Level of Languages

Phonetics/phonology/morphology:
- What words or subwords we are dealing with
- ex. cats, dogs, boxes

Semantics
- Literal meaning of the sentense
- ex. Papa eats caviar

  Papa $\xleftarrow{\text{agent}}$ eat $\xrightarrow{\text{agent}}$ caviar

  relationship + definitions

Syntax
- What phrases are we dealing with? which words modify one another
- ex. Subject + verb + Noun

Pragmatics
- What should you conclude from the sentense
- How should you reac
- ex. Can you open the fridge? - Tejas
  No - Tony

Text Classification (sentiment Analysis)

Rule-based Classifier
- make class decision based on if...else rules
- ex. pos/neg wordlist, +pts for pos, -pts for neg
- Problems:
  - Coverage - some records not covered by curr rules
  - negation - "not bad" ≠ "good"
  - word composition
  - domain differences
  - new words / concept
  - word sense ambiguity

Supervised Classification
- Data-driven approach
- (text, label) pair, learn a model

Probabilistic Classifier
- Learn a probabilistic for $p(y|x)$

Approach 1: Direct MLE estimation of $p(y|x)$
- will not work cuz the space of X is exponentially too large an unlimited

Approach 2: Model Assumptions
1. Bag of Word - assume order of words doesn't matter
   - represented as unordered words + frequency
   - reduce computational complexity
   - does not model sequence
2. Naive Bayes assumption - words are independent conditioned in their class
   - $P(A\,B\,C|Y) = P(A|Y) \times P(B|Y) \times P(C|Y)$
   - assume words are independent of each other
   - reduce computational complexity

   Bayes Theorem for $P(Y|X)$

   $$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Learn Probabilistic Model Complexity
- K labels - d output dimension
- V vocabs
- $P(x_1, x_2 \cap, x_d | y)$ require $k(V^d - 1)$ params

After Naive Bayes Assumption:
- $P(x_1, x_2, \cap, x_d|y) = P(x_1|y) \times \dots \times P(x_d|y)$
- $k(V-1)$ params

Naive Bayes Classifier
$P(Y=y_i) = \text{Count}(y_i)/\text{Count}(Y)$

$P(x|y_i) = \frac{\text{Count}(W=x,\ y_i)}{\text{Count}(W,\ y_i)}$ * count of all words in y_i category (w/ dup)

$h_{NB}(x) = \arg\max_y P(y) \prod_j P(x_j|y)$

Steps:
1. Prior from Training: get $P(Y=y_i)$
2. Drop unseen word from test sentence?
3. Likelihood from training: get $P(x|y_i)$
4. Scoring Test set: $P(y_i)P(x|y_i)$

Discriminative Models vs. Generative Models

Discriminative Models	Generative Models
- Learn decision boundary	- learn the input dist
- Maximize cond prob: $P(y|x)$	- Maximize Joint: $P(y,x)$
- Directly estimates $P(Y|X)$	- Use $P(X|Y)$ + Bayes' rule to find $P(Y|X)$
- Cannot generate new data	- Can generate new data
- Used for classification	- Not used for classification
- don't possess generative properties	- Possess discriminative properties

Logistic Regression
Maximize: $\prod_{(x^i,y^i)\in D} \frac{\exp(W^T\phi(x^i,y^i))}{\sum_{y'\in Y}\exp(W^T\phi(x^i,y'))}$

use log $= \arg\max_w \sum_{i=1}^n \log \frac{\exp(W^T\phi(x^i,y^i))}{\sum_{y'\in Y}\exp(W^T\phi(x^i,y'))}$

loss: $L_{logreg} = -W^T\phi(x,y) + \log \sum_{y'\in Y}\exp(W^T\phi(x,y'))$

$\frac{\partial}{\partial W} L_{logreg} = -\phi(x,y) + \sum_{y'\in Y} P(y'|x)\phi(x,y')$

Naive Bayes vs. Logistic Regression

Naive Bayes	Logistic Regression
- generative	- discriminative
- close form soln	- iterative soln
- independent assump	- no indep. assum.
- ez to write	- not ez to write
- fast to learn	- slow to learn
- can overfit	

Lexical Semantics

How to represent words
Naive:
- atomic symbols (BoW)
- One-hot vector: [0,0,0,1]
- issue: large dimensions, sparse vector
  : No similarity, all orthogonal
  : cannot represent new words

Two classes of algorithm for lexical semantics

Thesaurus-based:
- are words nearby in a thesaurus hierarchy
- do words have similar definition?

WordNet
Lemma - a rep. of all forms w/ same sys, part of speech, rough semantics

Wordform - as it appear in text

Ex. Wordform = sung, singing, Lemma = sing

Sense - a discrete representation of an aspect of a word's meaning
- a word can have mult. senses

Homonymy: words that share a form (spelling or pronounciation) but have unrelated distinct meanings:
- Homographs: bank/bank
- Homophones: piece/peace

Polysemy: related multi-sense, word with related meanings
- ex. serve breakfast, serve a state

Synonyms: Words w/ diff. forms have same meanings in some or all context
- relation btn senses

Antonyms: senses are opposites w/ respect to 1 feature of meaning

Hyponymy: the sense is a subclass of another

Hypernymy: the sense is a superclass of another

Meronymy: the sense is a part of another

Holonymy: the other sense is a part of this sense

Sense defined in Wordnet:
- synonym set share the same sense
- Hypernym Hierarchy

Path Similarity
- Path length(c_1,c_2) = 1 + # edge in hypernym graph between c_1 and c_2
- simpath$(c_1,c_2) = \frac{1}{\text{len}(c_1,c_2)}$
- Wordsim$(a,b) = \max_{c_1 \in \text{senses}(w_1),\, c_2 \in \text{senses}(w_2)}$ simpath(c_1,c_2)

Thesaurus limitation:
- source dependent (missing new concepts)
- Limited in scope (IS-A relationship, best for noun)
- No context
- not domain adaptable
- not available in many languages

Distributional Based:
Sparse vector Representations:
1. Mutual info weighted word co-occurrence matrices

Dense Vector Representation

2. Singular value decomposition (latent semantic Analysis)
3. NN inspired models (skip-grams, CBOW)

Term document matrix
Column: document ⇒ each column is a count vector
row : word

Problem:
- Doc can be very long
- Some far away words in the same document are no longer relevant/related
- usually small amount of document
- small dimension for each word
- less robust/reliable

Word-Context/word-word
- use smaller contexts (paragraph, windows)
- word defined by a vector over counts of context words
- very sparse = most values are 0
- shorter window = more syntactic representation
- longer window = semantic representation (topical if longer)

Problem:
- raw word count is not a great measure of association

Pointwise Mutual Information
$$PMI(\text{Word1}, \text{Word2}) = \log_2 \frac{P(\text{Word1}, \text{Word2})}{P(\text{Word1})P(\text{Word2})}$$

$$PPMI(\text{Word1}, \text{Word2}) = \max\left(\log_2 \frac{P(\text{word1}, \text{word2})}{P(\text{word1})P(\text{word2})}, 0\right)$$

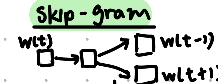
Low-dimensional Representation
Problem with W-D and W-W matrix
- Num of basis concept is large
- Basis are not orthogonal
- Some words too frequent (the)

Latent Semantic Analysis
Factorization - Apply SVD to the matrix to find latent component
- Uncover relation not explicit in the corpora
- term vectors projected to k dim latent space

Word2Vec
- LSA: a compact/low dim representation of co-ocurrence matrix
- Prediction-based model: another way to get dense vectors
- Ez to add new words or sentence
- Train a NN to predict neighboring words
  - less dense embeddings for words in train corpus
- Advantages
  - Fast, ez to train, pretrained

CBOW


use neighbor words to predict a word

Skip-gram


use a word to predict neighboring word

- We want low-dimensional vector rep for words
- word rep as vectors, initially randomized

Skip Gram
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-m\le j\le m,\, j\ne 0} \log p(w_{t+j}|w_t)$$

maximize log likelihood of context word to give center word

$$p(w_{t+j}|w_t) = \frac{\exp(u_{w_{t+j}} \cdot v_{w_t})}{\sum_{w'\in V}\exp(u_{w'} \cdot v_{w_t})}$$

v_w: when w is the center word (input embed)

u_w: when w is the context word (output embed)

Ngrams (not a perfect model)
- a contiguous sequence of n tokens from a given piece of text. Params Size = $|V|^n$
- models $p(x_{\{t:t+n\}})$ or $p(x_{t+n}|x_{\{t:t+n-1\}})$
- To capture beginning behavior of sequence add n-1 <bos> to the front
- .... end behavior of sequence add <eos>

$P(w) = \prod_{i=1}^{n+1} P(w_i|w_{i-2}\,w_{i-1})$ (tri-gram)
- use log addition in practice bc mult normally lead to small prob

Evaluation:
Extrinsic - measure perform. on downstream app
- most useful eval, plug into downstream sys
- time consuming + need eval metrics

Intrinsic - measure designed for curr task
- easier, faster
- not ez to figure out good measurement

loss: $\prod_{i=1}^{N} P(w_i|\dots)$

cross entropy: $\frac{-\sum_{i=1}^{N}\log_2(P(w_i|\dots))}{N}$

Perplexity: $2^{\text{cross entropy}}$
- how surprised is the model?
- smaller = better

Estimate Sentence Probability
type = distinct vocab items

token = occurrence of type

Independent Assumption for unseen sentence

ngram independent assump.

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

$P_{MLE}(w_i|w_{i-n}\dots w_{i-1}) = \frac{\text{Count}(w_{i-n},\dots,w_i)}{\text{Count}(w_{i-n}\dots w_{i-1})}$

Smoothing
- discount positive counts and relocate to unseen words
- **Add one smoothing**
  - add 1 to the count of word
  - add V to denominator
  - Problem: allocate too much prob to unseen words, making the model think that high chance of novel event when large dictionary
- **Add Lambda Smoothing**
  - add λ to all counts, adjust λ to get best result
  - add λV to denominator

K-fold Dumb down:
- divide training dataset into k section
- Use 1 of k section as dev to determine λ
- evaluate on rest of k-1 sections
- pick best λ and test on test dataset
+ assess λ on all of training set
+ test λ on all of training set
+ fast (retrain on sentence w/ 1 word diff)

Backoff Smoothing
- Consider backoff prob (lower order)
- ex. unigram, bigram ..
- Holds out same prob for novel events but divide up unevenly in proportion to backoff prob
- $P_{avg}(z|xy) = \mu_3 P(z|xy) + \mu_2 P(z|y) + \mu_1 P(z)$

Log-Linear and Neural LM
- Convert linear scoring function to $p(y|x)$
- K features

$score(x,y) = \sum_k \theta_k f_k(x,y)$

$Z(x) = \sum_{y'} \exp score(x,y')$ → used for normalized softmax

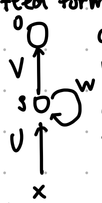
$P_{\vec\theta}(y|x) = \frac{1}{Z(x)} \exp(score(x,y))$

Maximize log: $\sum_{i=1}^{n} \log p_{\vec\theta}(y_i|x_i)$

gradient: $\nabla_{\vec\theta} \log P_{\vec\theta}(y|x) = \nabla_{\vec\theta} score(x,y) - \nabla_{\vec\theta} \log Z$
$= \vec f(x,y) - \sum_{y'} P_{\vec\theta}(y'|x) \vec f(x,y')$

Neural language Model
- help generalize unseen contexts
- linear transformation + activation function
- Forward Pass - store intermediate result for ez gradient calc
- Back propogation
  - Compute local gradient
  - Combine w/ upstream grad for full grad

RNN
- make use of sequential information, while feed forward model assume independent



$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$ - lin comb of curr input and past mem

$h^{(t)} = \tanh(a^{(t)})$ - activation func

$o^{(t)} = c + Vh^{(t)}$ - lin trans convert mem to output mem

$\hat y^{(t)} = softmax(o^{(t)})$ - dim = $|V|$
- prob of next words in sequence

- incapable of storing long term dependency IN PRACTICE (vanishing grad)
- hard to know which past information to store

Long Short Term Memory (LSTM)
- designed to capture long-term dependency
- Solve vanishing gradient
- memory cell state

Steps:

Input gate - decide what info used from curr input and store in cell state
1. sigmoid layer - decide what val we'd update
2. tanh layer - create a vector of new candidate value $\tilde C_t$

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

Forget gate - decide what info we don't need
- look at h_{t-1} and x_t and output 0-1
  - 0 = get rid of completely
  - 1 = keep it completely

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

Next Step:
- update old state by C_{t-1} into new C_t
- mult old state by f_t - forget
- add $i_t * \tilde C_t$ - add new info

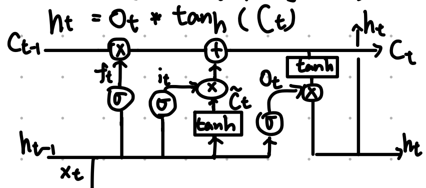
$C_t = f_t * C_{t-1} + i_t * \tilde C_t$

Final Step:

Output gate - what to output

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

$h_t = o_t * \tanh(C_t)$



RNN Problems:
- no parallel computation
- vanishing gradient still exist for LSTM

Transformers
- Use attention to measure dependencies

Encoder:
$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$

layer 1: self attention

layer 2: feedforward NN

Decoder:
$h_t = f(W^{(hh)} h_{t-1})$

layer 1: Self Attention

layer 2: Encoder-Decoder Attention

layer 3: Feed Forward NN

RNN Attention
$a_{ts} = \frac{\exp(score(h_t, \bar h_s))}{\sum_{s'=1}^{S} \exp(score(h_t, \bar h_{s'}))}$ attention weight

$c_t = \sum_s a_{ts} \bar h_s$ context vector

$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t])$

Self Attention:
- look at other position in input sequence to generate better encoding

Step 1:

query = $W^Q x$, target

key = $W^K x$, offer

> compute compatability

value = $W^V x$

Step 2:

$Q \times K$ - first word will query each other word based on keys to decide attention

Step 3-4: scale and softmax
- scale normalize wrt the query/key vectors dimension
- softmax gives attention weights of val

$Softmax\left(\frac{Q \times K}{\sqrt{d_k}}\right)$

Step 5: mult each val vec by attention

Step 6: sum weighted value vec

Single → Multi-head
+ expands model's ability to focus on diff pos
+ give attention layer multiple rep subspace

Added Steps:
- train attention on z_i multiple heads indep.ly
- Final: use W_o to multiply $\begin{bmatrix} z_0 \\ \vdots \\ z_n \end{bmatrix}$ by

Positional Encoding
t = pos   i = dimen

$\vec p_t^{(i)} = f(t)^{(i)} = \begin{cases} \sin(w_k t) & \text{if } i = 2k \\ \cos(w_k t) & \text{if } i = 2k+1 \end{cases}$

$w_k = \frac{1}{10000^{2k/d}}$

Residual Connections and Layer Norm
- Issue of information loss - self attention can decide not to attend to itself
- add input back after self attention

LayerNorm(x + Sublayer(x))
↳ after self attention

$LayerNorm(v) = \gamma \frac{v - \mu}{\sigma} + \beta$

F1: $2 \frac{precision \cdot recall}{precision + recall}$

Accuracy: $\frac{tp + tn}{tp + tn + fp + fn}$

Precision: $\frac{tp}{tp + fp}$

Recall: $\frac{tp}{tp + fn}$

Softmax:

$\sigma(\vec z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

$\frac{\partial}{\partial z} \sigma = \sigma(1 - \sigma)$