (Xu et al, 2019)
  - encode consecutive sentences s, t
  - learn score using s, t features
  - SSL with NCE