Speech Processing 11-492/18-492

Speech Recognition
Language Modeling
But not just acoustics

• But not all phones are equi-probable
• Find word sequences that maximizes
  \[ P(W \mid O) \]

• Using Bayes’ Law
  \[ P(W)P(O \mid W) \]
  \[ \frac{P(O)}{P(W)} \]

• Combine models
  – Use HMMs to provide
  \[ P(O \mid W) \]
  \[ P(W) \]
  – Use language model to provide
Language Predictions

- **What are the most likely words?**
  - “the” more common than “loom”

- **Different domains, different distributions**
  - Bus, timetable, 4:15, late
  - LCD, storage card, usb

- **Context helps prediction**
  - Carnegie …
  - President …
  - As quiet as a …
Look at n-gram models

- Unigram: $W_f$
- Bigram: $\{W_1 \mid W_{n-1}\}$
- Trigram: $\{W_1 \mid W_{n-1}, W_{n-3}\}$
- N-gram: $\{W_1 \mid W_{n-1}, \ldots\}$

But need lots of data to train
What is the word distribution

- Total 22.5M word tokens
- Total 508K different word types
- 15K types appear more than 100 times
- 45% types appear only once.
- Top: the, of, to, a, in, and, that, for, is, on
- said(16), Mr(17), million(24), company(39)
News tokens per day

News words per day (WSJ1995)

Y x 10^3

X
As we increase the N-gram
   • We need much more data

Vocabulary of 50K words 125T trigrams
   • At least 40T words (if equi-probable)
   • About 5000 years of WSJ
Simplifying Assumptions

- **Limit vocabulary**
  - < 64K

- **Make them all UPPERCASE**

- **Remove punctuation**
  - People don’t say punctuation
  - Maybe make into phrases at punctuation

- **Have a “unknown word” token**
  - Replace all low frequency words with UNK

- **Collapse similar words**
  - All numbers to NUM
  - Call Cities to CITY ….
Still not enough data

- **Backoff:**
  - If no trigram data use bigram data
  - If no bigram data use unigram

- **Smoothing:**
  - Assume there is at least 1 occurrences
  - Allow non-integer frequencies

- “Good-Turing” smoothing
  - If \( \text{Numof}(n\text{-1gram}) < \text{threshold} \)
    \[
    F(n\text{gram}) = \text{Numof}(n\text{-1gram}) \times P(n\text{-1gram})
    \]
You build a language model

How good is it:

- Test it in the ASR (takes time)
- Have abstract measure
Entropy and Perplexity

• Entropy

\[ H = -\frac{1}{Q} \sum_{i=1}^{Q} P(w_i|w_{i-1}, ... w_{i-N+1}) \log P(w_i|w_{i-1}, ... w_{i-N+1}) \]

- Related to predictability
- Q is number of words
- N is order of ngram

• For sufficiently large Q

\[ H = -\frac{1}{Q} \sum_{i=1}^{Q} \log P(w_i|w_{i-1}, ... w_{i-N+1}) \]

• Perplexity

\[ B = 2^H \]
Larger number, harder problem
- Sort of an average branching factor
- If 20, about 20 choices per word
- If 300, about 300 choices per word

20 is typically an “easy” task

300 is typically an “hard” task

Sometimes its only sometimes hard
- I want to go to X.

Lower perplexity measures give better recognition
- Not true, but there is a correlation
But surely we can do better

- Just using the last two words?
- Syntax, semantics ...
- Writing grammars is hard
  - Beyond simple tasks
- Training grammars is even harder
- Semantics is even harder than that
Some LM improvements

- **Looking at more than previous two words**
- **Replace words with types**
  - *I want to go from City to City*
- **Trigger-based models**
  - *If you see a word you’ll likely see related ones*
  - *“president” triggers “vice-president”*
Model Combination

- Use background model
  - General (for domain)
- Use specific model to adapt
- Combination by
  - Simple linear weights
  - Maximum Entropy
  - CART
- **Switch LM in dialog system**
- **Build separate models from different states**
  - **State1**: Where do you want to go to?
  - **State2**: When do you want to leave?
  - **State3**: When do you want to arrive?
What about OOVs?

- **OOV “out of vocabulary”**
  - *Words not in the lexicon*

- **Ignore them**
  - *They might be irrelevant*

- **Try to recognize them**
  - *They might be names*

- **Avoid them**
  - *Design your system so there aren’t any important ones*
Summary

- **Language Models**
  - Bayes equation
- **N-grams**
- **Smoothing, backoff, adaptation**