language models and corpora
Lecture 1-2 overview

• This lecture:
  • Linguistic data,
  • Language models, and
  • $N$-grams

• Some slides may be based on content from Bob Carpenter, Dan Klein, Roger Levy, Josh Goodman, Dan Jurafsky, Christopher Manning, and Gerald Penn.
Statistics: what are we counting?

• Almost all statistics are based on simple counting.
• **What are we counting?**

> First, we shape our tools and thereafter our tools shape us.

• **Tokens:** *n.pl. instances* of words or punctuation (13).

• **Types:** *n.pl. ‘kinds’* of words or punctuation (10).
Other confounding factors

• How should we count the following token pairs?
  • (run, runs) (verb conjugation)
  • (happy, happily) (adjective vs. adverb)
  • (fragme(1)nt, fragme(1)nt) (spoken stress)
  • (realize, realise) (spelling)
  • (We, we) (capitalization)

• How do we count speech disfluencies?
  • e.g., I _uh main_-mainly do data processing
  • Answer: It depends on your task.
  • e.g., if you’re doing summarization, you usually don’t care about ‘uh’.
Does it matter how we count things?

• Answer: See Lecture 2-2 on feature extraction.

• Preview: *yes, it matters.*
  • E.g., to diagnose Alzheimer’s disease from a patient’s speech, you may want to measure:
    • Excessive pauses (disfluencies),
    • Excessive word type repetition, and
    • Simplistic or short sentences.

• What is (perhaps) the simplest task in which one can use linguistic statistics?
Word prediction

• Guess the next word...
  • ...not be judged by the colour of their [???

• Famous quotes are part of world knowledge, but what do we do about everyday speech?

• You can do quite well by counting how often certain tokens occur given their contexts.
Word prediction with $N$-grams

- **$N$-grams**: *n.pl. token* sequences of length $N$.

- The fragment ‘*the colour of their*’ contains the following 2-grams (i.e., ‘*bigrams’*):
  - *(the colour), (colour of), (of their)*

- The next bigram **must** start with ‘*their*’.

- What word is most likely to follow ‘*their*’?
Use of \( N \)-gram models

• Given the **probabilities** of \( N \)-grams, we can compute the **conditional probabilities** of possible subsequent words.

• E.g., \( P(\text{their } \text{houses}) > P(\text{their } \text{skin}) \) \( \therefore \)
  \[
P(\text{houses}|\text{their}) > P(\text{skin}|\text{their})
\]

Then we would predict:

‘not be judged by the colour of their **houses**’.

(We’ll soon see how to avoid such mistakes (if this is one))
Applications of \( N \)-gram models

• Being able to predict the next word (or other linguistic unit) in a sequence is extremely useful.

• This simple idea lies at the core of
  • Automatic speech recognition,
  • Handwriting and character recognition,
  • Spelling correction,
  • Machine translation,
  • Et cetera.

From where do \( N \)-gram probabilities come?
Corpora

- **Corpus**: *n.* A body of language data of a particular sort (*pl.* corpora).

- Most **useful** corpora occur **naturally**
  - e.g., newspaper articles, telephone conversations, multilingual transcripts of the United Nations.

- We use corpora to gather statistics; more is better (typically between $10^7$ and $10^{12}$ tokens).
Historically notable corpora

- **Penn treebank**: Syntactically annotated Brown, plus others incl. 1989 *Wall Street Journal*.
- **Switchboard corpus**: 120 hours ≈ 2.4M tokens. 2.4K spoken telephone conversations between US English speakers.
Additional notable corpora

- **Hansard corpus**: Canadian parliamentary proceedings, French/English bilingual.
- **Gutenberg project**: 33K free eBooks, several languages. [http://www.gutenberg.org](http://www.gutenberg.org)
- **Google corpus**: Index of between $10^{11}$ and $10^{12}$ 5-word sequences (13,588,391 word types (incl. numbers, names, misspellings, etc.)) [http://ngrams.googlelabs.com/](http://ngrams.googlelabs.com/)
Building models

• Given lots of data from the real world, we can build a model, which is a set of parameters (e.g., bigram probabilities), that either describes the data or predicts/infers future/unseen data.
General process

1. We gather a big and relevant **training** corpus.
2. We learn our **parameters** (e.g., probabilities) from that corpus to build our **model**.
3. Once that model is fixed, we use those probabilities to evaluate **testing** data.
General process

• Often, **training data** consists of 80% to 90% of the available data.

• **Testing data** is *not* used for training but comes from the same source.
  • It often consists of the remaining 10% to 20% of the available data.

• **K-fold cross validation**: *n.* splitting all data into *K* partitions and iteratively testing on each after training on the rest (report means and variances).
Language models

• **Language model**: \( n \). The statistical model of a language (obviously).
  • e.g., probabilities of words in an *ordered* sequence.
    i.e., \( P(w_1, w_2, \ldots, w_n) \)

• What do we **do** with a language model?
Language models

• Language models can **score** and **sort** sentences.
  • e.g., \( P(I \text{ saw a van}) \gg P(\text{eyes awe of an}) \)

• Language models do **not** judge grammaticality.
  • e.g., \( P(\text{artichokes intimidate zippers}) \approx 0 \)

• Language models require suspension of disbelief.
  • i.e., can a sentence **really** have a probability?

• **How do we calculate** \( P(\ldots) \)?
Frequency statistics

- **Term count** (*Count*) of term \( w \) in corpus \( C \) is the number of tokens of term \( w \) in \( C \).
  \[
  \text{Count}(w, C)
  \]

- **Relative frequency** (*\( F_C \*)) is defined relative to the total number of tokens in the corpus, \( \|C\| \).
  \[
  F_C(w) = \frac{\text{Count}(w, C)}{\|C\|}
  \]

- In theory, \( \lim_{\|C\| \to \infty} P(w) = F_C(w) \). (the “frequentist view”)
The chain rule

• Recall,

\[ P(A, B) = P(B|A)P(A) \]
\[ P(B|A) = \frac{P(A, B)}{P(A)} \]

• This extends to longer sequences, e.g.,

\[ P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C) \]

• Or, in general,

\[ P(x_1, x_2, ..., x_n) = P(x_1)P(x_2|x_1) \cdots P(x_n|x_1, x_2, ..., x_{n-1}) \]
The chain rule

- **Sequences** can be demarked by subscripts as in:

\[
P(x_{1:n}) = P(x_1)P(x_2|x_1)P(x_3|x_{1:2}) \cdots P(x_n|x_{1:(n-1)})
\]

\[
= P(x_1) \prod_{k=2}^{n} P(x_k|x_{1:(k-1)})
\]

- The **chain rule** applies to sequences of **words**, e.g.,

\[
P(\text{not be judged by}) = P(\text{not})P(\text{be}|\text{not})P(\text{judged}|\text{not be}) \cdot P(\text{by}|\text{not be judged})
\]
Very simple predictions

• Let’s return to word prediction.
• We want to know the probability of the next word given the previous words in a sequence.

• We can approximate conditional probabilities by counting occurrences in large corpora of data.
  • E.g., \( P(\text{food} \mid \text{I want Chinese}) = \frac{P(\text{I want Chinese food})}{P(\text{I want Chinese})} \approx \frac{\text{Count}(\text{I want Chinese food})}{\text{Count}(\text{I want Chinese})} \)

Why?
Hint: Corpus size
Problem with the chain rule

• There are many ($\infty$?) possible sentences.
• In general, we won’t have enough data to compute reliable statistics for long prefixes
  • E.g.,
    \[
    P(\text{dog}|I \text{ wonder how many rubber balls we can feed to your cousin Bill's ugly little}) = \\
    \frac{P(I \text{ wonder } \ldots \text{ little dog})}{P(I \text{ wonder } \ldots \text{ little})} = 0
    \]

• How can we avoid \{0, \infty\}-probabilities?
Independence!

• We can simplify things if we’re willing to break from the distant past and focus on recent history.
  • e.g.,
    \[ P(\text{dog} | \text{I wonder how many rubber balls we can feed to your cousin Bill's ugly little}) \approx P(\text{dog} | \text{ugly little}) \approx P(\text{dog} | \text{little}) \]
  
• I.e., we assume statistical independence.
Markov assumption

• Assume each observation only depends on a short linear history of length $L$.

$$P(w_n|w_{1:(n-1)}) \approx P(w_n|w_{(n-L):(n-1)})$$

• Bigram version:

$$P(w_n|w_{1:(n-1)}) \approx P(w_n|w_{n-1})$$
Berkeley Restaurant Project corpus

• Let’s compute simple $N$-gram models of speech queries about restaurants in Berkeley California.
  • E.g.,
    • *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*
Example bigram counts

- Out of 9222 sentences,
  - e.g., “I want” occurred 827 times

<table>
<thead>
<tr>
<th>$\text{Count}(w_{t-1}, w_t)$</th>
<th>$w_t$</th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
<td></td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Example bigram probabilities

- Obtain likelihoods by dividing bigram counts by unigram counts.

Unigram counts:

<table>
<thead>
<tr>
<th></th>
<th>l</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

| $P(w_t|w_{t-1})$ | I     | want | to   | eat  | Chinese | food  | lunch | spend |
|-----------------|-------|------|------|------|---------|-------|-------|-------|
| l               | 0.002 | 0.33 | 0    | 0.0036 | 0       | 0     | 0     | 0.00079 |

$$P(want|l) \approx \frac{\text{Count}(\text{I want})}{\text{Count}(\text{I})} = \frac{827}{2533} \approx 0.33$$

$$P(\text{spend}|l) \approx \frac{\text{Count}(\text{I spend})}{\text{Count}(\text{I})} = \frac{2}{2533} \approx 7.9 \times 10^{-4}$$
Example bigram probabilities

• Obtain likelihoods by dividing bigram counts by unigram counts.

Unigram counts:

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

\[
P(w_t|w_{t-1})\]

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00079</td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.0011</td>
</tr>
<tr>
<td>to</td>
<td>0.00083</td>
<td>0</td>
<td>0.0017</td>
<td>0.28</td>
<td>0.00083</td>
<td>0</td>
<td>0.0025</td>
<td>0.087</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>0.0027</td>
<td>0</td>
<td>0.021</td>
<td>0.0027</td>
<td>0.056</td>
<td>0</td>
</tr>
<tr>
<td>Chinese</td>
<td>0.0063</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.52</td>
<td>0.0063</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0.014</td>
<td>0</td>
<td>0.014</td>
<td>0</td>
<td>0.00092</td>
<td>0.0037</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0059</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0029</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>0.0036</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bigram estimate of an unseen phrase

- We can **string** bigram probabilities together to estimate the probability of **whole sentences**.
  - We need to use the **start** (<s>) and **end** (</s>) tags here.

- E.g.,

\[
P(<s> I want english food </s>) \approx \\
P(I | <s>) \cdot P(want | I) \cdot \\
P(english | want) \cdot P(food | english) \cdot \\
P(</s> | food)
\]

\[
\approx 0.000031
\]
N-grams as linguistic knowledge

• Despite their simplicity, N-gram probabilities can **crudely** capture **interesting facts** about language and the world.

• E.g.,
  
  \[ P(\text{english}|\text{want}) = 0.0011 \]
  \[ P(\text{chinese}|\text{want}) = 0.0065 \]

  \[ P(\text{to}|\text{want}) = 0.66 \]
  \[ P(\text{eat}|\text{to}) = 0.28 \]
  \[ P(\text{food}|\text{to}) = 0 \]

  \[ P(i|<s>) = 0.25 \]
Probability of a corpus

• There are a few ways to evaluate the probability of an entire corpus, $P(\text{Corpus})$.

• For now, let’s keep using our current approach.
  • i.e., treat the corpus as one loooong sentence.

• Q: Why is this useful?
  A: Computing $P(\text{Corpus})$ helps us adjust or estimate its parameters (e.g., $P(\text{to}|\text{want})$).
Maximum likelihood estimate

- We estimate $P(\cdot)$ given a particular corpus, e.g., Brown.
- $P(\cdot)$ may be a bad estimate for some other corpus but it is the estimate that makes Brown most likely.

If

$$P_1(\cdot) \geq P_j(\cdot) \quad \forall j$$

then

$P_1$ is the best model of the Brown corpus.
Maximum likelihood estimate

• **Maximum likelihood estimate (MLE)** of parameters $\theta$ in a model $M$, given training data $T$ is the estimate that maximizes the likelihood of the training data using the model.

• e.g., $T$ is the Brown corpus, $M$ is the bigram and unigram tables $\theta_{(to|want)}$ is $P(to|want)$.

• In fact, we have been doing MLE all along with our simple counting.
Shannon’s method

• We can use a language model to generate random sequences.

• We ought to see sequences that are somehow similar to those we used for training.

• This approach is attributed to Claude Shannon.
Shannon’s method – unigrams

- **Sample** a model according to its probability.
- For unigrams, keep picking tokens.
- e.g., imagine throwing darts at this:

```
the
Cat
in
Hat
</s>
```
Problem with unigrams

- Unigrams give high probability to odd phrases.

  e.g., \( P(\text{the the the the the the} \langle /s \rangle) = P(\text{the})^5 \cdot P(\langle /s \rangle) \)
  
  \( > P(\text{the Cat in the Hat} \langle /s \rangle) \)
Shannon’s method – bigrams

• Bigrams have *fixed* context once that context has been sampled.
  • e.g., $P(\cdot \mid \text{the})$
Something about which to think

- Do we need to include the sentence-ending token `<s>` among our available tokens with Shannon’s method?
Shannon’s method on Shakespeare

<table>
<thead>
<tr>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
<th>Quadrigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have</td>
<td>• What means, sir. I confess she? Then all sorts, he is trim, captain.</td>
<td>• Sweet prince, Falstaff shall die. Harry of Monmouth’s grave.</td>
<td>• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch.</td>
</tr>
<tr>
<td>• Hill he late speaks; or! A more to leg less first you enter</td>
<td>• Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.</td>
<td>• This shall forbid it should be branded, if renown made it empty.</td>
<td>• Will you not tell me who I am?</td>
</tr>
<tr>
<td>• Are where exeunt and sighs have rise excellency took of.. Sleep knave we. Near; vile like.</td>
<td>• What we, hat got so she that I rest and sent to scold and nature bankrupt nor the first gentleman?</td>
<td>• Indeed the duke; and had a very good friend.</td>
<td>• It cannot be but so.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Indeed the short and the long. Marry. ‘tis a noble Lepidus.</td>
</tr>
</tbody>
</table>
Shakespeare as a corpus

- 884,647 tokens, vocabulary of $V = 29,066$ types.
- Shakespeare produced about 300,000 bigram types out of $V^2 \approx 845M$ possible bigram types.
  - $\therefore$ 99.96% of possible bigrams were never seen (i.e., they have 0 probability in the bigram table).

- Quadrigrams appear more similar to Shakespeare because, for increasing context, there are fewer possible next words, given the training data.
  - E.g., $P(\text{Gloucester} | \text{seek the traitor}) = 1$
### Shannon and the Wall Street Journal

<table>
<thead>
<tr>
<th><strong>Unigram</strong></th>
<th>Months the my and issue of year foreign new exchange’s September were recession exchange new endorsed a acquire to six executives.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bigram</strong></td>
<td>Last December through the way to preserve the Hudson corporation N.B.E.C. Taylor would seem to complete the major central planners one point five percent of U.S.E. has already old M.X. corporation of living on information such as more frequently fishing to keep her.</td>
</tr>
<tr>
<td><strong>Trigram</strong></td>
<td>They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.</td>
</tr>
</tbody>
</table>
The end of week 1

• Next week:
  • What to do about 0-probability $N$-grams.
  • Zipf’s Law.
  • Feature extraction.
  • Part-of-speech tagging.