Intro to Text Processing
Lecture 4
Behrang Mohit

Review: Learning from data

- Training
  - Input: training data
  - Output: A statistical model
- Testing
  - Input: test data (usually gold-standard)
  - Output: ML’s predictions or annotation
- Tuning
Supervised vs. unsupervised

- Supervised learning uses labeled training data
- Unsupervised learning uses unlabeled (raw) training data
- Semi-supervised: mix

Classification Examples

- Classification: Learner predicts the class label for each point (instance)
Review: Document classification

• Document classification
  – Assigning class label to a document
  – Supervised learning

• What happens during the learning?

• What is in the model?
  – Collect the statistics from the labeled data
Review: Document classification

• Document classification
  – Assigning class label to a document
  – Supervised learning

• What happens during the learning?
  – Collect the statistics from the labeled data

• What is in the model?
  – Prior and conditional probabilities

Cross Validation

• The training and testing split can be biased
  – Maybe lucky and easier instances go to the test

• Split the data into pools

• Rotate: train, test/evaluate
• Average the evaluations
Cross Validation

• The training and testing split can be biased
  – Maybe lucky and easier instances go to the test

• Split the data into pools

  train  train  train  train  test

• Rotate: train, test/evaluate
Cross Validation

• The training and testing split can be biased
  – Maybe lucky and easier instances go to the test
• Split the data into pools

  train  train  test  train  train

• Rotate: train, test/evaluate

Today: Language Modeling

• Admin:
  – Homework 1 due in a week
    • Start!
• Language Modeling
• Monday: Dr. Houda Bouamor (guest lecture)
Probabilistic Language Models

- **Today’s goal**: assign a probability to a sentence
  - Machine Translation:
    - $P(\text{high winds tonight}) > P(\text{large winds tonight})$
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
  - Speech Recognition
    - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Noisy Channel

- Noisy channel paradigm in text processing
  - Corrupted text which needs to be cleaned
    - Machine Translation
    - Optical Character Recognition
    - Spell checking
The Noisy-channel Paradigm

- A generator produces a “signal” and sends it over a “channel” which is prone to “noise”.
- A “corrupted” version of the signal is “observed”
- The “observer” tries to guess that the original “signal” was.
Bayes’ Rule in noisy channel

**Problem:**
- Given the channel’s observation/output, what’s the best source

\[ \hat{W} = \arg\max_W P(W|O) \]

- This requires estimation of \( P(W|O) \)
  - Not always simple

Bayes’ Rule in noisy channel

\[ \hat{W} = \arg\max_W P(W|O) \]

**Bayes’ Rule:**
- Decomposition:

\[ P(x|y) = \frac{P(y|x)P(x)}{P(y)} \]

\[ \hat{W} = \arg\max_W P(W|O) \]

\[ \hat{W} = \arg\max_W \frac{P(O|W)P(W)}{P(O)} = \arg\max_W P(O|W)P(W) \]
Language Model: the prior probability

\[ \hat{w} = \arg \max_w P(O|W)P(W) \]

- \( P(O|W) \): (Noisy-Input | Clear-Output)
  - Speech: \( P(\text{utterance} | \text{words}) \)
  - Translation: \( P(\text{Src. Lang.} | \text{Tgt. Lang.}) \)
  - OCR: \( P(\text{Scanned Img} | \text{Text}) \)

- \( P(W) \): Prior probability of source
  - \( P(\text{English Sentence}), P(\text{Phrase}), \text{etc.} \)
  - Language Model is used as a prior in many NLP tasks

Language Modeling

- Problem: What is the probability of an English sentence?
  - We need a model of English Language.
    - \( P(\text{Sun rises in the East}) = ? \)
    - \( P(\text{Sun rises in the West}) = ? \)
    - \( P(\text{West rises in the Sun}) = ? \)
    - \( P(\text{the rises in sun east}) = ? \)
Learning Framework in Language Modeling

- **Input**: A set of words: w1 w2 ... wn
- **Output**: The probability of the input

\[ P(w_1w_2...w_n) \]

- Training Data: Fine quality text in the language
  - e.g.: English News Text

Estimating the probabilities

- **Estimating** \( P(w_1w_2...w_n) \)
  - Corpus gives us the probabilities (parameters)
    - Maximum Likelihood Estimation
    - \( P(\text{Music}) = 400/1,000,000 = 0.00004 \)

- Corpus does not hold every English sentence.
  - Does it mean that the probability of missing sentences is zero!!
Probability of a sentence

- Probability of generating a sentence
  \[ P(w_1w_2...w_n) \]
  - Chain Rule:

  - Transformed Problem: Given a context of words, Probability of generating a new word

\[
\begin{align*}
P(w_1w_2) &= P(w_2|w_1)P(w_1)
\end{align*}
\]
Probability of a sentence

• Probability of generating a sentence
  \[ P(w_1w_2\ldots w_n) \]

• Chain Rule:
  \[ P(w_1w_2) = P(w_2|w_1)P(w_1) \]
  \[ P(w_1w_2w_3) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \]
  \[ P(w_1w_2\ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_n|w_1w_2\ldots w_{n-1}) = \prod_{k=1}^{n} P(w_k|w_1w_2\ldots w_{k-1}). \]

• Transformed Problem: Given a context of words, Probability of generating a new word
  \[ P(w_n|w_1w_2\ldots w_{n-1}) \]
How to estimate these probabilities

• Could we just count and divide?

\[ P(\text{its water is so transparent that}) = \frac{\text{Count(its water is so transparent that the)}}{\text{Count(its water is so transparent that)}} \]

• We’ll never see enough data for estimating these

N-gram Language Model

\[ P(w_1w_2...w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_1w_2...w_{n-1}) = \prod_{k=1}^{n} P(w_k|w_1w_2...w_{k-1}) \]

• Problem: \[ P(w_n|w_1w_2...w_{n-1}) \]
  – Sparseness problem continues to hold for long sentences

• Simplification: Markov Model
  – Predict the probability of future without looking too far into the past
Markov Assumption

• Simplifying assumption:

\[ P(\text{the l its water is so transparent that}) \approx P(\text{the l that}) \]

• Or maybe

\[ P(\text{the l its water is so transparent that}) \approx P(\text{the l transparent that}) \]

Markov Assumption

\[
P(w_1w_2\ldots w_n) \approx \prod P(w_i \mid w_{i-k} \ldots w_{i-1})
\]

• In other words, we approximate each component in the product

\[
P(w_i \mid w_1w_2\ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})
\]
Simplest case: Unigram model

\[ P(w_1w_2...w_n) \approx \prod_i P(w_i) \]

Some automatically generated sentences from a unigram model:

- fifth an of futures the an incorporated a a the inflation most dollars quarter in is mass
- thrift did eighty said hard ‘m july bullish
- that or limited the

Bigram model

- Condition on the previous word:

\[ P(w_i | w_1w_2...w_{i-1}) \approx P(w_i | w_{i-1}) \]

texaco rose one, in, this issue is pursuing growth in a boiler house said mr. gurria mexico ‘s motion control proposal without permission from five hundred fifty five, yen outside new car parking lot of the agreement reached this would be a record november
N-gram Language Model

- **Bigram:** \( w_{k-1}w_k \)
  \[ P(w_k | w_{k-1}) \]

- **Trigram:** \( w_{k-2}w_{k-1}w_k \)
  \[ P(w_k | w_{k-2}w_{k-1}) \]

- **N-gram Language Model**
  - A model of the language based on the context of the \( n-1 \) previous words.

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:
    "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models
Estimating N-Gram Probabilities

• How to compute the N-gram probabilities?
  – Use a large corpus of language
  – Collect the counts of N-Grams
  – Normalize the counts to obtain probabilities

\[ P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \]

\[ P(\text{Music}|\text{Arabic}) = \frac{C(\text{ArabicMusic})}{C(\text{Arabic})} \]
Estimating N-Gram Probabilities

- How to compute the N-gram probabilities?
  - Use a large corpus of language
  - Collect the counts of N-Grams
  - Normalize the counts to obtain probabilities

\[
P(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}
\]

\[
P(\text{Music} | \text{Arabic}) = \frac{C(\text{Arabic:Music})}{C(\text{Arabic})}
\]

\[
P(\text{Concert} | \text{Arabic:Music}) = \frac{C(\text{Arabic:Music:Concert})}{C(\text{Arabic:Music})}
\]

An example

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

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

\[
P(\text{I} | \text{s}) = \frac{2}{3} = .67
\]

\[
P(\text{Sam} | \text{s}) = \frac{1}{3} = .33
\]

\[
P(\text{am} | \text{I}) = \frac{2}{3} = .67
\]

\[
P(\text{do} | \text{I}) = \frac{1}{3} = .33
\]

\[
P(\text{I} | \text{s}) = \frac{2}{3} = .67
\]

\[
P(\text{Sam} | \text{s}) = \frac{1}{3} = .33
\]

\[
P(\text{am} | \text{I}) = \frac{2}{3} = .67
\]

\[
P(\text{do} | \text{I}) = \frac{1}{3} = .33
\]
Example: Berkeley Restaurant Corpus

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i’m looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

Bigram estimates of sentence probabilities

\[
P(<s> \text{ I want english food } </s>) = \\
P(I|<s>) \\
\times P(\text{want}|I) \\
\times P(\text{english}|\text{want}) \\
\times P(\text{food}|\text{english}) \\
\times P(</s>|\text{food}) \\
= .000031
\]
### Estimating N-grams

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
</tr>
</thead>
<tbody>
<tr>
<td>want</td>
<td>3437</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>1215</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>eat</td>
<td>3256</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
</tr>
<tr>
<td>Chinese</td>
<td>938</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>food</td>
<td>213</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>1506</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Estimating N-grams

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
</tr>
</thead>
<tbody>
<tr>
<td>want</td>
<td>3437</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>1215</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>eat</td>
<td>3256</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
</tr>
<tr>
<td>Chinese</td>
<td>938</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>food</td>
<td>213</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
</tr>
</thead>
<tbody>
<tr>
<td>want</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.038</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>.0025</td>
<td>0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
<td>.006</td>
</tr>
<tr>
<td>eat</td>
<td>.0009</td>
<td>0</td>
<td>.0031</td>
<td>.26</td>
<td>.00092</td>
<td>0</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0094</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.020</td>
<td>.0021</td>
</tr>
<tr>
<td>food</td>
<td>.013</td>
<td>0</td>
<td>.011</td>
<td>0</td>
<td>0</td>
<td>.56</td>
</tr>
</tbody>
</table>
Problem: Unseen Events

- $P(\text{Mozart died in 1791})$
  - $P(1791 \mid \text{in})$
  - $P(1791 \mid \text{died in})$
  - $P(\text{in} \mid \text{Mozart died})$

- What if $P(1791 \mid \text{in}) = 0$?
  - $P(\text{Mozart died in 1791}) = 0$?

- Zero is not an accurate estimation of unseen N-grams.
Parameter Smoothing

- **Smoothing**: Allocating a portion of probability mass for unseen events
  - $P(\text{died in } | 1791) = 0.00000001$

Add-one Smoothing

- Add one to all of the counts
  \[
P(w_i) = \frac{C(w_i)}{N} \quad \text{to} \quad P(w_i) = \frac{C(w_i)+1}{N+V}
\]
## Discounting by one

<table>
<thead>
<tr>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>eat</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Too much discounting

<table>
<thead>
<tr>
<th>I</th>
<th>Want</th>
<th>to</th>
<th>Eat</th>
<th>Chinese</th>
<th>food</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.0038</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>.0025</td>
<td>0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
</tr>
<tr>
<td>eat</td>
<td>.0009</td>
<td>0</td>
<td>.0031</td>
<td>.26</td>
<td>.00092</td>
</tr>
<tr>
<td>Chinese</td>
<td>0</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.020</td>
</tr>
<tr>
<td>food</td>
<td>.0094</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th>Want</th>
<th>to</th>
<th>Eat</th>
<th>Chinese</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want</td>
<td>.0018</td>
<td>.22</td>
<td>.0002</td>
<td>.0028</td>
<td>.0002</td>
</tr>
<tr>
<td>to</td>
<td>.0014</td>
<td>.0003</td>
<td>.28</td>
<td>.0003</td>
<td>.0025</td>
</tr>
<tr>
<td>eat</td>
<td>.0008</td>
<td>.0002</td>
<td>.0023</td>
<td>.18</td>
<td>.0008</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0003</td>
<td>.0003</td>
<td>.0012</td>
<td>.0003</td>
<td>.0078</td>
</tr>
<tr>
<td>food</td>
<td>.0016</td>
<td>.0005</td>
<td>.0005</td>
<td>.0005</td>
<td>.0005</td>
</tr>
<tr>
<td></td>
<td>.0064</td>
<td>.0003</td>
<td>.0058</td>
<td>.0003</td>
<td>.00032</td>
</tr>
</tbody>
</table>
Too much discounting

Language

Language Model

15383: txt proc

Too much discounting

Language

Language Model

15383: txt proc
Advanced Smoothing

• Modeling Story:
  – A zero-frequency word: A word which is not seen yet.
  – When we see it, that’s the first occurrence

• Use the 1-frequency N-grams to model the probability of zero-frequency N-grams.
  – The probability mass is taken from all N-grams proportionally.

• Smoothing based on 1-frequency events:
  – Witten Bell
  – Good-Turing

Back-off and Deleted Interpolation

• Back-off: Relying on lower order N-grams
  \[ P(w_i|w_{i-2}w_{i-3}) = \begin{cases} P(w_i|w_{i-3}w_{i-4}), & \text{if } C(w_{i-2}w_{i-3}w_i) > 0 \\ \alpha_1 P(w_i|w_{i-1}), & \text{if } C(w_{i-2}w_{i-3}w_i) = 0 \text{ & } C(w_{i-1}w_i) > 0 \\ \alpha_2 P(w_i), & \text{Otherwise} \end{cases} \]

• Deleted Interpolation:
  – Linear Interpolation of different N-gram orders
  \[ P(w_i|w_{i-2}w_{i-3}) = \lambda_1 P(w_i|w_{i-3}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i) \]
Working in the log space

• Computing the probability of long sentences

\[ P(w_1 w_2 \ldots w_n) = P(w_1) P(w_2 | w_1) P(w_3 | w_1 w_2) \ldots P(w_n | w_1 w_2 \ldots w_{n-1}) = \prod_{k=1}^{n} P(w_k | w_1 w_2 \ldots w_{k-1}) \]

– Underflow problem with multiplications
– Transform the probabilities to Log probabilities

\[ \log(P(w_1 w_2 \ldots w_n)) = \log(\prod_{k=1}^{n} P(w_k | w_1 w_2 \ldots w_{k-1})) = \sum_{k=1}^{n} \log(P(w_k | w_1 w_2 \ldots w_{k-1})) \]

Evaluation of LM

• Two kinds of evaluation:

  – Intrinsic evaluation: Evaluate the actual system
    • Evaluate text classification system
    • Evaluate question answering system

  – Extrinsic evaluation: Evaluate the effect of the system on another system:
    • Text classification within search
    • Language model within machine translation
Evaluation of Language Model

• Extrinsic evaluation of LM”
  – Evaluate the task that LM is used:
    • Evaluate Machine Translation
    • Evaluate Speech Processing

• Task Independent Evaluation
  – Cross-Entropy of the model

LM evaluation by Cross Entropy

• Entropy: Weighted average number of choice a random variable has to make.

• A Language model which has less choices for a given context is preferred
  – Lowering the uncertainty of the model
Why N-gram works?

- “The computer which I had just put into the machine room on the fifth floor crashed.”

- Infrequent long dependencies
  - Collin (1997)
    - 74% of dependencies are with an adjacent word (95% are in the context of 5 words).

- Strong performance of N-gram models
  - Simple implementation
  - Fast and easy to be incorporated

SRILM

- SRI-LM package
  - De facto LM tool for a decade

- Constructing the LM
  - Training data
  - Probabilities
    - Log-space
  - Back-off probability

```
data
ngram 1=5
ngram 2=5
ngram 3=4
\1-grams:
-0.8751 This -0.3358
-0.8751 a -0.3010
-0.8751 is -0.3358
-1.1761 second -0.3358
-0.8751 test -0.3979
\2-grams:
-0.2218 This is 0.0000
-0.5229 a second 0.0000
-0.5229 a test. -0.3979
-0.2218 is a 0.0000
-0.2218 second test -0.3979
\3-grams:
-0.2218 This is a
-0.2218 a second test
-0.5229 is a second
-0.5229 is a test
\end
```
SRILM’s capabilities

• Acquiring counts
• Acquiring N-gram models
• Many smoothing algorithms
• Model interpolation

Back to the big picture?

• Why do we need LM?
• Noisy channel paradigm in text processing
  – Corrupted text which needs to be cleaned
    • Machine Translation
    • Optical Character Recognition
    • Spell checking
The Noisy-channel Paradigm

Bayes’ Rule in noisy channel

- **Problem:**
  - Given the channel’s observation/output, what’s the best source
    \[ \hat{W} = \arg\max_W P(W|O) \]
  - This requires estimation of \( P(W|O) \)
    - Not always simple
Bayes’ Rule in noisy channel

\[ \hat{W} = \arg \max_{W} P(W|O) \]

• Bayes’ Rule Decomposition:

\[ \hat{W} = \arg \max_{W} P(W|O) \]
\[ \hat{W} = \arg \max_{W} \frac{P(O|W)P(W)}{P(O)} = \arg \max_{W} P(O|W)P(W) \]

Language Model: the prior probability

\[ \hat{W} = \arg \max_{W} P(O|W)P(W) \]

– \( P(O|W) \): (Noisy-Input | Clear-Output)
  • Speech: \( P(\text{utterance} | \text{words}) \)
  • Translation: \( P(\text{Src. Lang.} | \text{Tgt. Lang.}) \)
  • OCR: \( P(\text{Scanned Img} | \text{Text}) \)

– \( P(W) \): Prior probability of source
  • \( P(\text{English Sentence}), P(\text{Phrase}), \text{etc.} \)
  • Language Model is used as a prior in many NLP tasks
Probabilistic Language Models

- **LM**: assign a probability to a sentence
  - Machine Translation:
    - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
  - Spell Correction
    - The office is about fifteen *minuets* from my house
      - $P(\text{about fifteen minuets from}) > P(\text{about fifteen minutes from})$
  - Speech Recognition
    - $P(\text{I saw a van}) >> P(\text{eyes awe of an})$