11-751
Speech Recognition and Understanding

Language Modeling 1

Florian Metze

October 7th, 2013

Carnegie Mellon

Homework Reminder

- Please read the homework policy on the web site (though collaboration is encouraged)
- "Late fees" apply although extensions can be granted
- ~~Current homework due next Monday~~
- ~~Next homework will be distributed on Monday, will include parts on search, which will be discussed next week~~

Agenda

- Review: Speech Recognition so far, literature
- Motivation and Definition
- Language Modeling in ASR
- Perplexity
- Smoothing techniques

Language Modeling Literature

- Roni Rosenfeld, *"Two decades of statistical language modeling: Where do we go from here?"* presented at the 2000 Spoken Language Recognition and Understanding Workshop; Summit, NJ; Feb. 2000.
- Jerome Bellegarda, *"Statistical language model adaptation: Review and perspectives",* Speech Communication, vol. 42, pp. 93-108, 2004.

For all literature, see the website. Also, consult the books, in particular *Jelinek* and *Huang/ Acero/ Hon*.

Fundamental Problem of Speech Recognition

interACT

Given: an observation (ADC, FFT) $X = x_1, x_2, \ldots, x_T$

Wanted: the corresponding word sequence $W = w_1, w_2, \ldots, w_m$

Search: the most likely word sequence W'

$$W' = \arg\max_W P(W \mid X) = \arg\max_W \frac{p(X \mid W)P(W)}{p(X)} = \arg\max_W p(X \mid W)P(W)$$

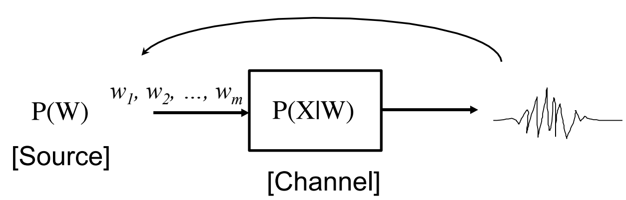
$p(X|W)$ = The **acoustic model** (how likely is it to observe X when W is spoken)

$P(W)$ = The **language model** (how likely is it that W is spoken a-priori)

- Dependent on domain, language, etc.
- Independent of specific utterance

Source-Channel Model for Speech Recognition

interACT



- The channel is *noisy,* so retrieving the message from source is not trivial (Speech recognition = decoding)
- The source is modeled by P(W) – Language model
- Each word sequence $w_1, w_2 .. w_m$ is a symbol emitted by the source and has a finite probability under P(W)

Stochastic Language Models

In formal language theory $P(W)$ is regarded either as

- 1 if word sequence W is accepted
- 0 if word sequence W is rejected

Inappropriate for spoken language since,

- Grammar has no complete coverage
- (Conversational) spoken language is often ungrammatical

Describe $P(W)$ from the probabilistic viewpoint

- Occurrence of word sequence W is described by a probability $P(W)$
- Find a good way to accurately estimate $P(W)$

Training problem: reliably estimate probabilities of W

Recognition problem: compute probabilities for generating W

Another Motivation

Equally important to recognize and understand natural speech:

- **Acoustic** pattern matching and knowledge about **language**

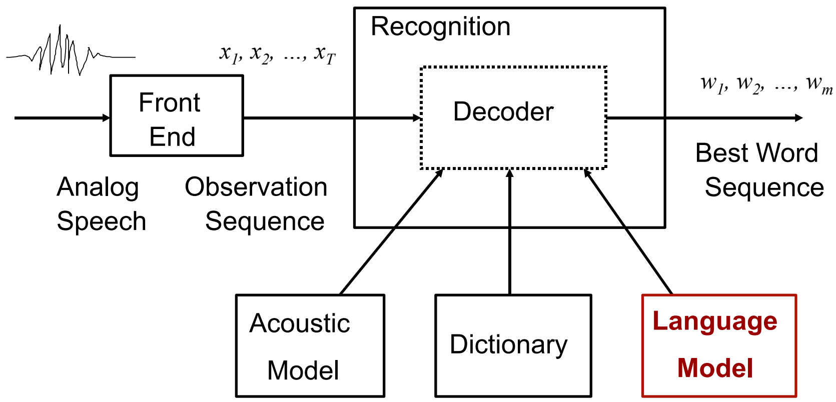
Language Knowledge:

- Lexical knowledge
  - vocabulary definition
  - word pronunciation
- Syntax and Semantics, i.e. rules that determine:
  - word sequence is grammatically well-formed
  - word sequence is meaningful
- Pragmatics
  - structure of extended discourse
  - what is likely to be said in particular context
- These different levels of knowledge are tightly integrated!!!

In ASR covered by:

Vocabulary
Dictionary
LM
Grammar
Context
LM
Grammar
Discourse

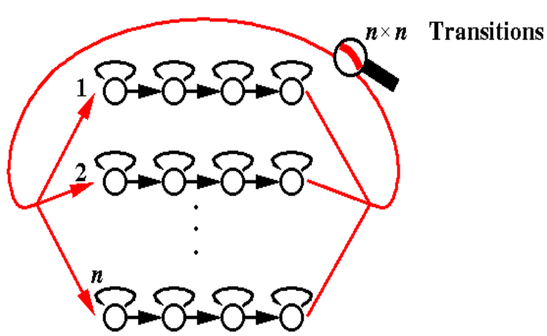
Speech Recognition Components



Deterministic vs. Stochastic Language Models

In HMM-recognizers, the language model is responsible for the computation of the word-to-word transition probabilities (a_{ij}).

These can be computed on the fly, and may depend on more than just the previous word.



LMs can be

- **Deterministic**: $P(w_j|w_i) = 0.0$ or $1.0/n$ (e.g. finite state grammars)
- **Stochastic**: transition probabilities are in the range 0.0 to 1.0

A Word Guessing Game

interACT

Good morning, how are ____

A Word Guessing Game

interACT

Good morning, how are you?

[…] very ______

A Word Guessing Game

Good morning, how are you?

[…] very ______

I apologize for being late, I am very ______

A Word Guessing Game

Good morning, how are you?

[…] very ______

I apologize for being late, I am very sorry!

A Word Guessing Game

Good morning, how are you?

[…] very ______

I apologize for being late, I am very sorry!

My favorite OS is _________

A Word Guessing Game

Good morning, how are you?

[…] very ______

I apologize for being late, I am very sorry!

My favorite OS is →
- Amiga OS
- Unix
- Windows
- Mac OS

A Word Guessing Game

interACT

Good morning, how are you?

[…] very ______

I apologize for being late, I am very sorry!

My favorite OS is
- Amiga OS
- Unix
- Windows
- Mac OS

Hello, I am very happy to see you, Mr. ____________

A Word Guessing Game

interACT

Good morning, how are you?

[…] very ______

I apologize for being late, I am very sorry!

My favorite OS is
- Amiga OS
- Unix
- Windows
- Mac OS

Hello, I am very happy to see you, Mr.
- Black
- Orange
- Jones
- Smith
- President
- …

A Word Guessing Game

What do we learn from the word guessing game?

- For some histories the number of expected words is rather small.
- For some histories we can make virtually no prediction about the next word.
- The more words fit at some point the more difficult it is to recognize the correct one (more errors are possible)
- The difficulty of recognizing a word sequence is correlated with the "branching degree"

What do we expect from Language Models in SR?

- **Improve speech recognizer**: add another information source
- **Disambiguate homophones**: find out that
  - "`I OWE YOU TOO`" is more likely than
  - "`EYE O U TWO`"
- **Search space reduction**: when vocabulary is n words, don't consider all n^k possible k-word sequences
- **Analysis**: analyze utterance to *understand* what has been said
  - Disambiguate homonyms (“bank”: money vs river)

Probabilities of Word Sequences

- The probability of a word sequence can be decomposed as:
  $P(W) = P(w_1\, w_2 \,..\, w_n) = P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1 w_2) \cdot\cdot\cdot P(w_n \mid w_1 w_2 \ldots w_{n\text{-}1})$

- The choice for a good w_n thus depends on the entire history of the input, so when computing $P(w \mid \text{history})$, we have a problem:
  - For a vocabulary of 64,000 words and average sentence lengths of 25 words (typical for Wall Street Journal), we end up with a huge number of possible histories ($64{,}000^{25} > 10^{120}$).
  - So it is impossible to pre-compute a special $P(w \mid \text{history})$ for every history.
- Two possible solutions:
  - Compute $P(w \mid \text{history})$ "on the fly" (rarely used, very expensive)
  - Replace the history by one out of a limited feasible number of equivalence classes $\boldsymbol{C}$ such that $P'(w \mid \text{history}) = P(w \mid \boldsymbol{C}(\text{history}))$

- **Question:** how do we find good equivalence classes $\boldsymbol{C}$?

Classification of Word Sequence Histories

We can use different equivalence classes using information about:

- Grammatical content (phrases like noun-phrase, etc.)
- POS = part of speech of previous word(s) (e.g. subject, object, ...)
- Semantic meaning of previous word(s)
- Context similarity (words that are observed in similar contexts are treated equally, e.g. weekdays, people's names etc.)
- Apply some kind of automatic clustering (top-down, bottom-up)
- Classes are simply based on previous words
  - **unigram**: $P'(w_k \mid w_1 w_2 \ldots w_{k\text{-}1}) = P(w_k)$
  - **bigram**: $P'(w_k \mid w_1 w_2 \ldots w_{k\text{-}1}) = P(w_k \mid w_{k\text{-}1})$
  - **trigram**: $P'(w_k \mid w_1 w_2 \ldots w_{k\text{-}1}) = P(w_k \mid w_{k\text{-}2}\, w_{k\text{-}1})$
  - ***n*-gram**: $P'(w_k \mid w_1 w_2 \ldots w_{k\text{-}1}) = P(w_k \mid w_{k\text{-}(n\text{-}1)} w_{k\text{-}n\text{-}2} \ldots w_{k\text{-}1})$

Estimation of N-grams

interACT

The standard approach to estimate $P(w|\ history)$ is

- Use a large amount of training corpus (“there's no data like more data”)
- Determine the FREQUENCY with which the word *w* occurs given the *history*
  - Count how often the word sequence “*history w*” occurs in the text
  - Normalize the count by the number of times *history* occurs

$$P(w|\ \text{history}) = \frac{\text{Count}(\text{history}\ w)}{\text{Count}(\text{history})}$$

Example

interACT

Training corpus consists of 3 sentences, use bigram model

John read a book.
I read a different book.
John read a book by Mulan.

$$P(w|\ \text{history}) = \frac{\text{Count}(\text{history}\ w)}{\text{Count}(\text{history})}$$

```
P(John|<s>)   = C(<s>,John) / C(<s>)     = 2/3
P(read|John)  = C(John,read) / C(John)   = 2/2
P(a|read)     = C(read,a) / C(read)      = 3/3
P(book|a)     = C(a,book) / C(a)         = 2/3
P(</s>|book)  = C(book, </s>) / C(book)  = 2/3
```

Calculate the probability of sentence *John read a book.*

```
P(John read a different book)
 = P(John|<s>) P(read|John) P(a|read) P(different|a)
   P(book|different) P(</s>|book)
 = 0.148
```

What about “*Mulan read a book*” – We don’t have P(read|Mulan) !

Bigrams and Trigrams

- Are Bigrams / Trigrams any good?
- First experiment:
  - 1.5 million words used for training
  - 300,000 words used for testing
  - restricted to 1,000 most frequent words
  - ➢ 23% of trigrams occurring in test corpus were absent from training corpus
- Second experiment (bag of words):
  - Take any meaningful 10-word sentence (from dictation task)
  - Scramble the words into an arbitrary order
  - Find most probable order with trigram model
  - ➢ 63% perfect word-by-word reconstruction
  - ➢ 79% reconstruction that preserves meaning

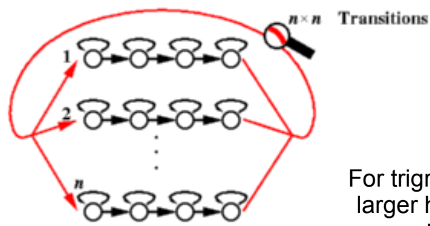
The Bag of Words Experiment

Most likely trigram sequences from randomly scrambled dictated sentence:

- I expect that the output will improve with experience.
  I expect that the output will improve with experience.
- would I report directly to you?
  I would report directly to you?
- now let me mention some of the disadvantages.
  let me mention some of the disadvantages now.
- these people have a fairly large rate of turnover.
  of these people have a fairly large turnover rate.
- exactly how this might be done is not clear.
  clear is not exactly how this might be done.

Bigrams vs. Trigrams

Bigrams can be easily incorporated into an HMM recognizer (Markov 1!):



For trigrams, we need a larger history. What if a word can have many predecessors?

Typical solution for incorporating trigrams:

- use time asynchronous search (easier to handle long history)
- for time-synchronous search: use "poor man's trigrams", i.e. consider only the predecessor's *best* predecessor instead of all.

Bigrams vs. Trigrams

- Other disadvantages of trigrams compared to bigrams:
  - coverage of test data is smaller than with bigrams
  - estimation of $P(w_k \mid w_{k-2}\, w_{k-1})$ is more difficult
- Typical error reductions:
  - **Bigrams** 30%-50% (vs Unigrams)
  - **Trigrams** 10%-20%
  - **Four-grams** 3%-5%

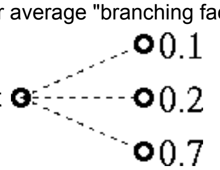
Measuring the Quality of Language Models

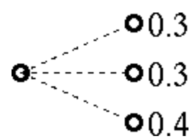
Obvious approach to finding out whether LM1 or LM2 is better: let recognizer run with both and choose the one that produces fewer errors, but:

- Performance of recognizer depends also on acoustic model.
- Performance of recognizer depends also on combination mechanism of acoustic model with language model.
- Expensive and time consuming

What, if no recognizer available?

- We would like to have an independent measure:
- Declare a LM to be good, if it makes the task easier for the recognizer (i.e. if it has a smaller average "branching factor").

Easy to see that 

is better than 

The Perplexity of a Language Model

Language can be thought of as an information source whose output are words w_i belonging to the vocabulary of that language. The entropy of an information source emitting words $w_1\ w_2$ ... from language L is defined as:

$$H(L) = -\sum_{w1\ w2\ ...\ wn \in L} P(w_1\ w_2\ ...\ w_n) \cdot \log_2 P(w_1\ w_2\ ...\ w_n)$$

The entropy rate for a finite length sequence is measured as entropy per word. It can be expressed as

$$H_R(L) = (1/n)\ H(L) = -\ (1/n) \sum_{w1\ w2\ ...\ wn \in L} P(w_1\ w_2\ ...\ w_n) \cdot \log P(w_1\ w_2\ ...\ w_n)$$

The true entropy rate $H_R(L)$ of a language is calculated over infinite length sequences

$$H_R(L) = -\lim_{n\to\infty} (1/n) \sum_{w1\ w2\ ...\ wn \in L} P(w_1\ w_2\ ...\ w_n) \cdot \log P(w_1\ w_2\ ...\ w_n)$$

According to Shannon-McMillan-Breiman theorem for infinite length sequences,

$$H_R(L) = -\lim_{n\to\infty} (1/n) \log P(w_1\ w_2\ ...\ w_n)$$

The Perplexity of a Language Model

interACT

The entropy rate $H_R(L)$ for an infinite sequence can be approximated by a (sufficiently long) sequence of finite length from L

$$H_R(L) = - (1/n) \log P(w_1 w_2 \dots w_n)$$

The entropy rate of a source $H_R(L)$ is a measure for the difficulty for a recognizer to recognize the source's language.

- High entropy → difficult task
- Low entropy → easy task

If we consider P as *the* language model, then the entropy rate is also a measure for the quality of a language model. The term

$$PP = 2\^{}H_R(L) = P(w_1 w_2 \dots w_n)^{-1/n}$$

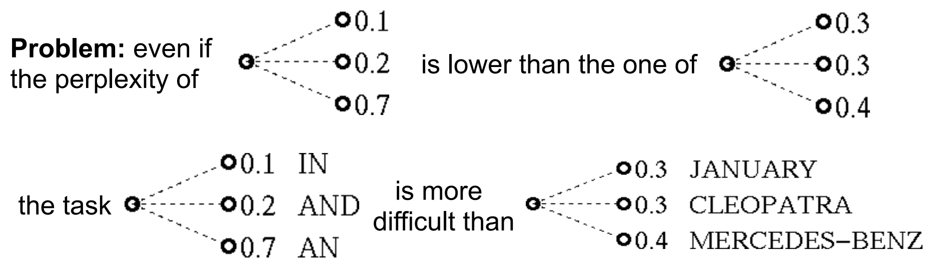
is called the **perplexity** of a language model P on the test set $W = w_1 w_2 \dots w_n$. The perplexity can also be regarded as the geometric mean of all branching factors.

Some Perplexities

interACT

Task	Vocabulary	Language Model	Perplexity
Conference Registration	400	Bigrams	7
Resource Management	1000	Bigrams	20
Wall Street Journal	60.000	Bigrams	160
Wall Street Journal	60.000	Trigrams	130
Arabic Broadcast News	220.000	Fourgrams	212
Chinese Broadcast News	90.000	Fourgrams	430

Some Perplexities



Smoothing

Key problem in n-gram modeling is the **data sparseness** problem

- many possible word successions may not be well observed
- or even not be seen at all!

Example: Given several million word collection of English text

- 50% of trigrams occur only once
- 80% of trigrams occur less than 5 times
- ⇨ Smoothing is critical to make probabilities robust for unseen data

Smoothing

Remember example: *Mulan read a book*

We had: Count(*Mulan,read*) = 0

$$P(read|Mulan) = \frac{\text{Count}(Mulan,read)}{\sum_w \text{Count}(Mulan,w)} = \frac{0}{1}$$

In ASR if $P(W)=0$, the string W will never be considered as hypothesis, thus whenever a string W with $P(W)$ occurs, an error will be made

⇨ Assign all strings a nonzero probability to prevent ASR errors

Smoothing concept

- Steps:
  - Subtract counts from **seen** events
  - Redistribute collected counts to **unseen** events (count=0)
- Analogy: Taxation
  - Collect tax from the rich
  - Redistribute it to the poor