U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

M.EIC

Natural Language Processing

Henrique Lopes Cardoso

FEUP / LIACC

hlc@fe.up.pt

Language Models

N-grams, Markov assumption, evaluation, perplexity, smoothing

Probabilistic Language Models

- A Language model (LM) assigns probabilities to sequences of words
  - Given a sequence of words, predict the next word by assigning a probability to each possibility
    - *Today, we are having meatballs for …*
  - Or assign a probability for an entire sentence, given a context

- Important to identify words in noisy, ambiguous input
  - Speech recognition: *What are you doing?* >> *Water you doing?*
  - Spelling or grammatical error correction: *Did you sea that?* → *Did you see that?*

Probabilistic Language Models

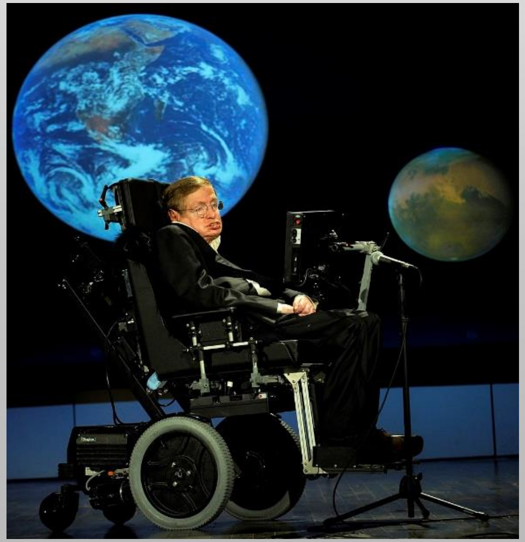
- Useful for machine translation

他	向	记者	介绍了	主要	内容
He	to	reporters	introduced	main	content

→

he introduced reporters to the main contents of the statement
he briefed to reporters the main contents of the statement
he briefed reporters on the main contents of the statement

- Augmentative and alternative communication systems (AAC)
- Contextualized predictive typing
- Summarization
- Question-answering
- Genre and language bias detection
- Natural language generation



Probability of a Sequence

- $P(w|h)$: probability of a word w given some history h
  - $P(the|its\ water\ is\ so\ transparent\ that)$

- Making use of frequency counts
  - $P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$
  - Language is creative, too many possible sentences!
  - Not enough data: any particular context might have never occurred before!

Probability of a Sequence

- Chain rule of probability:

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned}$$

  - w_1^n is a sequence of N words $w_1 \ldots w_n$

- Computing the probability of a word given its entire history is hard

  $\Rightarrow$ Approximate the history by using just the last few words!

N-grams

- An n-gram is a sequence of *N* words
  - 2-gram (bigram), a two-word sequence: “please turn”, “turn your”, “your homework”
  - 3-gram (trigram), a three-word sequence: “please turn your”, “turn your homework”
  - …

- N-gram models can be used to estimate the probability of the last word of an n-gram given the previous n-1 words

Markov Models

- Markov assumption: we can predict the probability of some future unit without looking too far into the past

- Bigram model: approximate $P\left(w_n|w_1^{n-1}\right)$ by $P(w_n|w_{n-1})$, i.e., using only the preceding word
  - $P(the|its\ water\ is\ so\ transparent\ that) \approx P(the|that)$
- General formulation for an n-gram approximation: $P\left(w_n|w_1^{n-1}\right) \approx P\left(w_n|w_{n-N+1}^{n-1}\right)$

- Probability of a sequence based on bigrams: $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$

Maximum Likelihood Estimation (MLE)

- Getting normalized counts (relative frequencies) from a corpus
  - Bigram model: $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$
  - General n-gram case: $P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$
- Example mini-corpus:

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

  - Some bigram probabilities:

$$P(\texttt{I}|\texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam}|\texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am}|\texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>}|\texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam}|\texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do}|\texttt{I}) = \tfrac{1}{3} = .33$$

Approximating the Probability of a Sequence

- Bigram counts (subset of words in a corpus):

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Bigram probabilities:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- Unigram counts (whole corpus):

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Approximating the Probability of a Sequence

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

$P(\texttt{i}\,|\,\texttt{<s>}) = 0.25$ $\quad P(\texttt{english}\,|\,\texttt{want}) = 0.0011$

$P(\texttt{food}\,|\,\texttt{english}) = 0.5$ $\quad P(\texttt{</s>}\,|\,\texttt{food}) = 0.68$

- $P(\text{<s> i want chinese food </s>}) =$

$$= P(\text{i}|\text{<s>})P(\text{want}|\text{i})P(\text{chinese}|\text{want})P(\text{food}|\text{chinese})P(\text{</s>}|\text{food}) =$$
$$= .25 \times .33 \times .0065 \times .52 \times .68 = .000189618$$

- $P(\text{<s> i want english food </s>}) =$

$$= P(\text{i}|\text{<s>})P(\text{want}|\text{i})P(\text{english}|\text{want})P(\text{food}|\text{english})P(\text{</s>}|\text{food}) =$$
$$= .25 \times .33 \times .0011 \times .5 \times .68 = .000030855$$

Linguistic Phenomena

- Syntactic
  - What comes after “eat” is usually a noun or an adjective
  - What comes after “to” is usually a verb
- Corpus-specific
  - Many sentences start with “I”: $P(\mathrm{i} | \text{<s>}) = .25$
- Cultural
  - $P(\mathrm{english}|\mathrm{want}) \ll P(\mathrm{chinese}|\mathrm{want})$, $P(\mathrm{food}|\mathrm{english}) < P(\mathrm{food}|\mathrm{chinese})$
  - Again, in this corpus
- …

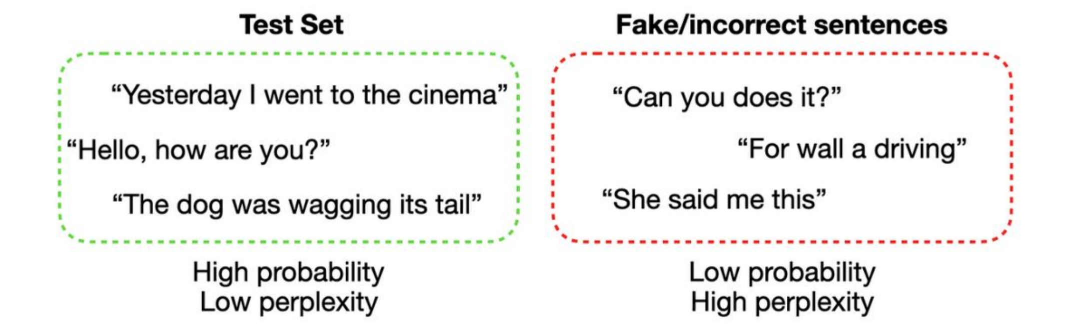
Practical Issues

- Trigram, 4-gram, 5-gram, …
  - Capture longer-distance dependencies, given enough training data
  - Hard to capture sentences like "The computer which I had just put into the machine room on the fifth floor crashed."
- Computing probabilities in log space
  - Multiplying many values between 0 and 1 gets very small numbers $\Rightarrow$ numerical underflow issues
  - Take the log probabilities and add them!
  - Adding in log space is equivalent to multiplying in linear space
    - $p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$

Evaluating Language Models

- Extrinsic evaluation
  - LMs can be embedded in more complex systems (translation, classification, speech recognition, …)
  - Does the performance on a target task improve by using a new language model?
  - Expensive, time consuming

- Intrinsic evaluation
  - Hold out a test set of the corpus
  - Which language model best predicts an unseen test set?
    - That is, assigns higher probability to each of its sentences
  - Perplexity: normalized inverse probability of the test set
    - Minimizing perplexity = maximizing probability



Intuition of Perplexity

- The Shannon Game: predicting the next word
  - *I always order pizza with cheese and ____*
  - *The 33rd President of the US was ____*
  - *I saw a ____*

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
…
fried rice 0.0001
…
and 1e-100

- A better language model assigns a higher probability to the word that actually occurs
  - Will a unigram model do a good job?
  - What about a bigram model?

Perplexity

- Perplexity: inverse probability of the test set, normalized by the number of words

$$\begin{aligned}\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

  - Applying the chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

- Computing the perplexity of W with a bigram model:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- Comparing different n-gram models
  - Train set with 38M words from the Wall Street Journal
  - Test set with 1.5M words

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Perplexity and Information Theory

- Perplexity can also be seen as a weighted branching factor of a language
  - Weigh each possible next word by its probability
- Example: a sequence of random digits
  - $P = 1/10$ for each digit, $PP = \left[\left(\frac{1}{10}\right)^N\right]^{-\frac{1}{N}} = \left(\frac{1}{10}\right)^{-1} = 10$ (unigram perplexity)
- What if 0 is more frequent than other digits in the training set?
  - $P(0) = 0.91, P(d) = 0.01, d \in [1..9]$
  - Test set: $0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0 \rightarrow PP = \left[0.91^9 \times 0.01\right]^{\frac{-1}{10}} = 1.73$
  - Test set: $0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 \rightarrow PP = \left[0.91 \times 0.01^9\right]^{\frac{-1}{10}} = 63.69$

Generating Text

- Generating random sentences from different n-gram models
  - Assign a slice of [0..1] to each possible next word, proportional to its relative probability
  - Generate a random value in [0..1] and choose the word whose slice includes that value
  - Repeat this process until the generated word is a special final token, e.g. </s>

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Shakespeare

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	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
3 gram	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

Wall Street Journal

Training Corpus

- Longer context brings more coherent sentences…
  - …but also less generalization
    - "*It cannot be but so*" is a direct transcription from *King John*
- The model strongly depends on its training corpus
  - Need to choose genre, language and dialect relevant to the task

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Shakespeare

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	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
3 gram	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

Wall Street Journal

Sparsity and Unknown Words

- N-grams will work well for word prediction if the training and test corpus look similar

- Zero probability n-grams
  - Missing in the training corpus but might occur in the test set!

Training set

denied the allegations:	5
denied the speculation:	2
denied the rumors:	1
denied the report:	1

Test set

denied the offer
denied the loan

  - $P(\textit{offer} \mid \textit{denied the})$ is 0!

$\Rightarrow$ We are underestimating the probability!

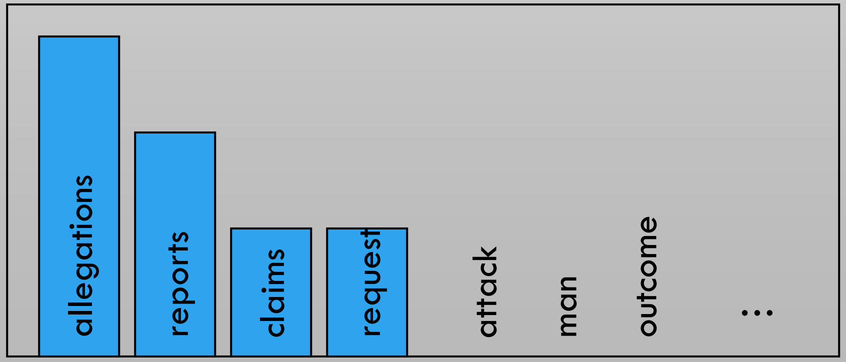
$\Rightarrow$ We cannot compute perplexity (divide by 0)!

- Out of vocabulary (OOV) words
  - Open vocabulary: we model unknown words in the test set by adding a pseudo-word <UNK>
  - Closed vocabulary: we assume every possible word of interest is known in advance (word list), and convert any OOV word in the training set to <UNK>

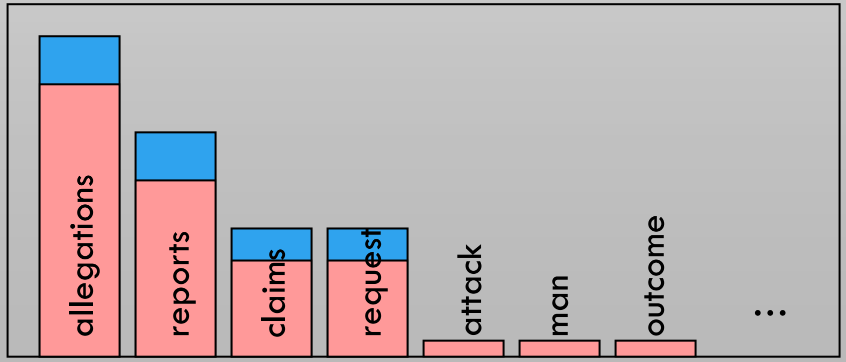
Smoothing

- Avoid assigning zero probability to unseen events
  - For instance, bigrams that appear in the test set but not in the training set
- Shave off a bit of probability mass from some more frequent events and give it to unseen ones

P(*w* | *denied the*)
3 *allegations*
2 *reports*
1 *claims*
1 *request*
----
7 total



P(*w* | *denied the*)
2.5 *allegations*
1.5 *reports*
0.5 *claims*
0.5 *request*
2.0 other
----
7 total



Laplace Smoothing

- Also known as add-one smoothing: add 1 to all counts
  - Unigrams:

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

  - Bigrams:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \longrightarrow P^*_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{\sum_w (C(w_{n-1}w) + 1)} = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

Laplace-smoothed Bigrams

- Bigram counts

$C(w_{n-1}w_n)$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

$C(w_{n-1}w_n)+1$

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

- Unigram counts and vocabulary size

$C(w_{n-1})$

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$V = 1446$

- Bigram probabilities

$$P^*_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Adjusted Counts

- For comparison with the original counts, reconstruct the count matrix

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

- Original counts:

	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Other Smoothing Strategies

- Add-1 smoothing is too blunt
  - But is still used in NLP models for text classification
  - Useful when the number of zeros is not too large
- Other smoothing techniques:
  - Add-k smoothing: $$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV}$$
    - Choice of *k* can be optimized on a devset
  - Backoff and interpolation
    - Estimate n-gram probabilities using (n-1)-gram probabilities

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1})\\ &+\lambda_2 P(w_n|w_{n-1})\\ &+\lambda_3 P(w_n)\end{aligned}$$

  - …

Web-scale N-grams

- Google N-grams

Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

- Efficiency
  - Words stored as 64-bit hash indexes, probabilities quantized using 4-8 bits
  - Efficient data structures
  - Bloom filters: approximate language models
- Pruning
  - Only store n-grams with counts above threshold
  - Use entropy to prune less-important n-grams

Smoothing in Web-scale N-grams

- Stupid backoff
  - Give up trying to make the language model a true probability distribution
  - No discounting; if an n-gram has 0 count, backoff to a lower order n-gram weighted by a fixed weight

$$S(w_i|w_{i-k+1:i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1:i})}{\text{count}(w_{i-k+1:i-1})} & \text{if count}(w_{i-k+1:i}) > 0 \\ \lambda S(w_i|w_{i-k+2:i-1}) & \text{otherwise} \end{cases}$$

  - Unigram probability:

$$S(w) = \frac{count(w)}{N}$$

Neural Language Models

- Based on word embeddings
  - Can generalize over contexts of similar words
  - Don’t need smoothing
- Can handle much longer histories
- Typically have much higher predictive accuracy than an n-gram language model

➢ All this comes at a cost: neural language models are much slower to train than traditional language models

The Python Notebook

jupyter

n-gram-language-models_.ipynb

- Unigram, bi-gram, tri-gram language models
- N-gram language models
- Text generation

NLTK
https://www.nltk.org/