108-NLP @ NCTU

e-Lab 3

Boaz

Good morning!

Please make sure your microphone/video is OFF

We’ll take a break or two 🕘🕙🕚🕛

Outline

Lab 2

N-Grams

Demo

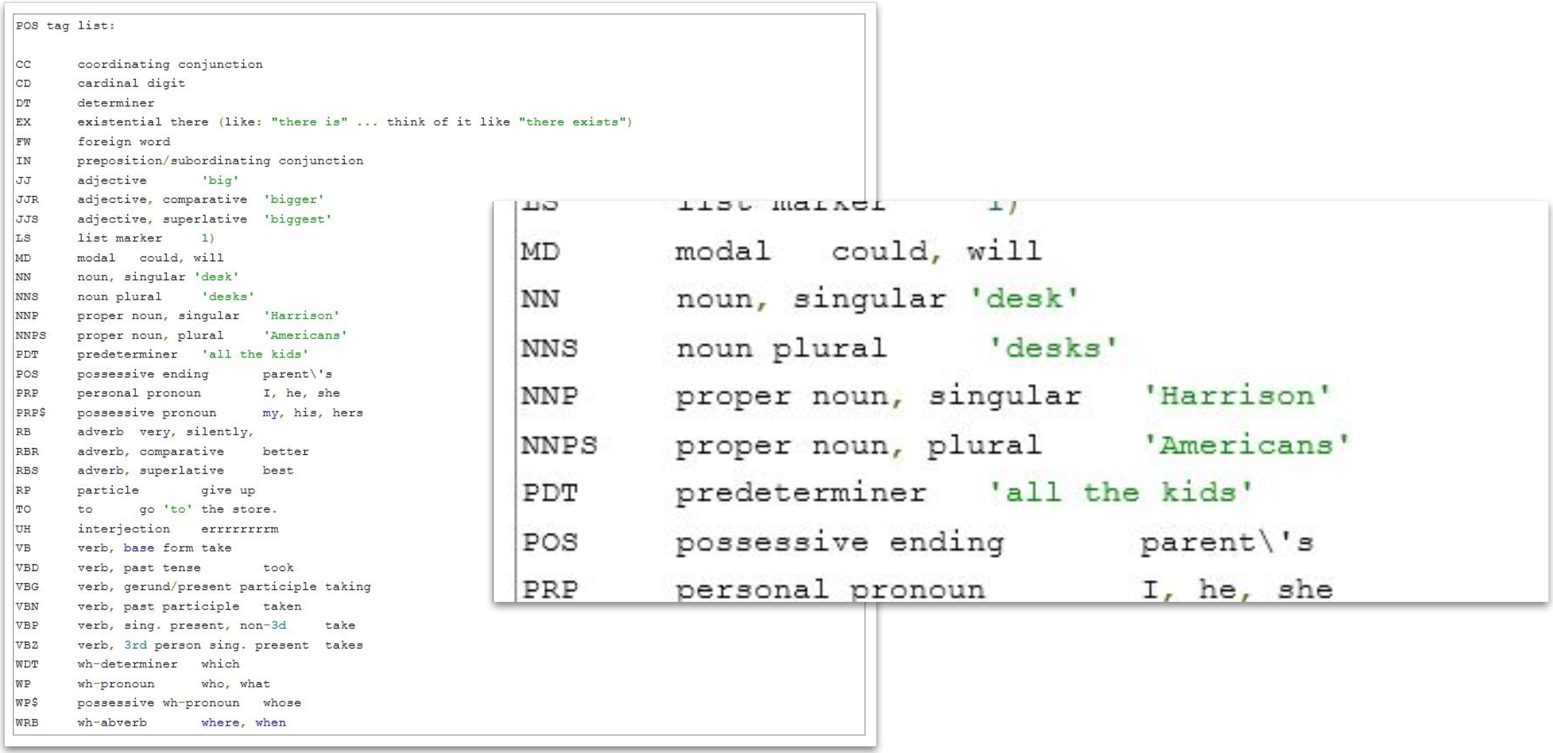
Assignment

Python is not JavaScript :)

Take advantage of Python!

https://docs.python-guide.org/writing/style/

Lab 2 Part 1 - Proper Nouns



```
POS tag list:

CC      coordinating conjunction
CD      cardinal digit
DT      determiner
EX      existential there (like: "there is" ... think of it like "there exists")
FW      foreign word
IN      preposition/subordinating conjunction
JJ      adjective       'big'
JJR     adjective, comparative  'bigger'
JJS     adjective, superlative  'biggest'
LS      list marker     1)
MD      modal   could, will
NN      noun, singular 'desk'
NNS     noun plural     'desks'
NNP     proper noun, singular   'Harrison'
NNPS    proper noun, plural     'Americans'
PDT     predeterminer   'all the kids'
POS     possessive ending       parent\'s
PRP     personal pronoun        I, he, she
PRP$    possessive pronoun      my, his, hers
RB      adverb  very, silently,
RBR     adverb, comparative     better
RBS     adverb, superlative     best
RP      particle        give up
TO      to      go 'to' the store.
UH      interjection    errrrrrrrm
VB      verb, base form take
VBD     verb, past tense        took
VBG     verb, gerund/present participle taking
VBN     verb, past participle   taken
VBP     verb, sing. present, non-3d     take
VBZ     verb, 3rd person sing. present  takes
WDT     wh-determiner   which
WP      wh-pronoun      who, what
WP$     possessive wh-pronoun   whose
WRB     wh-abverb       where, when
```

```
MD      modal   could, will
NN      noun, singular 'desk'
NNS     noun plural     'desks'
NNP     proper noun, singular   'Harrison'
NNPS    proper noun, plural     'Americans'
PDT     predeterminer   'all the kids'
POS     possessive ending       parent\'s
PRP     personal pronoun        I, he, she
```

Ask yourself: do the results MAKE SENSE?

```
[(('San', 'Qualcomm'), 5354), (('U.', 'S.'), 5354), (('S.', 'Supreme'), 5354),
(('Supreme', 'Court'), 5354), (('Lexmark', 'International'), 5354)]
```

```
[(('U.', 'S.'), 1354280435), (('Donald', 'Trump'), 244930323), (('New', 'York'),
193536141), (('Islamic', 'State'), 162978047), (('President', 'Donald'), 146460263)]
```

Lab 2 Part 2

Some news stories had missing content.

Here are the titles:

- All The Action At The Golden Globes After Parties
- All The Looks At The MTV Movie & TV Awards Red Carpet
- 44 Of The Most Iconic Pictures Of President Barack Obama

Why?

One way to solve this problem:

```
news = pd.read_csv("https://bit.ly/nlp-buzzfeed").fillna('')
```

This week’s topic: 2-gram models

sentence = <s>, w1, w2, w3, …, wi, wn, </s>

p(sentence) ~= p(w1|<s>) p(w2|w1) … w(</s>|wn)

Google Books Ngram Viewer

https://books.google.com/ngrams

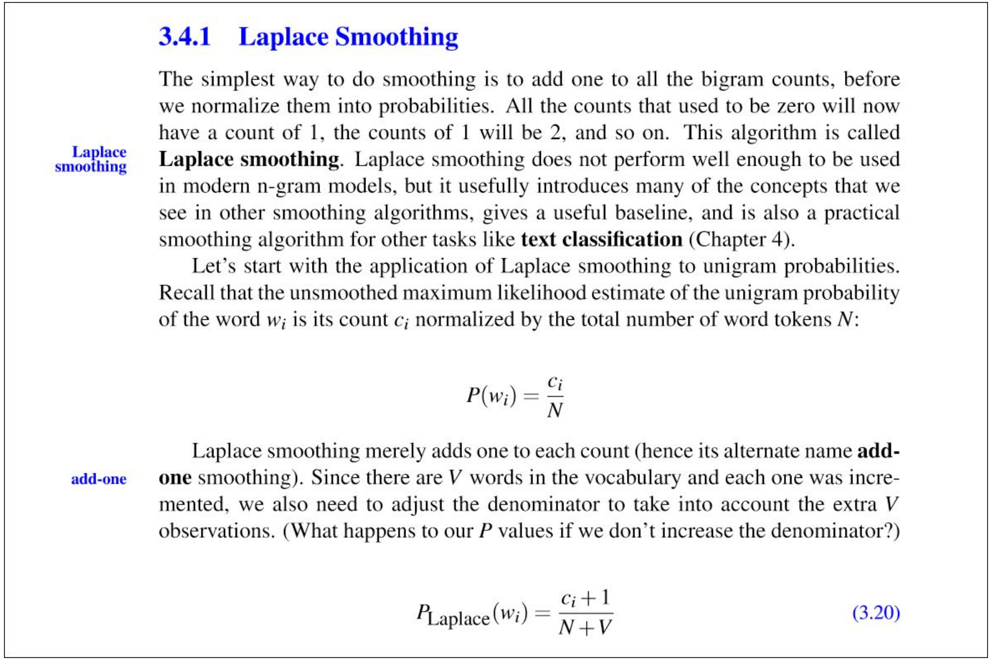
Back to n-gram: Smoothing

Phenomena: Need exponentially large training data as n grows

- Infrequent terms as <UNK>
- Laplace’s law: $P(w1,\ldots,wn) = (C(w1,\ldots wn)+1)/(N+B)$
  - Referred as adding one, giving a little bit of the prob space to unseen events, for large N gives too much prob to the unseen n-gram
- Hold-out: $P(w1,\ldots,wn) = Tr/(NrN)$

Resource

https://web.stanford.edu/~jurafsky/slp3/3.pdf



3.4.1 Laplace Smoothing

Laplace smoothing

The simplest way to do smoothing is to add one to all the bigram counts, before we normalize them into probabilities. All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on. This algorithm is called **Laplace smoothing**. Laplace smoothing does not perform well enough to be used in modern n-gram models, but it usefully introduces many of the concepts that we see in other smoothing algorithms, gives a useful baseline, and is also a practical smoothing algorithm for other tasks like **text classification** (Chapter 4).

Let's start with the application of Laplace smoothing to unigram probabilities. Recall that the unsmoothed maximum likelihood estimate of the unigram probability of the word w_i is its count c_i normalized by the total number of word tokens N:

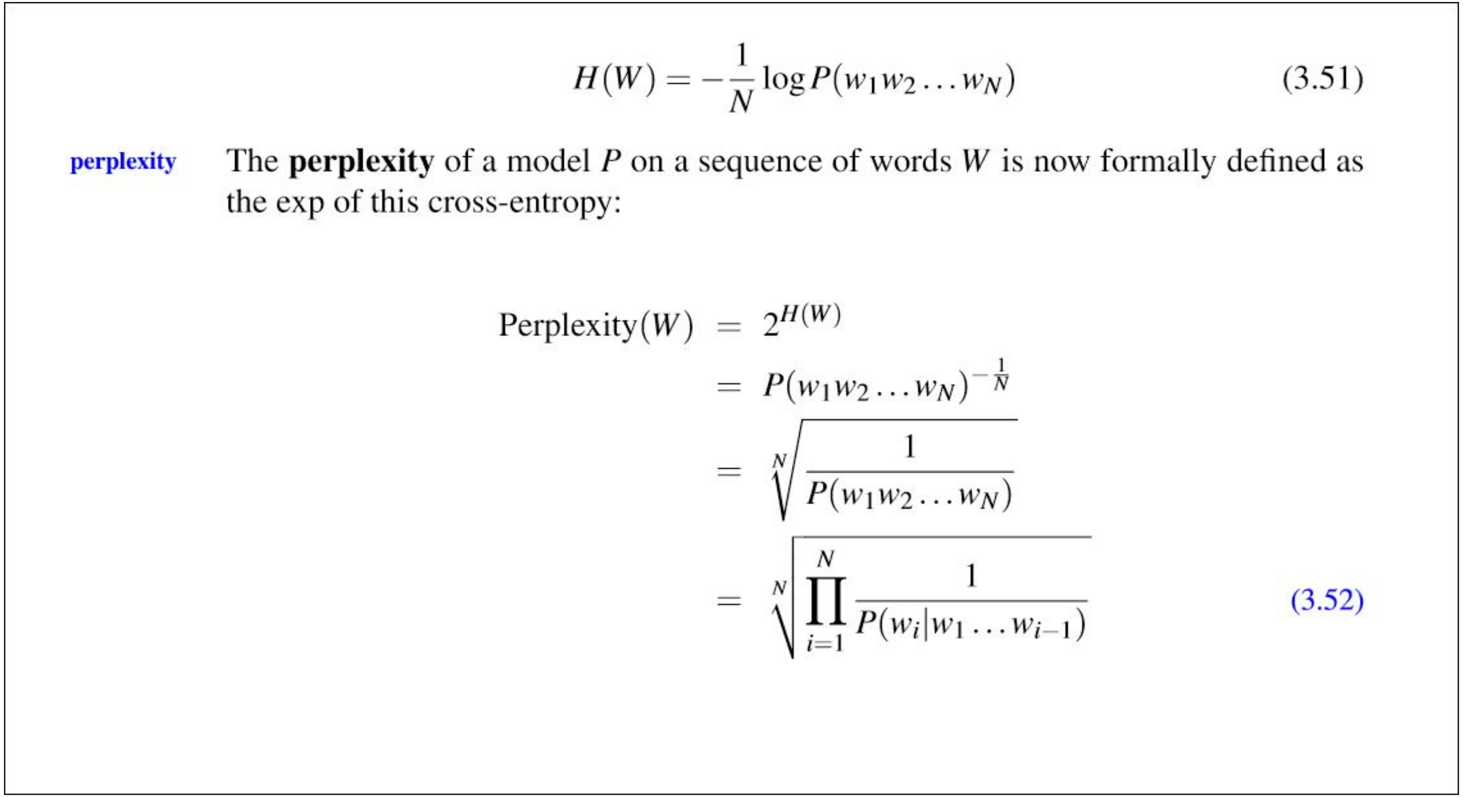
$$P(w_i) = \frac{c_i}{N}$$

add-one

Laplace smoothing merely adds one to each count (hence its alternate name **add-one** smoothing). Since there are V words in the vocabulary and each one was incremented, we also need to adjust the denominator to take into account the extra V observations. (What happens to our P values if we don't increase the denominator?)

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V} \qquad (3.20)$$

Cross-Entropy and Perplexity



$$H(W) = -\frac{1}{N}\log P(w_1w_2\dots w_N) \tag{3.51}$$

perplexity

The **perplexity** of a model P on a sequence of words W is now formally defined as the exp of this cross-entropy:

$$\begin{aligned}
\text{Perplexity}(W) &= 2^{H(W)} \\
&= P(w_1w_2\dots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1w_2\dots w_N)}} \\
&= \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i|w_1\dots w_{i-1})}}
\end{aligned} \tag{3.52}$$

Demo

Full

https://gist.github.com/bshmueli/42b80ef45fd1e704e8e93a76fb0a8210

Short

https://gist.github.com/bshmueli/b85471ad4f92b45ab91bab2b0f895c5a

Assignment

- Two parts (1 and 2).
- This week we will use the following data:
  - bit.ly/nlp-tweet-train for training the model
  - bit.ly/nlp-tweet-test for evaluating the model
- The data is in **JSON Lines** format. Each line is a JSON string.
- You can read more about JSON Lines in jsonlines.org

- TIP: you can use Pandas read_json(“filename”, lines=True) to read the data

Part 1

Build a 2-gram model for the Twitter train data:

- For tokenization, first convert to lowercase, then use the NLTK TweetTokenizer. That’s it!
- Don’t forget to add the ‘<s>’ and ‘</s>’ tokens to each tweet.
- Vocabulary: only frequent terms (terms appearing **3 or more** times in the train data). Replace infrequent terms (appearing only 1 or 2 times in the train data) by ‘<UNK>’ in both train and test data
- Use Laplace smoothing (identical to the one we used in the class)

Output (2 numbers)

- Average perplexity for:
  Train data tweets
  Test data tweets

Part 2

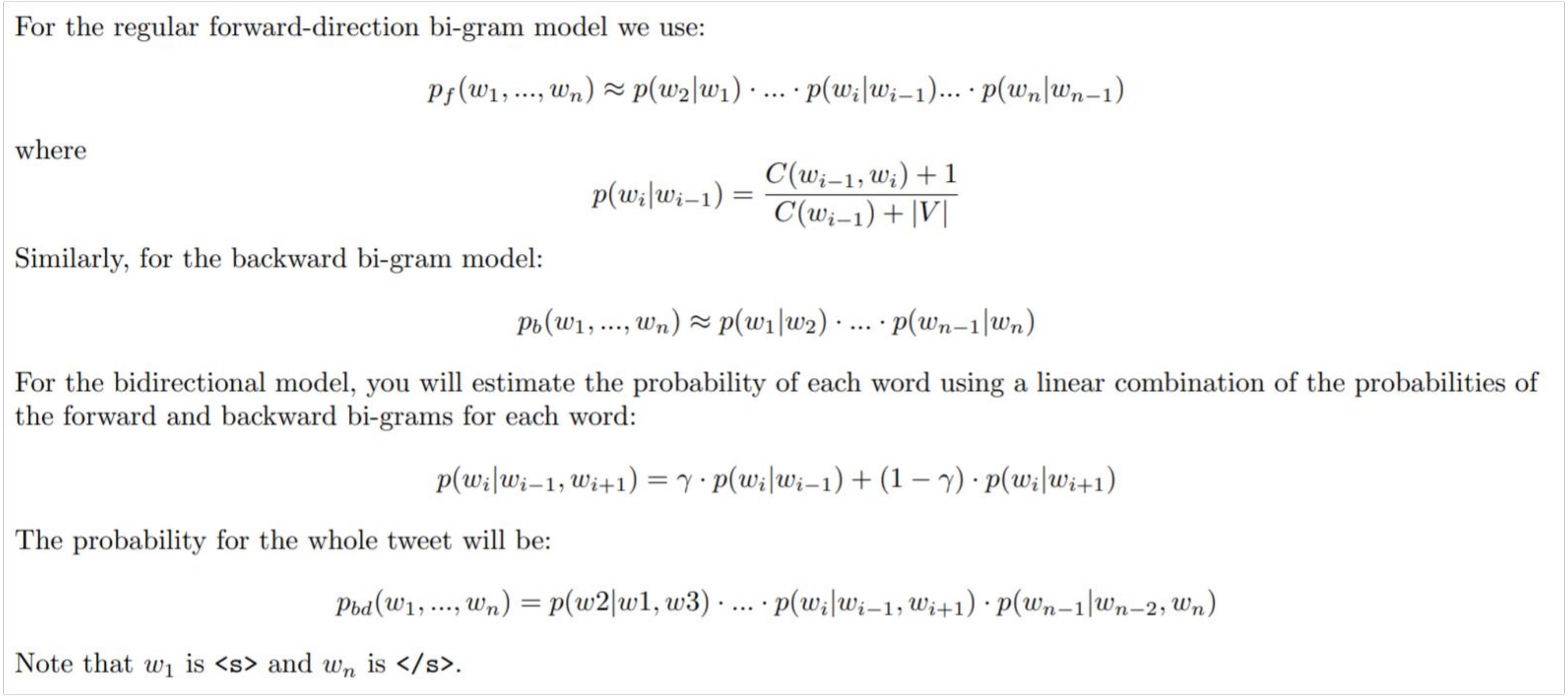
Build a **bi-directional** 2-gram model by training on the Twitter train data:

- Build 2-gram forward model (**identical to part 1**)
- Build 2-gram backward model (**identical to part 1**, but in the opposite direction)
- **NEW! See next slide for more information about the bi-directional language model**
- γ: a parameter between 0 and 1

Output (3 numbers)

- Print the γ that minimizes the perplexity of the Twitter test data (0.05 resolution)
- Average perplexity (at the optimal γ) for:
  Training data tweets
  Testing data tweets

Bi-directional Language Model



For the regular forward-direction bi-gram model we use:

$$p_f(w_1, ..., w_n) \approx p(w_2|w_1) \cdot ... \cdot p(w_i|w_{i-1}) ... \cdot p(w_n|w_{n-1})$$

where

$$p(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + |V|}$$

Similarly, for the backward bi-gram model:

$$p_b(w_1, ..., w_n) \approx p(w_1|w_2) \cdot ... \cdot p(w_{n-1}|w_n)$$

For the bidirectional model, you will estimate the probability of each word using a linear combination of the probabilities of the forward and backward bi-grams for each word:

$$p(w_i|w_{i-1}, w_{i+1}) = \gamma \cdot p(w_i|w_{i-1}) + (1 - \gamma) \cdot p(w_i|w_{i+1})$$

The probability for the whole tweet will be:

$$p_{bd}(w_1, ..., w_n) = p(w2|w1, w3) \cdot ... \cdot p(w_i|w_{i-1}, w_{i+1}) \cdot p(w_{n-1}|w_{n-2}, w_n)$$

Note that w_1 is `<s>` and w_n is `</s>`.

SOP for assignment submission

- □ Open your Python notebook
- □ Rename your notebook to lab3-<studentID>.ipynb
- □ “Restart and run all”
- □ Make sure the output is correct
- □ Download the Python file (File, Download .py)
- □ **Share the Google colab using “Get shareable link”; Copy link**
- □ Submit both link and file in e3:
    - □ Paste link into e3 (link starts with **https://colab.research.google.com/…**)
    - □ Upload file into e3 (file name is lab3-<studentID>.py)

Resources

https://web.stanford.edu/~jurafsky/slp3/3.pdf

How to find the perplexity of a corpus

https://rstudio-pubs-static.s3.amazonaws.com/115676_ab6bb49748c742b88127e8b5ce3e1298.html

https://www.youtube.com/watch?v=CTYqkWU8cBc

http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/lm.pdf

https://courses.cs.washington.edu/courses/csep517/18au/slides/csep517au18-LanguageModels.pdf

https://stackoverflow.com/questions/54941966/how-can-i-calculate-perplexity-using-nltk