Language Modeling

NLP ML Web
Fall 2013
Andrew Rosenberg
TA/Grader: David Guy Brizan
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \[ p(\text{message} | \text{observation}) = p(\text{observation} | \text{message}) \ p(\text{message}) \]
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \( p(\text{message} \mid \text{observation}) = p(\text{observation} \mid \text{message}) \cdot p(\text{message}) \)
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \( p(\text{message} \mid \text{observation}) = p(\text{observation} \mid \text{message}) \cdot p(\text{message}) \)
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \[ p(\text{message} \mid \text{observation}) = p(\text{observation} \mid \text{message}) p(\text{message}) \]
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \( p(\text{message} \mid \text{observation}) = p(\text{observation} \mid \text{message}) \cdot p(\text{message}) \)
Communication channels introduce errors.

Our goal is to identify the intended message given the observation.

\[ p(\text{message} | \text{observation}) = p(\text{observation} | \text{message}) \cdot p(\text{message}) \]
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \( p(\text{message} | \text{observation}) = p(\text{observation} | \text{message}) \cdot p(\text{message}) \)
Noisy Channel Model

Communication channels introduce errors.

Our goal is to identify the intended message given the observation.

\[ p(\text{message} \mid \text{observation}) = p(\text{observation} \mid \text{message}) \cdot p(\text{message}) \]
Noisy Channel Model

- Communication channels introduce errors.
- Our goal is to identify the intended message given the observation.
- \[ p(\text{message} | \text{observation}) = p(\text{observation} | \text{message}) \cdot p(\text{message}) \]
Representing Sequences

- $p(\text{msg} | \text{obs}) = p(\text{obs} | \text{msg}) \ p(\text{msg})$

- Assume you have a way to generate a large set of candidate messages.

- $p(\text{msg})$ allows rescoring hypotheses.

- $p(\text{message}) = p(w_1, w_2, \ldots, w_n)$
Language Modeling

- Calculate the likelihood of a sequence
  \[ p(\text{message}) = p(w_1, w_2, \ldots, w_n) \]
- There are many possible messages.
- If message is a k-lengthed sequence of items drawn from vocabulary V, there are \( V^k \) possible messages.
- Equally likely messages have a likelihood of \( 1/V^k \) (approaches 0)
Language Modeling

- \( p(\text{message}) = p(\mathbf{w}) = p(w_1, w_2, ..., w_n) \)

- \( p(\mathbf{w}) = p(w_0)p(w_1|w_0)p(w_2|w_1, w_0) \cdots p(w_n|w_{n-1}, ..., w_0) \)

- \( p(\mathbf{w}) = \prod p(w_i \mid \text{history}) = \prod p(w_i \mid h) \)

- Floating point underflow is more than likely.

- \( \log p(\mathbf{w}) = \sum \log p(w_i \mid \text{history}) \)
Evaluating Language Models

• Log Likelihood:
  \[ \log p(w) = \sum \log p(w_i | \text{history}) \]

• Perplexity:
  \[ 2 - \sum_x p(x) \log p(x) \quad \text{and} \quad 2 - \sum_i \frac{1}{N} \log p_m(w_i) \]

• Cross-entropy:
  \[ H(p, q) = - \sum_x p(x) \log q(x) \]
Evaluating Language Models

- **Log Likelihood:**
  \[
  \log p(w) = \sum \log p(w_i | \text{history})
  \]

- **Perplexity:**
  \[
  2^{-\sum_x p(x) \log p(x)} \quad \text{Expected message length under a different distribution}
  \]

- **Cross-entropy:**
  \[
  H(p, q) = -\sum_x p(x) \log q(x)
  \]
Markov Assumption
Markov Assumption

- The future is independent of the past given the present.
Markov Assumption

- The future is independent of the past given the present.
Markov Assumption

- The future is independent of the past given the present.

- The “present” needs to encode all information about the past.
N-gram Modeling

- Conditional independence assumption
- Under an n-gram model
  - $p(w_i | h) = p(w_i | w_{i-1}, w_{i-2}, ..., w_{i-n+1})$
  - The history is limited to the previous N-1 tokens.
  - $p(w) = \prod p(w_i | h)$
    $$= p(w_1)p(w_2|w_1)...p(w_n|w_{n-1},...,w_1)\prod p(w_i | w_{i-1}, w_{i-2}, ..., w_{i-n+1})$$

- $p(w) = N(w) / N$
- $p(w_i | w_{i-1}) = N(w_{i-1}, w_i) / N(w_{i-1})$
Maximum Likelihood

- \( p(w) = \frac{N(w)}{N} \)

- Maximum Likelihood of a Discrete Distribution

\[
p(\vec{x}; \vec{\theta}) = \prod_{i=1}^{n} \theta_{i}^{x_{i}} \quad \sum_{i} \theta_{i} = 1
\]

- Worth looking at. Not going to derive here.

- Fairly direct application of Lagrange multipliers.
Speech Recognition

- \[ p(\text{words} \mid \text{speech}) = p(\text{speech} \mid \text{words}) \ p(\text{words}) \]

Speech is usually represented as short frame (10ms) features extracted from the waveform.
Speech Recognition

- $p(\text{words} \mid \text{speech}) = p(\text{speech} \mid \text{words}) \ p(\text{words})$

Speech is usually represented as short frame (10ms) features extracted from the waveform.
Speech Recognition

- \( p(\text{words} \mid \text{speech}) = p(\text{speech} \mid \text{words}) \ p(\text{words}) \)

Speech is usually represented as short frame (10ms) features extracted from the waveform.
Character Recognition

- $p(\text{words} \mid \text{image}) = p(\text{image} \mid \text{words}) \ p(\text{words})$

http://www.lookupinfo.org/images/macro_eye_30042010.jpg
Machine Translation

- \[ p(\text{target} \mid \text{source}) = p(\text{source} \mid \text{target}) \cdot p(\text{target}) \]
Information Retrieval

- Compare a query and a document.
- Compare cross-entropy of a query and document

\[ H(q, d) = - \sum_i p_q(q_i) \log p_d(q_i) \]
Sequence Classification

• construct multiple sequence models
• compare perplexity or likelihood

\[ i^* = \arg\max_i p(\vec{w}; \theta_i) \]
Sequence Modeling

• Model parameter space rather than observation space.

• What are the parameters in a language model?

• Consider cross entropy

\[ H(o, m) = - \sum_w p_o(w) \log p_m(w) \]

• Let’s not call H entropy but “score”

\[ score = \frac{1}{N} \sum_w weight(w) \]

• weight(w) is the value of a word under this model, aka the parameters of the model.
Aside: Machine Learning Evaluation

- Training vs. Evaluation
- You can do whatever you want with training data.
- Evaluation data should be unseen.
- Estimation of generalization performance.
- Don’t train any parameters on evaluation data.
- Better not to evaluate more than once. (this is usually impossible)
Aside: Machine Learning Evaluation

- Development data (or tuning data)
- Subset of training data used to tune parameters.
- Typically 10-33% of training data, but this varies by application, experimenter and amount of available data.
Overfitting

• What is overfitting?

• How can you tell the difference between a bad model and an overfit model?
Overfitting

- Identify optimal parameter on development data.

http://upload.wikimedia.org/wikipedia/commons/thumb/1/1f/Overfitting_svg.svg/500px-Overfitting_svg.svg.png
Definition of Overfitting

• Overfitting occurs when the model is capturing idiosyncrasies of the specific sample, rather than properties of the population.
  • The model describes the noise rather than the signal.

• Stability
  • An appropriately fit model is stable under samples of the training data
  • An overfit model will be inconsistent

• Performance
  • A good model has low test error
  • An overfit model has high test error
Unseen N-grams

• Given a large enough N or a small enough corpus, the probability of observing a specific N-gram approaches zero.

• How do you avoid letting your likelihood flatline?
Smoothing

• Set the probability of any n-gram to be $> 0$

• If it’s legal in the language, the model can score it.
Types of smoothing

- Differ by their estimation of the probability of unseen events.
  - Laplace
  - Good-Turing
  - Knesser-Ney
Laplace Smoothing

- Pretend you’ve seen everything once.
- add 1

\[
p(w) = \frac{N(w)}{N} \quad \text{MLE} \quad \quad p(w) = \frac{N(w) + 1}{N + V} \quad \text{Laplace}
\]
Laplace Smoothing

- How much is every OOV term increased?
  \[ \frac{1}{N + V} \]

- How much is an IV term decreased?
  \[ \frac{N(w)}{N} - \frac{N(w) + 1}{N + V} \]

N=1000, V=50
Good Turing

• Intuition:
  • I see 10 blue cars, 3 black cars, 2 red cars, 1 yellow car, 1 purple car, 1 gold car
  • What’s the likelihood of me seeing a new color? $p(\text{oov})$

• Good Turing
  • Use the estimate of things you’ve seen once to approximate things you’ve seen zero times.
Good Turing

- \( N_c = \) number of things seen \( c \) times.

**MLE**

\[
p(w) = \frac{N(w)}{N}
\]

**GT**

\[
p(w) = \frac{c^*(w)}{N}
\]

\[
c(w)^* = \frac{(c(w) + 1) N_{c(w)} + 1}{N_{c(w)}}
\]

\[
p(\text{dots}) = \frac{N_1}{N} = \frac{3}{18}
\]

\[
c(\text{yellow})^*: \frac{2 \times 1}{3} = \frac{2}{3}
\]

\[
p(\text{yellow}) = \frac{2/3}{18} = \frac{2}{27}
\]
You want to model 3 grams, but some of them are too rare.

Can you estimate the likelihood of a 3 gram using 2 gram probabilities?

Backoff:

- Use the 3-gram model if you have enough evidence. (i.e. counts are high enough)
- Otherwise use the 2-gram model, otherwise, unigram.
Backoff

- Unigram Model. \( p(w_i) \)
- Bigram Model. \( p(w_i|w_{i-1}) \)
- Trigram Model. \( p(w_i|w_{i-1}, w_{i-2}) \)
- Backoff Model.

\[
p^*(w_i|w_{i-1}, w_{i-2}) = \lambda_1 p(w_i|w_{i-1}, w_{i-2}) + \lambda_2 p(w_i|w_{i-1}) + \lambda_3 p(w_i)
\]

Tune based on held out data. \( \sum_i \lambda_i = 1 \)
“Los Angeles” is common.

High Bigram Probability

“Angeles” only appears after “Los”

\[ p(\text{Angeles} \mid \cdot) \] is very low.

Unigram model will over estimate “Angeles”
Kneser Ney

- Let the count assigned to each unigram be the number of different words it follows.

\[
N_{1+}(\bullet w_i) = |\{w_{i-1} : c(w_{i-1}w_i)\}|
\]

\[
N_{1+}(\bullet\bullet) = \sum_{w_i} N_{1+}(\bullet w_i)
\]

Unigram counts

\[
p_{kn}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}
\]

Unigram likelihood

\[
p_{kn}(w_i | w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^i) - \delta, 0\}}{\sum_{w_i} c(w_{i-n+1}^i)}\]

discount weighting

\[
\delta + \frac{\sum_{w_i} c(w_{i-n+1}^i)}{\sum_{w_i} c(w_{i-n+1}^i)} N_{1+}(w_{i-n+1}^{i-1}\bullet)p_{kn}(w_i | w_{i-n+2}^{i-1})
\]

dsmler model
Interpolation

- Linear interpolation
- Used in backoff.

\[ x = \lambda x_1 + (1 - \lambda)x_2 \quad \lambda \in [0, 1] \]

- Interpolating two models

\[ p(x) = \lambda p_1(x) + (1 - \lambda)p_2(x) \]

- Interpolating multiple models

\[ p(x) = \sum_i \lambda_i p_i(x) \quad \sum_i \lambda_i = 1 \]
Generalized Interpolation

- Interpolate models
  \[ p(x) = \sum_{i} \lambda_i p_i(x) \quad \sum_{i} \lambda_i = 1 \]

- Interpolate weights for each history
  \[ p^*(w|h) = \sum_{i} \lambda_i(h) p(w|h) \quad \sum_{i} \lambda_i(h) = 1 \]

- \( \lambda_i \) corresponds to the “relevance” of a history under a given model
Domain Adaptation

- What is a “domain”?
  - politics vs. sports
  - formal vs. informal
  - topic
- At what point of granularity do we stop?
Trade offs

• Specificity
  • Good performance on limited domain
  • Risk:
    • If you split too narrowly, you won’t have enough data to effectively model what you care about.

• Generalization
  • Modest performance on across domains
  • Risk:
    • Modeling distinct groups can blur distinctions and may violate model assumptions.
Adaptation

- Linear Interpolation for Adaptation
- Train one model per domain.
- Interpolate all models.

\[ p(w|h) = \sum_i \lambda_i p_i(w|h) \]
Background Model

- Train N + 1 models.
  - one per domain and one universal model.
- Interpolate each model and the universal model.

\[ p_i(w|h) = \lambda_i p_i(w|h) + (1 - \lambda_i) p_{ubm}(w|h) \]
Class-based Language Models

- Talked about these in the context of broadclassing

\[ \tilde{c} \] DET N VB ADV ADJ CON VB ADV

\[ \tilde{w} \] THE MAN IS VERY FAT BUT RUNS QUICKLY
Class Language Models

- General form of a class language model

\[ p(w|h) = p(w|w_h, c, c_h) \]

- Version we’ve seen before “Brown model”

\[ p(w|h) = p(w|c, c_h) = p(w|c)p(c|c_h) \]

Conditional independence assumption
Differing assumptions

Model M

\[ p(w) = \sum \prod_{c} \prod_{i} p(c_i|c_{i-1}, w_{i-1}) \prod_{i} p(w_i|c_i w_{i-1}) \]

Model L

\[ p(w) = \sum \prod_{c} \prod_{i} p(c_i|c_{i-1} w_{i-1}) \prod_{i} p(w_i|c_i w_{i-1} c_{i-1}) \]

all combinations of \( c_{i-1}, w_{i-1}, c_{i-1} w_{i-1} \) à la backoff
Class Language Models

- **Model M**

\[ p(w) = \sum_c \prod_i p(c_i|c_{i-1}, w_{i-1}) \prod_i p(w_i|c_i w_{i-1}) \]

- **Model L**

\[ p(w) = \sum_c \prod_i p(c_i|c_{i-1} w_{i-1}) \prod_i p(w_i|c_i w_{i-1} c_{i-1}) \]
Maximum Entropy Language Model

• Loads of features.
  • Unigram.
  • N-gram
  • Intermediate distance.
  • Existence in history \( p(w|w \in h) \)
  • Related words \( p(w | \text{related}(w,h)) \)

\[
p(w|h) = \sum_i \lambda_i p_i(w|h)
\]
Maximum Entropy Language Model

- Each of these are measurable.
  - $p_{\text{unigram}}(w) = \frac{N(w)}{N}$
  - $p_{\text{bigram}}(w|w-1) = \frac{N(w-1,w)}{N(w)}$
  - $p_{\text{near}}(w,w_2,d) = \frac{|\{\text{pos}(w) - \text{pos}(w_2)\} < d|}{N(w)}$

- Each have a general form:
  - $f(w,h)$ - a relation between $w$ and $h$ which is true for some pairs and false for others.
  - $P(w,h)$ - an expected number of observations
  - $K$ - expected value
Maximum Entropy Language Model

- Each of these are measurable.
  - \( p_{\text{unigram}}(w) = \frac{N(w)}{N} \)
    - \( f(w,h) : w \) is in the vocabulary (existential)
    - \( p(w,h) : \) number of words that are \( w \)
    - \( K = \frac{N(w)}{N} \)
  - \( p_{\text{bigram}}(w|w-1) = \frac{N(w-1,w)}{N(w)} \)
    - \( f(w,h) : w \) is \( wi \)
    - \( p(w,h) : w \) immediately follows \( w-1 \)
    - \( K : \frac{N(w-1,w)}{N(w)} \)
  - \( p_{\text{near}}(w,w2,d) = \frac{|\{|\text{pos}(w) - \text{pos}(w2)| < d\}|}{N(w)} \)
    - \( f(w,h) : w \) is \( wi \)
    - \( p(w,h) : w \) is within \( d \) words of \( w2 \)
    - \( K : \{|\text{pos}(w) - \text{pos}(w2)| < d\}|/N(w) \)
Maximum Entropy

- Approach to combine information sources (i.e. features)

- Maximum Entropy Principle
  - All information sources are constraints
  - Among all available sources, incorporate that which has the highest Entropy under the current model.
  - This has the most information I don’t already have.
Maximum Entropy Language Model

- Interpolation gives a way to combine the weights of these models.

\[
p_{comb}(w|h) = \sum_i \lambda_i p_i(w|h) \quad \text{By Def'n}
\]

\[
\sum p_{comb}(\bar{w}) f_i(w) = K_i \quad \text{By Derivation}
\]

\[
p_{comb}(\bar{w}) = \prod_i \theta_i^{f_i(w)}
\]

- Solve by identifying theta.

- Maximum Entropy is identical to Logistic Regression
Language Model as Linear Model

\[ p_{comb}(\vec{w}) = \prod_i \theta_i^{f_i(w)} \]

\[ \log p_{comb}(\vec{w}) = \sum_i f_i(w) \log \theta_i \]

• \(P(w)\) is estimated by a log linear combination of feature functions.

• \(P(w)\) is a linear model over features \(f_i(w)\)
Language Model as Linear Model

\[ \log p_{comb}(\overrightarrow{w}) = \sum_i f_i(\overrightarrow{w}) \log \theta_i \]

- This is a general way to describe a language model
- There are \( i \) features defined over \( w \) and \( h \).
- Each has a weight
Domain Adaptation of Linear Models

- Interpolation can now happen at the feature level as well as the model level.

\[ \theta_c = \gamma \theta_i + (1 - \gamma) \theta_j \]

- This is another route towards Generalized Interpolation
Frustratingly Easy Domain Adaptation

- Adaptation by feature augmentation.
- Consider two domains, A and B.
  - Same form of language model
- Construct three features spaces.
  - One shared
  - One for each of domain A and B.

\[ \Phi(X) = \langle x, x, 0 \rangle \quad \Phi(X) = \langle x, 0, x \rangle \]

- Project X to this new combined feature space.
- Learn new weights.
  - These new weights decouple the projected dimensions
### Frustratingly Easy Domain Adaptation

<table>
<thead>
<tr>
<th>Genre</th>
<th>Interp</th>
<th>Augment</th>
</tr>
</thead>
<tbody>
<tr>
<td>broadcast news</td>
<td>2.21</td>
<td>1.98</td>
</tr>
<tr>
<td>broadcast conversation</td>
<td>4.01</td>
<td>3.47</td>
</tr>
<tr>
<td>newswire</td>
<td>3.79</td>
<td>3.39</td>
</tr>
<tr>
<td>blog</td>
<td>2.33</td>
<td>2.12</td>
</tr>
<tr>
<td>usenet</td>
<td>2.10</td>
<td>1.91</td>
</tr>
<tr>
<td>telephone speech</td>
<td>0.44</td>
<td>0.32</td>
</tr>
</tbody>
</table>

- NB: It’s not *always* better

---

**Name Tagging Error Rates**

Monday, October 7, 13
Recurrent Neural Nets

Previous Hidden Layer
Alternate Visualization

Model description - recurrent NNLM

- **Input layer** $w(t)$ and **output layer** $y(t)$ have the same dimensionality as the vocabulary (10K - 200K).
- **Hidden layer** $s(t)$ is orders of magnitude smaller (50 - 1000 neurons).
- **$U$** is the matrix of weights between input and hidden layer, **$V$** is the matrix of weights between hidden and output layer.

Without the recurrent weights $W$, this model would be a bigram neural network language model.
Unfolding

Standard “deep” Backpropagation still works for training.

Only difference is that U and W are tied.
Unfolding

Backpropagation through time

Standard “deep” Backpropagation still works for training.

Only difference is that U and W are tied
Toolkits

- SRILM
- MITLM
  - https://code.google.com/p/mitlm/
- NLTK
  - http://nltk.org/
- RNNLM toolkit
- Maximum Entropy Language Model (ME-LM)
Discriminative Language Models

• Everything so far has rested on a generative assumption to decompose $p(w)$

• Why do we care about evaluating the likelihood of a sequence $w$?

• ASR, OCR, MT

• What do these have in common?

• We have a “correct” answer
Discriminative Language Models

- Actual score doesn’t matter.
- Relative score matters a lot.
- Measure distance between correct and alternatives
  - \( d(X) = -p(X, W) + \frac{1}{m-1} \sum p(x,w) \)
- Now optimize this Loss Function
Optimization

Tune weights based on whether a sequence has been correctly identified or incorrectly identified.

\[
\begin{align*}
C(w_{i-2}, w_{i-1}, w_i) &= C(w_{i-2}, w_{i-1}, w_i) + \alpha \\
C(w_{i-1}, w_i, w_{i+1}) &= C(w_{i-1}, w_i, w_{i+1}) + \alpha \\
C(w_i, w_{i+1}, w_{i+2}) &= C(w_i, w_{i+1}, w_{i+2}) + \alpha \\
C(w_{j-2}, w_{j-1}, w_j) &= C(w_{j-2}, w_{j-1}, w_j) - \beta \\
C(w_{j-1}, w_j, w_{j+1}) &= C(w_{j-1}, w_j, w_{j+1}) - \beta \\
C(w_j, w_{j+1}, w_{j+2}) &= C(w_j, w_{j+1}, w_{j+2}) - \beta
\end{align*}
\]
Aside: Discriminative vs. Generative

- **Generative Models.**
  - Multiple classes.
  - A lot of things are average.
  - \( p(\text{observation} \mid \text{class}) \)

- **Discriminative Models.**
  - Explicitly distinguish class A from class B.
  - \( p(\text{class} \mid \text{obs}) \)
  
  \[
p(c \mid o) = \frac{\text{score}(c \mid o)}{\sum_i \text{score}(c_i \mid o)}
  \]