Mid-term Reviews

Preprocessing,
language models
Sequence models,
Syntactic Parsing
Preprocessing

• What is a Lemma? What is a wordform?
• What is a word type? What is a token?
• What is tokenization?
• What is lemmatization?
• What is stemming?
How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss’s *cat* in the hat is different from other *cats*!
  - **Lemma**: same stem, part of speech, rough word sense
    - *cat* and *cats* = same lemma
  - **Wordform**: the full inflected surface form
    - *cat* and *cats* = different wordforms
How many words?

they lay back on the San Francisco grass and looked at the stars and their

• **Type**: an element of the vocabulary.
• **Token**: an instance of that type in running text.
• How many?
  • 15 tokens (or 14)
  • 13 types (or 12) (or 11?)
Issues in Tokenization

- Finland’s capital → Finland Finlands Finland’s ?
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??
Lemmatization

• Reduce inflections or variant forms to base form
  • am, are, is → be
  • car, cars, car's, cars' → car
  • the boy's cars are different colors → the boy car be different color
• Lemmatization: have to find correct dictionary headword form

Context dependent. for instance: in our last meeting (noun, meeting). We're meeting (verb, meet) tomorrow.
Stemming

• Reduce terms to their stems in information retrieval
• **Stemming** is crude chopping of affixes
  • language dependent
  • e.g., *automate(s), automatic, automation* all reduced to *automat*.

*for example compressed and compression are both accepted as equivalent to compress.*

*for exampl compress and compress ar both accept as equivel to compress*
Naïve Bayes

• How to train a naïve bayes model? How to estimate prior probabilities and conditional probabilities?
• How to apply laplace smoothing?
Bayes’ Rule Applied to Documents and Classes

• For a document $d$ and a class $c$

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$
Learning the Multinomial Naïve Bayes Model

• First attempt: maximum likelihood estimates
  • simply use the frequencies in the data

\[
\hat{P}(c_j) = \frac{\text{doccount}(C = c_j)}{N_{\text{doc}}}
\]

\[
\hat{P}(w_i \mid c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}
\]
Laplace (add-1) smoothing: unknown words

Add one extra word to the vocabulary, the “unknown word” \( w_u \)

\[
\hat{P}(w_u \mid c) = \frac{\text{count}(w_u, c) + 1}{\left( \sum_{w \in V} \text{count}(w, c) \right) + |V + 1|} = \frac{1}{\left( \sum_{w \in V} \text{count}(w, c) \right) + |V + 1|}
\]
Maxent and Perceptron

• What are the differences between a generative model and a discriminate model?
• What are features in a discriminate model?
• What’s the relation between maxent and logistic regression?
• What’s the general form of maxent?
• What’s the form of a perceptron classifier?
Joint vs. Conditional Models

• We have some data \{(d, c)\} of paired observations \(d\) and hidden classes \(c\).

• **Joint (generative) models** place probabilities over both observed data and the hidden stuff (gene-rate the observed data from hidden stuff): $P(c, d)$
  
  • All the classic StatNLP models:
    • \(n\)-gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models
Joint vs. Conditional Models

- **Discriminative (conditional) models** take the data as given, and put a probability over hidden structure given the data:
  - Logistic regression, conditional loglinear or maximum entropy models, conditional random fields
  - Also, SVMs, (averaged) perceptron, etc. are discriminative classifiers (but not directly probabilistic)
Features

• In NLP uses, usually a feature specifies
  1. an indicator function – a yes/no boolean matching function – of properties of the input and
  2. a particular class

\[ f_i(c, d) \equiv [\Phi(d) \land c = c_j] \]  
[Value is 0 or 1]

• Each feature picks out a data subset and suggests a label for it
Feature-Based Linear Classifiers

• Exponential (log-linear, maxent, logistic, Gibbs) models:
  • Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c, d)$

$$P(c \mid d, \lambda) = \frac{\exp \sum \lambda_i f_i(c, d)}{\sum \exp \sum \lambda_i f_i(c', d)}$$

- $P(\text{LOCATION} \mid \text{in Québec}) = \frac{e^{1.8} e^{-0.6}}{(e^{1.8} e^{-0.6} + e^{0.3} + e^{0})} = 0.586$
- $P(\text{DRUG} \mid \text{in Québec}) = \frac{e^{0.3}}{(e^{1.8} e^{-0.6} + e^{0.3} + e^{0})} = 0.238$
- $P(\text{PERSON} \mid \text{in Québec}) = \frac{e^{0}}{(e^{1.8} e^{-0.6} + e^{0.3} + e^{0})} = 0.176$

• The weights are the parameters of the probability model, combined via a “soft max” function

$\sum \lambda_i f_i(c, d)$
Perceptron Algorithm

- Algorithm is Very similar to logistic regression
- Not exactly computing gradients

```
Initialize weight vector w = 0
Loop for K iterations
    Loop For all training examples x_i
        if sign(w * x_i) != y_i
            w += (y_i - sign(w * x_i)) * x_i
```
Language Modeling

• How to calculate the probability of a sentence using a language model?
• What are the main Smoothing Algorithms for language models?
• Extrinsic v.s Intrinsic Evaluation
• Intrinsic Evaluation Metric of language models
Bigram estimates of sentence probabilities

\[
P(<s> \text{I want english food } </s>) = \\
P(\text{I}|<s>) \times P(\text{want}|\text{I}) \times P(\text{english}|\text{want}) \times P(\text{food}|\text{english}) \times P(</s>|\text{food}) \\
= .000031
\]
An example

\[ P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \]

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

\[
P(I \mid <s>) = \frac{2}{3} = .67 \quad P(\text{Sam} \mid <s>) = \frac{1}{3} = .33 \quad P(\text{am} \mid I) = \frac{2}{3} = .67
\]

\[
P(<s> \mid \text{Sam}) = \frac{1}{2} = 0.5 \quad P(\text{Sam} \mid <s>) = \frac{1}{2} = .5 \quad P(\text{do} \mid I) = \frac{1}{3} = .33
\]
Backoff and Interpolation

• Sometimes it helps to use **less** context
  • Condition on less context for contexts you haven’t learned much about

• **Backoff:**
  • use trigram if you have good evidence,
  • otherwise bigram, otherwise unigram

• **Interpolation:**
  • mix unigram, bigram, trigram

• Interpolation works better
Advanced smoothing algorithms

• Intuition used by many smoothing algorithms
  • Good-Turing
  • Kneser-Ney

• Use the count of things we’ve seen
  • to help estimate the count of things we’ve never seen
Kneser-Ney Smoothing I (smart backoff)

• Better estimate for probabilities of lower-order unigrams!
  • Shannon game: *I can’t see without my reading________*?
  • “Francisco” is more common than “glasses”
  • ... but “Francisco” always follows “San”

• Instead of \( P(w) \): “How likely is \( w \)”

• \( P_{\text{continuation}}(w) \): “How likely is \( w \) to appear as a novel continuation?"
  • For each word, count the number of unique bigram types it completes
  • Every bigram type was a novel continuation the first time it was seen

\[
P_{\text{CONTINUATION}}(w) \propto \left| \{w_{i-1} : c(w_{i-1}, w) > 0\} \right|
\]
Extrinsic evaluation of N-gram models

• Best evaluation for comparing models A and B
  • Put each model in a task
    • spelling corrector, speech recognizer, MT system
  • Run the task, get an accuracy for A and for B
    • How many misspelled words corrected properly
    • How many words translated correctly
  • Compare accuracy for A and B
Perplexity

The best language model is one that best predicts an unseen test set

• Gives the highest \( P(\text{sentence}) \)

Perplexity is the inverse probability of the test set, normalized by the number of words:

\[
PP(W) = P(w_1w_2...w_N)^{-\frac{1}{N}}
\]

\[
= \sqrt[N]{\frac{1}{P(w_1w_2...w_N)}}
\]

Chain rule:

For bigrams:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1...w_{i-1})}}
\]
Sequence Tagging

• What is sequence tagging? what are common sequence tagging problems in NLP?
• What is the form of Trigram HMM?
• What’s the run time complexity of the viterbi algorithm for Trigram HMM?
Part-of-Speech Tagging

**INPUT:**
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**OUTPUT:**
Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./

N  = Noun
V  = Verb
P  = Preposition
Adv = Adverb
Adj = Adjective
...

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location
...
Why the Name?

\[ p(x_1 \ldots x_n, y_1 \ldots y_n) = q(\text{STOP} | y_{n-1}, y_n) \prod_{j=1}^{n} q(y_j | y_{j-2}, y_{j-1}) \]

\[ \times \prod_{j=1}^{n} e(x_j | y_j) \]

Markov Chain

\[ x_j \text{'s are observed} \]
The Viterbi Algorithm with Backpointers

**Input:** a sentence $x_1 \ldots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.

**Initialization:** Set $\pi(0, *, *) = 1$

**Definition:** $S_{-1} = S_0 = \{ * \}$, $S_k = S$ for $k \in \{1 \ldots n\}$

**Algorithm:**

- For $k = 1 \ldots n$,
  - For $u \in S_{k-1}$, $v \in S_k$,
    $$\pi(k, u, v) = \max_{w \in S_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$
    $$bp(k, u, v) = \text{arg max}_{w \in S_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$
  - Set $(y_{n-1}, y_n) = \text{arg max}_{(u,v)} (\pi(n, u, v) \times q(\text{STOP}|u, v))$
  - For $k = (n-2) \ldots 1$, $y_k = bp(k+2, y_{k+1}, y_{k+2})$

- **Return** the tag sequence $y_1 \ldots y_n$
The Viterbi Algorithm: Running Time

- $O(n|S|^3)$ time to calculate $q(s|u, v) \times e(x_k|s)$ for all $k, s, u, v$.

- $n|S|^2$ entries in $\pi$ to be filled in.

- $O(|S|)$ time to fill in one entry

- $\Rightarrow O(n|S|^3)$ time in total
Syntactic Parsing

• What’s a PCFG?
• What’s the probability of a parse tree under a PCFG?
• What’s the Chomsky normal form of CFG?
• What’s the run time complexity of the CKY algorithm?
### A Probabilistic Context-Free Grammar (PCFG)

<table>
<thead>
<tr>
<th></th>
<th>( \Rightarrow )</th>
<th>( \Rightarrow )</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>NP</td>
<td>VP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP</td>
<td>Vi</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>Vt</td>
<td>NP</td>
<td>0.4</td>
</tr>
<tr>
<td>VP</td>
<td>VP</td>
<td>PP</td>
<td>0.2</td>
</tr>
<tr>
<td>NP</td>
<td>DT</td>
<td>NN</td>
<td>0.3</td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>PP</td>
<td>0.7</td>
</tr>
<tr>
<td>PP</td>
<td>P</td>
<td>NP</td>
<td>1.0</td>
</tr>
<tr>
<td>Vi</td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Vt</td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td>0.7</td>
</tr>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Probability of a tree \( t \) with rules

\[
\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n
\]

is \( p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i) \) where \( q(\alpha \rightarrow \beta) \) is the probability for rule \( \alpha \rightarrow \beta \).
Chomsky Normal Form

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- $N$ is a set of non-terminal symbols
- $\Sigma$ is a set of terminal symbols
- $R$ is a set of rules which take one of two forms:
  - $X \rightarrow Y_1Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
  - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol
The Full Dynamic Programming Algorithm

Input: a sentence $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.

Initialization:
For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} 
q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\
0 & \text{otherwise}
\end{cases}$$

Algorithm:

- For $l = 1 \ldots (n - 1)$
  - For $i = 1 \ldots (n - l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate
      $$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$
      
      and
      $$bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

What’s the run time Complexity?
Dependency Parsing

• Can you draw a dependency parse tree for a simple sentence?
• What is projectivity?
Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies.

The arrow connects a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate).

Usually, dependencies form a tree (connected, acyclic, single-head).

Diagram:

```
Bills
  |_________
  |           __________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 submitted          auxpass
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep          by
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 nn
  |_________
  |           __________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 nn
  |_________
  |           __________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 nn
  |_________
  |           __________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep
  |__________________
  |          /          \
  |         /           \
  |        /            \
  |       /             \
  |      /              \
  |     /               \
  |    /                \
  |   /                 \
  |  /                  \
  | /                   \
  |/                    \
 prep
```
Projectivity

- Dependencies from a CFG tree using heads, must be **projective**
  - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- But dependency theory normally does allow non-projective structures to account for displaced constituents
  - You can’t easily get the semantics of certain constructions right without these nonprojective dependencies

Who did Bill buy the coffee from yesterday?