Accelerated Natural Language Processing
Lecture 5
N-gram models, entropy

Sharon Goldwater
(some slides based on those by Alex Lascarides and Philipp Koehn)

25 September 2018
Recap: Language models

- **Language models** tell us $P(\tilde{w}) = P(w_1 \ldots w_n)$: *How likely to occur is this sequence of words?*

  Roughly: *Is this sequence of words a “good” one in my language?*
Example uses of language models

- Machine translation: reordering, word choice.

\[ P_{LM}(\text{the house is small}) > P_{LM}(\text{small the is house}) \]
\[ P_{LM}(\text{I am going home}) > P_{LM}(\text{I am going house}) \]
\[ P_{LM}(\text{We’ll start eating}) > P_{LM}(\text{We shall commence consuming}) \]

- Speech recognition: word choice:

\[ P_{LM}(\text{morphosyntactic analyses}) > P_{LM}(\text{more faux syntactic analyses}) \]
\[ P_{LM}(\text{I put it on today}) > P_{LM}(\text{I putted onto day}) \]

But: How do systems use this information?
Today’s lecture:

• What is the Noisy Channel framework and what are some example uses?

• What is a language model?

• What is an n-gram model, what is it for, and what independence assumptions does it make?

• What are entropy and perplexity and what do they tell us?

• What’s wrong with using MLE in n-gram models?
Noisy channel framework

- Concept from Information Theory, used widely in NLP
- We imagine that the observed data (output sequence) was generated as:

\[
P(Y) \quad \text{P(X|Y)}
\]

Sharon Goldwater  ANLP Lecture 5 4
Noisy channel framework

• Concept from Information Theory, used widely in NLP

• We imagine that the observed data (output sequence) was generated as:

\[
P(Y) \rightarrow P(X|Y) \rightarrow P(X)
\]

<table>
<thead>
<tr>
<th>Application</th>
<th>Y</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech recognition</td>
<td>true words</td>
<td>acoustic signal</td>
</tr>
<tr>
<td>Machine translation</td>
<td>words in ( L_1 )</td>
<td>words in ( L_2 )</td>
</tr>
<tr>
<td>Spelling correction</td>
<td>true words</td>
<td>typed words</td>
</tr>
</tbody>
</table>
Example: spelling correction

- $P(Y)$: Distribution over the words (sequences) the user intended to type. A language model.

- $P(X|Y)$: Distribution describing what user is likely to type, given what they meant. Could incorporate information about common spelling errors, key positions, etc. Call it a noise model.

- $P(X)$: Resulting distribution over what we actually see.

- Given some particular observation $x$ (say, effert), we want to recover the most probable $y$ that was intended.
Noisy channel as probabilistic inference

- Mathematically, what we want is $\arg\max_y P(y|x)$.
  - Read as “the $y$ that maximizes $P(y|x)$”

- Rewrite using Bayes’ Rule:

  \[
  \arg\max_y P(y|x) = \arg\max_y \frac{P(x|y)P(y)}{P(x)} = \arg\max_y P(x|y)P(y)
  \]
Noisy channel as probabilistic inference

So to recover the best $y$, we will need

- **a language model** $P(Y)$: relatively task-independent.

- **a noise model** $P(X|Y)$, which depends on the task.
  - acoustic model, translation model, misspelling model, etc.
  - won’t discuss here; see courses on ASR, MT.

Both are normally trained on corpus data.
You may be wondering

If we can train $P(X|Y)$, why can’t we just train $P(Y|X)$? Who needs Bayes’ Rule?

• Answer 1: sometimes we do train $P(Y|X)$ directly. Stay tuned...

• Answer 2: training $P(X|Y)$ or $P(Y|X)$ requires input/output pairs, which are often limited:
  – Misspelled words with their corrections; transcribed speech; translated text

But LMs can be trained on huge unannotated corpora: a better model. Can help improve overall performance.
Estimating a language model

- $Y$ is really a sequence of words $\vec{w} = w_1 \ldots w_n$.

- So we want to know $P(w_1 \ldots w_n)$ for big $n$ (e.g., sentence).

- What will not work: try to directly estimate probability of each full sentence.
  - Say, using MLE (relative frequencies): $C(\vec{w})/(\text{tot } \# \text{ sentences})$.
  - For nearly all $\vec{w}$ (grammatical or not), $C(\vec{w}) = 0$.
  - A **sparse data** problem: not enough observations to estimate probabilities well.
A first attempt to solve the problem

Perhaps the simplest model of sentence probabilities: a unigram model.

• Generative process: choose each word in sentence independently.

• Resulting model: 
\[
\hat{P}(\vec{w}) = \prod_{i=1}^{n} P(w_i)
\]
A first attempt to solve the problem

Perhaps the simplest model of sentence probabilities: a unigram model.

• Generative process: choose each word in sentence independently.

• Resulting model: \[ \hat{P}(\vec{w}) = \prod_{i=1}^{n} P(w_i) \]

• So, \( P(\text{the cat slept quietly}) = P(\text{the quietly cat slept}) \)
A first attempt to solve the problem

Perhaps the simplest model of sentence probabilities: a unigram model.

• Generative process: choose each word in sentence independently.

• Resulting model: \( \hat{P}(\vec{w}) = \prod_{i=1}^{n} P(w_i) \)

• So, \( P(\text{the cat slept quietly}) = P(\text{the quietly cat slept}) \)
  – Not a good model, but still a model.

• Of course, \( P(w_i) \) also needs to be estimated!
MLE for unigrams

• How to estimate $P(w)$, e.g., $P(\text{the})$?

• Remember that MLE is just relative frequencies:

$$P_{\text{ML}}(w) = \frac{C(w)}{W}$$

– $C(w)$ is the token count of $w$ in a large corpus
– $W = \sum_{x'} C(x')$ is the total number of word tokens in the corpus.
Unigram models in practice

- Seems like a pretty bad model of language: probability of word obviously *does* depend on context.

- Yet unigram (or bag-of-words) models are surprisingly useful for some applications.
  - Can model “aboutness”: topic of a document, semantic usage of a word
  - Applications: lexical semantics (disambiguation), information retrieval, text classification. (See later in this course)
  - But, for now we will focus on models that capture at least some syntactic information.
General N-gram language models

Step 1: rewrite using chain rule:

\[ P(\vec{w}) = P(w_1 \ldots w_n) \]
\[ = P(w_n|w_1, w_2, \ldots, w_{n-1})P(w_{n-1}|w_1, w_2, \ldots, w_{n-2}) \ldots P(w_1) \]

• Example: \( \vec{w} = \) the cat slept quietly yesterday.

\[ P(\text{the, cat, slept, quietly, yesterday}) = \]
\[ P(\text{yesterday}|\text{the, cat, slept, quietly}) \cdot P(\text{quietly}|\text{the, cat, slept}) \cdot P(\text{slept}|\text{the, cat}) \cdot P(\text{cat}|\text{the}) \cdot P(\text{the}) \]

• But for long sequences, many of the conditional probs are also too sparse!
General N-gram language models

Step 2: make an independence assumption:

\[ P(\vec{w}) = P(w_1 \ldots w_n) \]
\[ = P(w_n|w_1, w_2, \ldots, w_{n-1}) P(w_{n-1}|w_1, w_2, \ldots, w_{n-2}) \ldots P(w_1) \]
\[ \approx P(w_n|w_{n-2}, w_{n-1}) P(w_{n-1}|w_{n-3}, w_{n-2}) \ldots P(w_1) \]

- **Markov** assumption: only a finite history matters.

- Here, two word history (**trigram** model):
  \[ w_i \text{ is cond. indep. of } w_1 \ldots w_{i-3} \text{ given } w_{i-1}, w_{i-2}. \]
  \[ P(\text{the, cat, slept, quietly, yesterday}) \approx \]
  \[ P(\text{yesterday}|\text{slept, quietly}) \cdot P(\text{quietly}|\text{cat, slept}). \]
  \[ P(\text{slept}|\text{the, cat}) \cdot P(\text{cat}|\text{the}) \cdot P(\text{the}) \]
Trigram independence assumption

• Put another way, a trigram model assumes these are all equal:
  
  – $P(\text{slept}|\text{the cat})$
  – $P(\text{slept}|\text{after lunch the cat})$
  – $P(\text{slept}|\text{the dog chased the cat})$
  – $P(\text{slept}|\text{except for the cat})$

  because all are estimated as $P(\text{slept}|\text{the cat})$

• Not always a good assumption! But it does reduce the sparse data problem.
Another example: bigram model

- Bigram model assumes one word history:

\[ P(\vec{w}) = P(w_1) \prod_{i=2}^{n} P(w_i|w_{i-1}) \]

- But consider these sentences:

\[ \begin{array}{cccc}
  w_1 & w_2 & w_3 & w_4 \\
  1 & \text{the} & \text{cats} & \text{slept} & \text{quietly} \\
  2 & \text{feeds} & \text{cats} & \text{slept} & \text{quietly} \\
  3 & \text{the} & \text{cats} & \text{slept} & \text{on} \\
\end{array} \]

- What’s wrong with (2) and (3)? Does the model capture these problems?
Example: bigram model

• To capture behaviour at beginning/end of sentence, we need to augment the input:

\[
\begin{align*}
&\begin{array}{ccccccc}
  w_0 & w_1 & w_2 & w_3 & w_4 & w_5 \\
  (1) & <s> & \text{the} & \text{cats} & \text{slept} & \text{quietly} & </s> \\
  (2) & <s> & \text{feeds} & \text{cats} & \text{slept} & \text{quietly} & </s> \\
  (3) & <s> & \text{the} & \text{cats} & \text{slept} & \text{on} & </s>
\end{array}
\end{align*}
\]

• That is, assume \( w_0 = <s> \) and \( w_{n+1} = </s> \) so we have:

\[
P(\tilde{w}) = P(w_0) \prod_{i=1}^{n+1} P(w_i|w_{i-1}) = \prod_{i=1}^{n+1} P(w_i|w_{i-1})
\]
Estimating N-Gram Probabilities

• Maximum likelihood (relative frequency) estimation for bigrams:
  - How many times we saw $w_2$ following $w_1$,
    out of all the times we saw anything following $w_1$:

\[
P_{ML}(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1, \cdot)} = \frac{C(w_1, w_2)}{C(w_1)}
\]
Estimating N-Gram Probabilities

• Similarly for trigrams:

\[ P_{ML}(w_3|w_1, w_2) = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} \]

• Collect counts over a large text corpus
  – Millions to billions of words are usually easy to get
  – (trillions of English words available on the web)
Evaluating a language model

• Intuitively, trigram model captures more context than bigram model, so should be a “better” model.

• That is, more accurately predict the probabilities of sentences.

• But how can we measure this?
Entropy

- Definition of the entropy of a random variable $X$:

$$H(X) = \sum_x -P(x) \log_2 P(x)$$

- Intuitively: a measure of uncertainty/disorder

- Also: the expected value of $-\log_2 P(X)$
Reminder: logarithms

\[ \log_a x = b \iff a^b = x \]
Entropy Example

One event (outcome)

\[ P(a) = 1 \]
\[ H(X) = - \log_2 1 \]
\[ = 0 \]
Entropy Example

2 equally likely events:

\[ P(a) = 0.5 \]
\[ P(b) = 0.5 \]

\[ H(X) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 \]
\[ = - \log_2 0.5 \]
\[ = 1 \]
Entropy Example

4 equally likely events:

\[
P(a) = 0.25 \quad H(X) = - 0.25 \log_2 0.25 - 0.25 \log_2 0.25
\]

\[
P(b) = 0.25 \quad - 0.25 \log_2 0.25 - 0.25 \log_2 0.25
\]

\[
P(c) = 0.25 \quad = - \log_2 0.25
\]

\[
P(d) = 0.25 \quad = 2
\]
Entropy Example

3 equally likely events and one more likely than the others:

\[
P(a) = 0.7 \\
P(b) = 0.1 \\
P(c) = 0.1 \\
P(d) = 0.1
\]

\[
H(X) = -0.7 \log_2 0.7 - 0.1 \log_2 0.1
\]

\[
= -0.7 \log_2 0.7 - 0.1 \log_2 0.1
\]

\[
= -0.7 \log_2 0.7 - 0.3 \log_2 0.1
\]

\[
= -(0.7)(-0.5146) - (0.3)(-3.3219)
\]

\[
= 0.36020 + 0.99658
\]

\[
= 1.35678
\]
Entropy Example

3 equally likely events and one much more likely than the others:

\[ H(X) = -0.97 \log_2 0.97 - 0.01 \log_2 0.01 \]
\[ - 0.01 \log_2 0.01 - 0.01 \log_2 0.01 \]
\[ = -0.97 \log_2 0.97 - 0.03 \log_2 0.01 \]
\[ = -(0.97)(-0.04394) - (0.03)(-6.64) \]
\[ = 0.04262 + 0.19932 \]
\[ = 0.24194 \]
Entropy take-aways

Entropy of a uniform distribution over $N$ outcomes is $\log_2 N$:

- $H(X) = 0$
- $H(X) = 1$
- $H(X) = 2$
- $H(X) = 2.585$
Entropy take-aways

Any **non-uniform** distribution over $N$ outcomes has **lower** entropy than the corresponding uniform distribution:

$$H(X) = 2$$

$$H(X) = 1.35678$$

$$H(X) = 0.24194$$
Entropy as y/n questions

How many yes-no questions (bits) do we need to find out the outcome?

- Uniform distribution with $2^n$ outcomes: $n$ q’s.

- Other cases: entropy is the average number of questions per outcome in a (very) long sequence of outcomes, where questions can consider multiple outcomes at once.
Entropy as encoding sequences

• Assume that we want to encode a sequence of events $X$

• Each event is encoded by a sequence of bits

• For example
  – Coin flip: heads = 0, tails = 1
  – 4 equally likely events: $a = 00$, $b = 01$, $c = 10$, $d = 11$
  – 3 events, one more likely than others: $a = 0$, $b = 10$, $c = 11$
  – Morse code: $e$ has shorter code than $q$

• Average number of bits needed to encode $X \geq$ entropy of $X$
The Entropy of English

- Given the start of a text, can we guess the next word?

- Humans do pretty well: the entropy is only about 1.3.

- But what about $N$-gram models?
  - Ideal language model would match the true entropy of English.
  - The closer we get to that, the better the model.
  - Put another way, a good model assigns high prob to real sentences (and low prob to everything else).
How good is the LM?

- **Cross entropy** measures how well model $M$ predicts the data.

- For data $w_1 \ldots w_n$ with large $n$, well approximated by:

$$H_M(w_1 \ldots w_n) = -\frac{1}{n} \log_2 P_M(w_1 \ldots w_n)$$

  - Avg neg log prob our model assigns to each word we saw

- Or, **perplexity**:

$$PP_M(\vec{w}) = 2^{H_M(\vec{w})}$$
Perplexity

- On paper, there’s a simpler expression for perplexity:

\[ PP_M(\vec{w}) = 2^{H_M(\vec{w})} = 2^{-\frac{1}{n} \log_2 P_M(w_1...w_n)} = 2^{\log_2 P_M(w_1...w_n) - \frac{1}{n}} = P_M(w_1...w_n)^{-\frac{1}{n}} \]

- 1 over the geometric average of the probabilities of each \( w_i \).

- But in practice, when computing perplexity for long sequences, we use the version with logs (see week 3 lab for reasons...).
Example: trigram (Europarl)

<table>
<thead>
<tr>
<th>prediction</th>
<th>$P_{ML}$</th>
<th>$-\log_2 P_{ML}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{ML}(i</td>
<td>&lt;s&gt;&lt;s&gt;)$</td>
<td>0.109</td>
</tr>
<tr>
<td>$P_{ML}(\text{would}</td>
<td>&lt;s&gt;i)$</td>
<td>0.144</td>
</tr>
<tr>
<td>$P_{ML}(\text{like}</td>
<td>i \text{ would})$</td>
<td>0.489</td>
</tr>
<tr>
<td>$P_{ML}(\text{to}</td>
<td>\text{would like})$</td>
<td>0.905</td>
</tr>
<tr>
<td>$P_{ML}(\text{commend}</td>
<td>\text{like to})$</td>
<td>0.002</td>
</tr>
<tr>
<td>$P_{ML}(\text{the}</td>
<td>\text{to commend})$</td>
<td>0.472</td>
</tr>
<tr>
<td>$P_{ML}(\text{rapporteur}</td>
<td>\text{commend the})$</td>
<td>0.147</td>
</tr>
<tr>
<td>$P_{ML}(\text{on}</td>
<td>\text{the rapporteur})$</td>
<td>0.056</td>
</tr>
<tr>
<td>$P_{ML}(\text{his}</td>
<td>\text{rapporteur on})$</td>
<td>0.194</td>
</tr>
<tr>
<td>$P_{ML}(\text{work}</td>
<td>\text{on his})$</td>
<td>0.089</td>
</tr>
<tr>
<td>$P_{ML}(.</td>
<td>\text{his work})$</td>
<td>0.290</td>
</tr>
<tr>
<td>$P_{ML}(&lt;/s&gt;</td>
<td>\text{work .})$</td>
<td>0.99999</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td>2.634</td>
</tr>
</tbody>
</table>
## Comparison: 1–4-Gram

<table>
<thead>
<tr>
<th></th>
<th>unigram</th>
<th>bigram</th>
<th>trigram</th>
<th>4-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>6.684</td>
<td>3.197</td>
<td>3.197</td>
<td>3.197</td>
</tr>
<tr>
<td>would</td>
<td>8.342</td>
<td>2.884</td>
<td>2.791</td>
<td>2.791</td>
</tr>
<tr>
<td>like</td>
<td>9.129</td>
<td>2.026</td>
<td>1.031</td>
<td>1.290</td>
</tr>
<tr>
<td>to</td>
<td>5.081</td>
<td>0.402</td>
<td>0.144</td>
<td>0.113</td>
</tr>
<tr>
<td>commend</td>
<td>15.487</td>
<td>12.335</td>
<td>8.794</td>
<td>8.633</td>
</tr>
<tr>
<td>the</td>
<td>3.885</td>
<td>1.402</td>
<td>1.084</td>
<td>0.880</td>
</tr>
<tr>
<td>rapporteur</td>
<td>10.840</td>
<td>7.319</td>
<td>2.763</td>
<td>2.350</td>
</tr>
<tr>
<td>on</td>
<td>6.765</td>
<td>4.140</td>
<td>4.150</td>
<td>1.862</td>
</tr>
<tr>
<td>his</td>
<td>10.678</td>
<td>7.316</td>
<td>2.367</td>
<td>1.978</td>
</tr>
<tr>
<td>work</td>
<td>9.993</td>
<td>4.816</td>
<td>3.498</td>
<td>2.394</td>
</tr>
<tr>
<td>.</td>
<td>4.896</td>
<td>3.020</td>
<td>1.785</td>
<td>1.510</td>
</tr>
<tr>
<td>&lt;/s&gt;</td>
<td>4.828</td>
<td>0.005</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>average</td>
<td>8.051</td>
<td>4.072</td>
<td>2.634</td>
<td>2.251</td>
</tr>
<tr>
<td>perplexity</td>
<td>265.136</td>
<td>16.817</td>
<td>6.206</td>
<td>4.758</td>
</tr>
</tbody>
</table>
Unseen N-Grams

• What happens when I try to compute $P(\text{consuming}|\text{shall commence})$?
  – Assume we have seen \textit{shall commence} in our corpus
  – But we have never seen \textit{shall commence consuming} in our corpus
Unseen N-Grams

• What happens when I try to compute $P(\text{consuming}|\text{shall commence})$?
  
  – Assume we have seen \textit{shall commence} in our corpus
  
  – But we have never seen \textit{shall commence consuming} in our corpus
  
  $\rightarrow P(\text{consuming}|\text{shall commence}) = 0$

• Any sentence with \textit{shall commence consuming} will be assigned probability 0

  The guests shall commence consuming supper
  Green inked shall commence consuming garden the
The problem with MLE

- MLE estimates probabilities that make the observed data maximally probable
- by assuming anything unseen cannot happen (and also assigning too much probability to low-frequency observed events).
- It over-fits the training data.
- We tried to avoid zero-probability sentences by modelling with smaller chunks ($n$-grams), but even these will sometimes have zero prob under MLE.

Next time: smoothing methods, which reassign probability mass from observed to unobserved events, to avoid overfitting/zero probs.
Questions for review:

- What is the Noisy Channel framework and what are some example uses?

- What is a language model?

- What is an n-gram model, what is it for, and what independence assumptions does it make?

- What are entropy and perplexity and what do they tell us?

- What’s wrong with using MLE in n-gram models?