Algorithms for NLP

Language Modeling I

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
The Noisy-Channel Model

- We want to predict a sentence given acoustics:

\[ w^* = \arg \max_w P(w|a) \]

- The noisy-channel approach:

\[ w^* = \arg \max_w P(w|a) \]

\[ = \arg \max_w P(a|w)P(w)/P(a) \]

\[ \propto \arg \max_w P(a|w)P(w) \]

Acoustic model: HMMs over word positions with mixtures of Gaussians as emissions

Language model: Distributions over sequences of words (sentences)
$\text{argmax } P(w|a) = \text{argmax } P(a|w)P(w)$
Acoustic Confusions

the station signs are in deep in english -14732
the stations signs are in deep in english -14735
the station signs are in deep into english -14739
the station 's signs are in deep in english -14740
the station signs are in deep in the english -14741
the station signs are indeed in english -14757
the station 's signs are indeed in english -14760
the station signs are indians in english -14790
the station signs are indian in english -14799
the stations signs are indians in english -14807
the stations signs are indians and english -14815
“Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded— one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: ‘This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.’ ”

Warren Weaver (1947)
MT System Components

Language Model

```
source
P(e)
```

Translation Model

```
channel
P(f|e)
```

```
argmax P(e|f) = argmax P(f|e)P(e)
```

```
best
e
```

```
decoder
```

```
oberved
f
```

```
``
Other Noisy Channel Models?

- We’re not doing this only for ASR (and MT)
  - Grammar / spelling correction
  - Handwriting recognition, OCR
  - Document summarization
  - Dialog generation
  - Linguistic decipherment
  - ...
A language model is a distribution over sequences of words (sentences)

\[ P(w) = P(w_{\downarrow 1} \ldots w_{\downarrow n}) \]

- What’s w? (closed vs open vocabulary)
- What’s n? (must sum to one over all lengths)
- Can have rich structure or be linguistically naive

Why language models?
- Usually the point is to assign high weights to plausible sentences (cf acoustic confusions)
- This is not the same as modeling grammaticality
N-Gram Models
N-Gram Models

- Use chain rule to generate words left-to-right

\[ P(w_1 \ldots w_n) = \prod_i P(w_i|w_1 \ldots w_{i-1}) \]

- Can’t condition on the entire left context

\[ P(??? \mid \text{Turn to page 134 and look at the picture of the}) \]

- N-gram models make a Markov assumption

\[ P(w_1 \ldots w_n) = \prod_i P(w_i|w_{i-k} \ldots w_{i-1}) \]

\[ P(\text{please close the door}) = \]

\[ P(\text{please|START})P(\text{close|please}) \ldots P(\text{STOP|door}) \]
Empirical N-Grams

- How do we know $P(w \mid \text{history})$?
  - Use statistics from data (examples using Google N-Grams)
  - E.g. what is $P(\text{door} \mid \text{the})$?

| Training Counts | \( \hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162} \) |
|-----------------|----------------------------------------------------------|
| 198015222       | the first                                               |
| 194623024       | the same                                                |
| 168504105       | the following                                           |
| 158562063       | the world                                               |
| ...             |                                                         |
| 14112454        | the door                                                |
| 23135851162     | the *                                                   |

- This is the maximum likelihood estimate
Increasing N-Gram Order

- **Higher orders capture more dependencies**

<table>
<thead>
<tr>
<th>Bigram Model</th>
<th>Trigram Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>198015222 the first</td>
<td>197302 close the window</td>
</tr>
<tr>
<td>194623024 the same</td>
<td>191125 close the door</td>
</tr>
<tr>
<td>168504105 the following</td>
<td>152500 close the gap</td>
</tr>
<tr>
<td>158562063 the world</td>
<td>116451 close the thread</td>
</tr>
<tr>
<td>...</td>
<td>87298 close the deal</td>
</tr>
<tr>
<td>14112454 the door</td>
<td><strong>------------------</strong></td>
</tr>
<tr>
<td><strong>------------------</strong></td>
<td>3785230 close the *</td>
</tr>
<tr>
<td>23135851162 the *</td>
<td></td>
</tr>
</tbody>
</table>

\[
P(\text{door} | \text{the}) = 0.0006 \quad \text{P(\text{door} | \text{close the}) = 0.05}
\]
Increasing N-Gram Order

- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Every enter now severally so, let
- Hill he late speaks; or! a more to leg less first you enter
- Are where exeunt and sighs have rise excellency took of. Sleep knave we. near; vile like
Sparsity

Please close the first door on the left.

3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
...
0 please close the first
------------
13951 please close the *
Sparsity

- **Problems with n-gram models:**
  - New words (open vocabulary)
    - Synaphtitate
    - 132,701.03
    - multidisciplinarization
  - Old words in new contexts

- **Aside: Zipf’s Law**
  - Types (words) vs. tokens (word occurrences)
  - Broadly: most word types are rare ones
  - Specifically:
    - Rank word types by token frequency
    - Frequency inversely proportional to rank
  - Not special to language: randomly generated character strings have this property (try it!)
  - This law qualitatively (but rarely quantitatively) informs NLP
N-Gram Estimation
Smoothing

- We often want to make estimates from sparse statistics:

\[ P(w \mid \text{denied the}) \]
- 3 allegations
- 2 reports
- 1 claims
- 1 request
- 7 total

- Smoothing flattens spiky distributions so they generalize better:

\[ P(w \mid \text{denied the}) \]
- 2.5 allegations
- 1.5 reports
- 0.5 claims
- 0.5 request
- 2 other
- 7 total

- Very important all over NLP, but easy to do badly
Likelihood and Perplexity

- How do we measure LM “goodness”?  
  - Shannon’s game: predict the next word

  When I eat pizza, I wipe off the __________

- Formally: define test set (log) likelihood

  \[ \log P(X|\theta) = \sum_{w \in X} \log P(w|\theta) \]

- Perplexity: “average per word branching factor”

  \[ \text{perp}(X, \theta) = \exp \left( - \frac{\log P(X|\theta)}{|X|} \right) \]
Measuring Model Quality (Speech)

- We really want better ASR (or whatever), not better perplexities.

- For speech, we care about word error rate (WER).

\[
\text{Correct answer: } \quad \text{Andy saw a part of the movie}
\]

\[
\text{Recognizer output: } \quad \text{And he saw apart of the movie}
\]

\[
\text{WER: } \quad \frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{true sentence size}} = \frac{4}{7} = 57\%
\]

- Common issue: intrinsic measures like perplexity are easier to use, but extrinsic ones are more credible.
Key Ideas for N-Gram LMs
Idea 1: Interpolation

Please close the first door on the left.

<table>
<thead>
<tr>
<th>4-Gram</th>
<th>3-Gram</th>
<th>2-Gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>3380 please close the door</td>
<td>197302 close the window</td>
<td>198015222 the first</td>
</tr>
<tr>
<td>1601 please close the window</td>
<td>191125 close the door</td>
<td>194623024 the same</td>
</tr>
<tr>
<td>1164 please close the new</td>
<td>152500 close the gap</td>
<td>168504105 the following</td>
</tr>
<tr>
<td>1159 please close the gate</td>
<td>116451 close the thread</td>
<td>158562063 the world</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0 please close the first</td>
<td>8662 close the first</td>
<td>13951 please close the *</td>
</tr>
<tr>
<td>0.0</td>
<td>0.002</td>
<td>0.009</td>
</tr>
</tbody>
</table>

Specific but Sparse       Dense but General
(Linear) Interpolation

- Simplest way to mix different orders: linear interpolation

\[ \lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w) \]

  - How to choose lambdas?
  - Should lambda depend on the counts of the histories?

- Choosing weights: either grid search or EM using held-out data

- Better methods have interpolation weights connected to context counts, so you smooth more when you know less
Want to maximize likelihood on test, not training data

- Empirical n-grams won’t generalize well
- Models derived from counts / sufficient statistics require generalization parameters to be tuned on held-out data to simulate test generalization

Set hyperparameters to maximize the likelihood of the held-out data (usually with grid search or EM)
Idea 2: Discounting

- Observation: N-grams occur more in training data than they will later

Empirical Bigram Counts (Church and Gale, 91)

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Future c* (Next 22M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Absolute Discounting

- **Absolute discounting**
  - Reduce numerator counts by a constant $d$ (e.g. 0.75)
  - Maybe have a special discount for small counts
  - Redistribute the “shaved” mass to a model of new events

- **Example formulation**

\[
P_{\text{ad}}(w|w') = \frac{c(w', w) - d}{c(w')} + \alpha(w') \hat{P}(w)
\]
**Idea 3: Fertility**

- **Shannon game:** “There was an unexpected _____”
  - “delay”?
  - “Francisco”?

- **Context fertility:** number of distinct context types that a word occurs in
  - What is the fertility of “delay”?
  - What is the fertility of “Francisco”?
  - Which is more likely in an arbitrary new context?
Kneser-Ney Smoothing

- Kneser-Ney smoothing combines two ideas
  - Discount and reallocate like absolute discounting
  - In the backoff model, word probabilities are proportional to context fertility, not frequency

\[ P(w) \propto |\{w' : c(w', w) > 0\}| \]

- Theory and practice
  - Practice: KN smoothing has been repeatedly proven both effective and efficient
  - Theory: KN smoothing as approximate inference in a hierarchical Pitman-Yor process [Teh, 2006]
Kneser-Ney Details

- All orders recursively discount and back-off:

\[
P_k(w|\text{prev}_{k-1}) = \frac{\max(c'(\text{prev}_{k-1}, w) - d, 0)}{\sum_v c'(\text{prev}_{k-1}, v)} + \alpha(\text{prev } k - 1)P_{k-1}(w|\text{prev}_{k-2})
\]

- Alpha is computed to make the probability normalize (see if you can figure out an expression).

- For the highest order, \(c'\) is the token count of the n-gram. For all others it is the context fertility of the n-gram:

\[
c'(x) = |\{u : c(u, x) > 0\}|
\]

- The unigram base case does not need to discount.

- Variants are possible (e.g. different \(d\) for low counts)
What Actually Works?

- **Trigrams and beyond:**
  - Unigrams, bigrams generally useless
  - Trigrams much better
  - 4-, 5-grams and more are really useful in MT, but gains are more limited for speech

- **Discounting**
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell, etc...

- **Context counting**
  - Kneser-Ney construction of lower-order models

- See [Chen+Goodman] reading for tons of graphs...
Idea 4: Big Data

There’s no data like more data.
Data >> Method?

- Having more data is better...

- ... but so is using a better estimator
- Another issue: N > 3 has huge costs in speech recognizers
Tons of Data?

[Brants et al, 2007]
What about...
Unknown Words?

- What about totally unseen words?

- Most LM applications are closed vocabulary
  - ASR systems will only propose words that are in their pronunciation dictionary
  - MT systems will only propose words that are in their phrase tables (modulo special models for numbers, etc)

- In principle, one can build open vocabulary LMs
  - E.g. models over character sequences rather than word sequences
  - Back-off needs to go down into a “generate new word” model
  - Typically if you need this, a high-order character model will do
What’s in an N-Gram?

- Just about every local correlation!
  - Word class restrictions: “will have been ___”
  - Morphology: “she ___”, “they ___”
  - Semantic class restrictions: “danced the ___”
  - Idioms: “add insult to ___”
  - World knowledge: “ice caps have ___”
  - Pop culture: “the empire strikes ___”

- But not the long-distance ones
  - “The computer which I had just put into the machine room on the fifth floor ___.”
Linguistic Pain?

- **The N-Gram assumption hurts one’s inner linguist!**
  - Many linguistic arguments that language isn’t regular
    - Long-distance dependencies
    - Recursive structure

- **Answers**
  - N-grams only model local correlations, but they get them all
  - As N increases, they catch even more correlations
  - N-gram models scale much more easily than structured LMs

- **Not convinced?**
  - Can build LMs out of our grammar models (later in the course)
  - Take any generative model with words at the bottom and marginalize out the other variables
What Gets Captured?

- **Bigram model:**
  - [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]

- **PCFG model:**
  - [This, quarter, 's, surprisingly, independent, attack, paid, off, the, risk, involving, IRS, leaders, and, transportation, prices, .]
  - [It, could, be, announced, sometime, .]
  - [Mr., Toseland, believes, the, average, defense, economy, is, drafted, from, slightly, more, than, 12, stocks, .]
Scaling Up?

- There’s a lot of training data out there...

... next class we’ll talk about how to make it fit.
Other Techniques?

- Lots of other techniques
  - Maximum entropy LMs (soon)
  - Neural network LMs (soon)
  - Syntactic / grammar-structured LMs (much later)