UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Natural Language Processing

Language Models: N-gram

LESSON 20

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

What's (Probabilistic) Language Modeling

- Goal
  - compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- Related task: the probability of an upcoming word:

$$P(w_5|w_1,w_2,w_3,w_4)$$

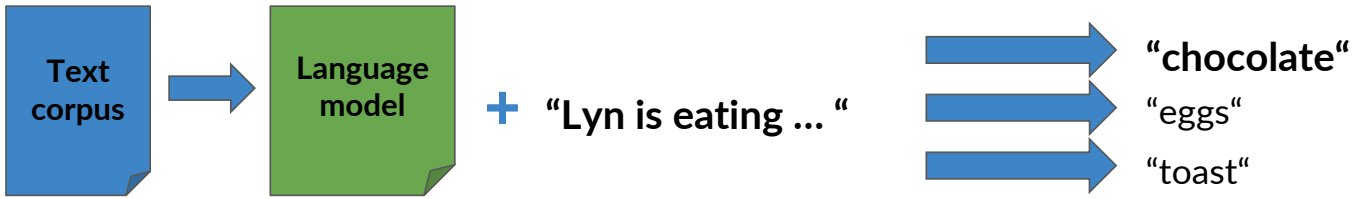
- A model that computes either of these:

$P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**

  - Also called the grammar, but language model or LM is standard

Overview

- Create language model (LM) from text corpus to
  - Estimate probability of word sequences
  - Estimate probability of a word following a sequence of words
- Apply this concept to autocomplete a sentence with most likely suggestions



Other Applications

- Speech recognition
  - P(I saw a van) > P(eyes awe of an)
- Spelling correction
  - "He entered the ship to buy some groceries" – "ship" a dictionary word
    - P(entered a shop to buy)>P(entered the ship to buy)
- Augmentative communication
  - Predict most likely word from menu for people unable to physically talk or sign

What we'll be learning

- Process text corpus to N-gram language model
- Handle out-of-vocabulary words
- Smoothing for previously unseen N-grams
- Language model evaluation

N-grams and Probabilities

- What are N-grams?
- N-grams and conditional probabilities from corpus

N-gram

- An N-gram is a sequence of N words
- Corpus: I am happy because I am learning
  - Note that when processing a corpus, the punctuation is treated like words

Unigrams: { I , am , happy , because , learning }

Bigrams: { I am , am happy , happy because ... }

I happy

Trigrams: { I am happy , am happy because, ... }

Sequence notation

Corpus: This is great ... teacher drinks tea.
$w_1 \; w_2 \; w_3$ $w_{498} \; w_{499} \; w_{500}$

$m = 500$

$$w_1^m = w_1 \; w_2 \; \ldots \; w_m$$

$$w_1^3 = w_1 \; w_2 \; w_3$$

$$w_{m-2}^m = w_{m-2} \; w_{m-1} \; w_m$$

Unigram probability

- Corpus: I am happy because I am learning
  - Corpus size m=7
  - P(*I*) = 2/7, P(*happy*) = 1/7
- Probability of unigram
  - P(w) = C(w)/m

Bigram probability

- Corpus: I am happy because I am learning

$$P(am|I) = \frac{C(I\ am)}{C(I)} = \frac{2}{2} = 1 \qquad P(happy|I) = \frac{C(I\ happy)}{C(I)} = \frac{0}{2} = 0$$ ✖ I happy

$$P(learning|am) = \frac{C(am\ learning)}{C(am)} = \frac{1}{2}$$

Probability of a bigram: $P(y|x) = \frac{C(x\ y)}{\sum_w C(x\ w)} = \frac{C(x\ y)}{C(x)}$

Trigram probability

- Corpus: I am happy because I am learning

$$P(happy|I\ am) = \frac{C(I\ am\ happy)}{C(I\ am)} = \frac{1}{2}$$

Probability of a trigram: $$P(w_3|w_1^2) = \frac{C(w_1^2\ w_3)}{C(w_1^2)}$$

$$C(w_1^2\ w_3) = C(w_1\ w_2\ w_3) = C(w_1^3)$$

N-gram probability

- Probability of N-gram:

$$P(w_N|w_1^{N-1}) = \frac{C(w_1^{N-1}\, w_N)}{C(w_1^{N-1})}$$

$$C(w_1^{N-1}\, w_N) = C(w_1^N)$$

Example

- Corpus: "*In every place of great resort the monster was the fashion. They sang of it in the cafes, ridiculed it in the papers, and represented it on the stage.* " (Jules Verne, *Twenty Thousand Leagues under the Sea*)
- In the context of our corpus, what is the probability of word "*papers*" following the phrase "*it in the*"?
  - P(*papers* | *it in the*) = 1/2
  - P(*papers* | *it in the*) = 2/3
  - P(*papers* | *it in the*) = 1
  - P(*papers* | *it in the*) = 0

Sequence probabilities

- Given a sentence, what is its probability?
  - *P(the teacher drinks tea) = ?*
- Remind that (conditional probability and chain rule)

$$P(B|A) = \frac{P(A,B)}{P(A)} \implies P(A,B) = P(A)P(B|A)$$

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

The Chain Rule for words

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

- Example
- P(*the teacher drinks tea*) =
  P(*the*)P(*teacher*|*the*)P(*drinks*|*the teacher*)P(*tea*|*the teacher drinks*)

Sentence not in corpus

- A corpus almost never contains the exact sentence we're interested in or even its longer subsequences!
- Example: The teacher drinks tea (input)

$$P(the\ teacher\ drinks\ tea) = P(the)P(teacher|the)P(drinks|the\ teacher)P(tea|the\ teacher\ drinks)$$

$$P(tea|the\ teacher\ drinks) = \frac{C(the\ teacher\ drinks\ tea)}{C(the\ teacher\ drinks)}$$

← Both likely 0

Approximation of sequence probability

- What if instead of looking for all the words before tea, one just considers the previous word (*drinks* in this case)

the teacher drinks tea

$$P(tea|the\ teacher\ drinks) \approx P(tea|drinks)$$

$$P(teacher|the)$$
$$P(drinks|teacher)$$
$$P(tea|drinks)$$

$$P(the\ teacher\ drinks\ tea) =$$

$$P(the)P(teacher|the)P(drinks|the\ teacher)P(tea|the\ teacher\ drinks)$$

$$P(the)P(teacher|the)P(drinks|teacher)P(tea|drinks)$$

Approximation of sequence probability

- Markov assumption: Only the last N words matter
  - Bigram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$
  - N-gram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$
  - Entire sequence modeled with bigram

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$$

$$P(w_1^n) \approx P(w_1)P(w_2|w_1)...P(w_n|w_{n-1})$$

unigram probability of the first word in the sentence

Example

- Given these conditional probabilities:
  - P(*Mary*)=0.1
  - P(*likes*)=0.2
  - P(*cats*)=0.3
  - P(*Mary* | *likes*) =0.2
  - P(*likes* | *Mary*) =0.3
  - P(*cats* | *likes*)=0.1
  - P(*likes* | *cats*)=0.4
- Approximate the probability of the following sentence with bigrams: "*Mary likes cats*"
  - P(Mary likes cats) = 0
  - P(Mary likes cats) =1
  - P(Mary likes cats) = 0.003
  - P(Mary likes cats) = 0.008

Starting of sentence token <s>

- How do we handle the beginning and the end of a sentence?
  - We don't have a context for the previous word

the teacher drinks tea

$$P(the\ teacher\ drinks\ tea) \approx P(the)P(teacher|the)P(drinks|teacher)P(tea|drinks)$$

<s> the teacher drinks tea

$$P(<s>\ the\ teacher\ drinks\ tea) \approx P(the|<s>)P(teacher|the)P(drinks|teacher)P(tea|drinks)$$

Start of sentence token <s> for N-grams

- In general, a similar principles applies to N-grams

Trigram:

$$P(the\ teacher\ drinks\ tea) \approx P(the)P(teacher|the)P(drinks|the\ teacher)P(tea|teacher\ drinks)$$

the teacher drinks tea => <s> <s> the teacher drinks tea

$$P(w_1^n) \approx P(w_1|<s>\ <s>)P(w_2|<s>\ w_1)...P(w_n|w_{n-2}\ w_{n-1})$$

N-gram model: add N-1 start tokens <s>

End of sentence token </s> - motivation

- What about the end of sentences?
- Recall that $P(y|x) = \frac{C(x\ y)}{\sum_w C(x\ w)} = \frac{C(x\ y)}{C(x)}$

Corpus:
<s> Lyn drinks chocolate
<s> John drinks

$$\sum_w C(drinks\ w) = 1$$

$$C(drinks) = 2$$

End of sentence token </s> - motivation

- There's another issue with your N-gram probabilities

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 2:

<s> yes yes

<s> yes no

<s> no no

<s> no yes

$$P(< s > \text{ yes yes}) =$$

$$P(\text{yes} \mid < s >) \times P(\text{yes} \mid \text{yes}) =$$

$$\frac{C(< s > \text{ yes})}{\sum_w C(< s > w)} \times \frac{C(\text{yes yes})}{\sum_w C(\text{yes } w)} =$$

$$\frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$$

End of sentence token </s> - motivation

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 2:

<s> yes yes

<s> yes no

<s> no no

<s> no yes

$$P(< s > \text{ yes yes}) = \frac{1}{3}$$

$$P(< s > \text{ yes no}) = \frac{1}{3}$$

$$P(< s > \text{ no no}) = \frac{1}{3}$$

$$P(< s > \text{ no yes}) = 0$$

$$\sum_{2 \text{ word}} P(\cdots) = 1$$

End of sentence token </s> - motivation

- Now, look at all the possible three-words sentences

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 3:

<s> yes yes yes

<s> yes yes no

...

<s> no no no

$$P(< s > \text{ yes yes yes}) = \dots$$

$$P(< s > \text{ yes yes no}) = \dots$$

$$\dots = \dots$$

$$P(< s > \text{ no no no}) = \dots$$

$$\sum_{3 \text{ word}} P(\cdots) = 1$$

End of sentence token </s> - motivation

- However, you really want the sum of the probabilities for all sentences of any length to be equal to 1
  - To compare the probabilities of two sentences of different lengths
- That is, the probabilities of all 2-word sentences plus the probabilities of all 3-word sentences, plus the probabilities of all other sentences of arbitrary lengths, should be equal to 1

Corpus

<s> yes no

<s> yes yes

<s> no no

$$\sum_{\text{2 word}} P(\cdots) + \sum_{\text{3 word}} P(\cdots) + \ldots = 1$$

End of sentence token </s> - solution

- There's a simple solution
  - Add a special symbol, </s>, for the end of a sentence
- Example
  - Bigrams

<s> the teacher drinks tea => <s> the teacher drinks tea </s>

$$P(the|<s>)P(teacher|the)P(drinks|teacher)P(tea|drinks)P(</s>|tea)$$

Corpus:
<s> Lyn drinks chocolate </s>
<s> John drinks </s>

$$\sum_w C(drinks\ w) = 2$$
$$C(drinks) = 2$$

End of sentence token </s> for N-grams

- N-gram => just one </s>
- Example
  - Trigram
    - The teacher drinks tea => <s> <s> the teacher drinks tea </s>

Example: Bigram

- Bigram probabilities

Corpus

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

$$P(John|<s>) = \frac{1}{3} \qquad P(</s>|tea) = \frac{1}{1}$$

$$P(chocolate|eats) = \frac{1}{2} \qquad P(Lyn|<s>) = ? = \frac{2}{3}$$

$$P(sentence) = \frac{2}{3} * \frac{1}{2} * \frac{1}{2} * \frac{2}{2} = \frac{1}{6}$$

- Note that the result is 1/6, which is lower than the value of 1/3 you might expect when calculating the probability of one of the three sentences in the training corpus
  - This also applies to the other two sentences in the corpus

Example

- Given these conditional probabilities:
  - P(*Mary*)=0.1
  - P(*likes*)=0.2
  - P(*cats*)=0.3
  - P(*Mary* | <s>)=0.2
  - P(</s> | *cats*)=0.6
  - P(*likes* | *Mary*) =0.3
  - P(*cats* | *likes*)=0.1
- Approximate the probability of the following sentence with bigrams: "*<s> Mary likes cats </s>*"
  - P(<s> *Mary likes cats* </s>) =0.0036
  - P(<s> *Mary likes cats* </s>) = 1
  - P(<s> *Mary likes cats* </s>) = 0.003
  - P(<s> *Mary likes cats* </s>) = 0

The N-gram Language Model

- Count matrix
- Probability matrix
- Language model
- Log probability to avoid underflow
- Generative language model

Count matrix

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

- Count matrix
  - Captures the numerator for all N-grams appearing in the corpus

- Bigram count matrix
  - The corpus *I study I learn*, the rows represent the first word of the bigram and the columns represent the second word of the bigram

Corpus: <s>I study I learn</s>

"study I" bigram →

	<s>	</s>	I	study	learn
<s>	0	0	1	0	0
</s>	0	0	0	0	0
I	0	0	0	1	1
study	0	0	1	0	0
learn	0	1	0	0	0

Probability matrix

- Divide each cell by its row sum

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

- Corpus: <s>I study I learn</s>

$$sum(row) = \sum_{w \in V} C(w_{n-N+1}^{n-1}, w) = C(w_{n-N+1}^{n-1})$$

Count matrix (bigram)

	<s>	</s>	I	study	learn	sum
<s>	0	0	1	0	0	1
</s>	0	0	0	0	0	0
I	0	0	0	1	1	2
study	0	0	1	0	0	1
learn	0	1	0	0	0	1

Probability matrix

	<s>	</s>	I	study	learn
<s>	0	0	1	0	0
</s>	0	0	0	0	0
I	0	0	0	0.5	0.5
study	0	0	1	0	0
learn	0	1	0	0	0

Language model

- Probability matrix => language model
  - Sentence probability
  - Next word prediction

	<s>	</s>	I	study	learn
<s>	0	0	1	0	0
</s>	0	0	0	0	0
I	0	0	0	0.5	0.5
study	0	0	1	0	0
learn	0	1	0	0	0

Sentence probability:

<s> I learn </s>

$$P(sentence) =$$
$$P(I|<s>)P(learn|I)P(</s>|learn) =$$
$$1 \times 0.5 \times 1 =$$
$$0.5$$

Log probability

- All probabilities <=1 and multiplying them brings the risk of underflow

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$$

- Use log of the probabilities in probability matrix and calculations

$$log(P(w_1^n)) \approx \sum_{i=1}^{n} log(P(w_i|w_{i-1}))$$

- To convert back from log

$$P(w_1^n) = exp(log(P(w_1^n))$$

Generative language model

- One interesting application of language models is text generation from scratch or using a small hint

Corpus:
<s> Lyn drinks chocolate </s>
<s> John drinks tea </s>
<s> Lyn eats chocolate </s>

1. (<s>, Lyn) or (<s>, John)?
2. (Lyn,eats) or (Lyn,drinks) ?
3. (drinks,tea) or (drinks,chocolate)?
4. (tea,</s>) - always

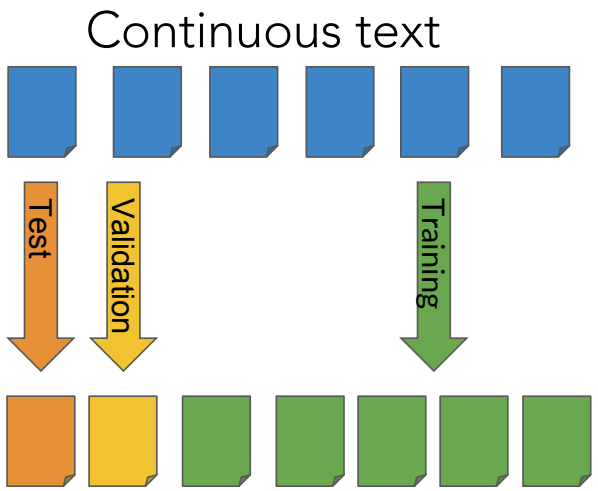
- Algorithm
  1. It chooses among all bigrams, starting with <s>, based on the bigram probability
     - That means the bigrams with higher values in the probability matrix are more likely to be chosen
  2. It chooses a new bigram at random from the bigrams beginning with the previously chosen word
  3. Then, this bigram is added to the sentence
  4. The algorithm continues like this until the end sentence token, </s> is chosen
     - This is done by randomly choosing a bigram that starts with the previous word and ends with </s>

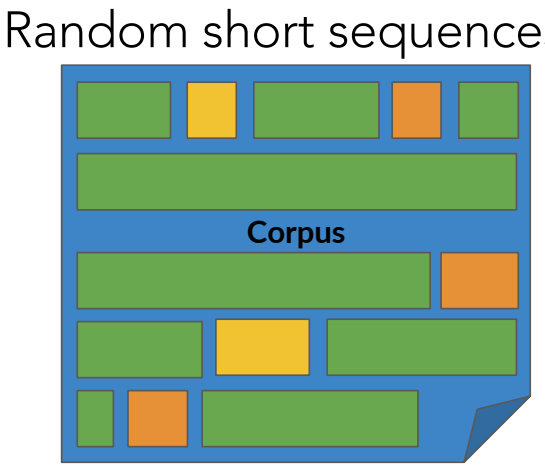
Language model evaluation

- Train/Validation/Test split
  - Small corpora
    - 80% Train
    - 10% Validation
    - 10% Test
  - Large corpora
    - 98% Train
    - 1% Validation
    - 1% Test
- Perplexity

Test data: Split method

- In NLP there are two main methods for splitting
  - split the corpus by choosing longer continuous segments like Wikipedia articles
  - randomly choose short sequences of words such as those in the sentences





A measure of quality of LMs

- Given a sentence, $W=w_1,w_2,\ldots,w_m$, its probability is $P(W)=P(w_1,w_2,\ldots,w_m)$
  - The higher the probability of W, the more accurate (or even realistic) the sentence W is
- Given a language model, we do not use directly the probability of a test sentence to evaluate the LM effectiveness, rather we use a measure called perplexity, defined as

$$PP(W) = P(w_1,w_2,\ldots,w_m)^{-1/m} = \sqrt[m]{\frac{1}{P(w_1,w_2,\ldots,w_m)}} = \sqrt[N]{\prod_{i=1}^{m}\frac{1}{P(w_i \mid w_1,w_2,\ldots,w_{i-1})}}$$

- If we use a bigram LM, then the perplexity become

$$\sqrt[m]{\prod_{i=1}^{m}\frac{1}{P(w_i \mid w_{i-1})}}$$

Perplexity

- The LM can be evaluated on test sets using the perplexity metric

$$PP(W) = P(s_1, s_2, \ldots, s_N)^{-1/m}$$

- W-> test set containing N sentences s_j, j=1,…,N
- s_j -> j-th sentence in the test set, each ending with </s>
- m -> number of all words in the entire test set W including </s> but not including <s>
  - If the length of $s_j=n_j$, j=1,…, N then m = $\sum_{j=1}^{N} n_j$

Perplexity: Example

- m=100

$$P(W) = 0.9 => PP(W) = 0.9^{-\frac{1}{100}} = 1.00105416$$
$$P(W) = 10^{-250} => PP(W) = (10^{-250})^{-\frac{1}{100}} \approx 316$$

- Smaller perplexity = better model
- Character level models PP < word-based models PP

Perplexity for bigrams models

- Concatenate all sentences in W

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

$w_i \rightarrow$ i-th word in test set

Log perplexity

- For ease of computation, the log perplexity is often used

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

$$logPP(W) = -\frac{1}{m}\sum_{i=1}^{m} log_2(P(w_i|w_{i-1}))$$

Example: Wall Street Journal Corpus

- Training 38 million words, test 1.5 million words
  - Perplexity Unigram: 962 Bigram: 170 Trigram: 109

Unigram
Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
Bigram
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
Trigram
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

Example: Wall Street Journal Corpus

- To better fix the perplexity changes in unigram, bigram, and trigram LM
  - Let's recall that the N of an N-gram language model defines the context used in each conditional probability
  - Consider the following sentences
    - *Gorillas always like to groom their friends*
    - *The computer that's on the 3rd floor of our office building crashed*
  - The red words depend on each other
  - If N is too small
    - the context cannot be captured, and those sentences get a very low probability, while sentences that fail basic linguistic rules get higher probabilities
  - If N is too big, however, it is hard getting good estimates of the parameters from our corpus, because of data sparsity

What about (possibly) missing training words?

- When using LM in real tasks, it happens to deal with words that did not occur in the training set
  - Need to handle words missing from a training corpus
    - Unknown words
- We need to
  - Update corpus with <UNK>
  - Choose a vocabulary

Out of vocabulary words

- Closed vs Open vocabulary
  - In some tasks, one will encounter and generate only words from a fixed set of words
    - The fixed set of words is called closed vocabulary
    - The test set can only contain words from that lexicon (no unknown words)
  - In other cases, we deal with words we haven't seen before
    - An open vocabulary contains potentially unknown words
- Unknown word = Out of vocabulary word (OOV)
  - Special tag <UNK> in corpus and in input

Using <UNK> in Corpus

- Two ways to train the LM for unknown words
  - Turn the problem back into a closed vocabulary one by choosing a fixed vocabulary in advance
    - Choose a vocabulary (word list) that is fixed in advance
    - Convert in the training set any OOV to the token <UNK>
    - Estimate the probability for <UNK> from its count like any other regular word in the training set
  - Create a vocabulary implicitly (we don't have a prior vocabulary in advance)
    - E.g., we can replace by <UNK> the words based on their frequency

Example

- Training corpus where (we decided) a vocabulary word appears at least twice (min frequency = 2)
  - Replace all words with freq < 2 with <UNK>

Corpus
<s> Lyn drinks chocolate </s>
<s> John drinks tea </s>
<s> Lyn eats chocolate </s>

Corpus
<s> Lyn drinks chocolate </s>
<s> <UNK> drinks <UNK> </s>
<s> Lyn <UNK> chocolate </s>

Min frequency f=2

Vocabulary
Lyn, drinks, chocolate

Input query
<s>Adam drinks chocolate</s>

<s><UNK> drinks chocolate</s>

- Any words in the query outside the vocabulary also replaced with <UNK>

How to create vocabulary V

- Criteria
  - Min word frequency f
  - Max |V|, include words by frequency
- Use <UNK> sparingly
  - If we have a lot of <UNK>, the model will generate a sequence of >UNK> with a high probability
- Perplexity
  - Only compare LMs with the same V

Example

- Given the training corpus and minimum word frequency=2, how would the vocabulary for corpus preprocessed with <UNK> look like?
- "<s> *I am happy I am learning* </s> <s> *I am happy I can study* </s>"
  - V = (*I*,*am*,*happy*,*I*,*am*)
  - V = (*I*,*am*,*happy*,*learning*,*can*,*study*,*<UNK>*)
  - V = (*I*,*am*,*happy*)
  - V = (*I*,*am*,*happy*,*learning*,*can*,*study*)

Smoothing

- It may happen that words are in a vocabulary but appear in a test set in an unseen context
  - E.g., they appear after a word they never appeared after, in training
  - The LM would assign zero probs to these unseen events
- Missing N-grams in the corpus
  - Smoothing
  - Backoff and interpolation

Missing N-grams in training corpus

- Problem: N-grams made of known words still might be missing in the training corpus
  - "John", "eats" in corpus, and "John eats" not in the corpus
- Their counts cannot be used for probability estimation

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

Can be 0

Smoothing

- Add-one smoothing (Laplacian smoothing)
  - Only work when real counts are large enough to outweigh the +1
    - The probability of missing words would be too high

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n) + 1}{\sum_{w \in V}(C(w_{n-1}, w) + 1)} = \frac{C(w_{n-1}, w_n) + 1}{C(w_{n-1}) + V}$$

- Add-k smoothing (for larger corpora)
  - k < 1, e.g., 0.5, 0.05, 0.01
    - Makes the probabilities smoother

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n) + k}{\sum_{w \in V}(C(w_{n-1}, w) + k)} = \frac{C(w_{n-1}, w_n) + k}{C(w_{n-1}) + k * V}$$

Smoothing as a discounting

- Smoothing can be viewed as discounting (lowering) some non-zero counts in order to get the probability mass that will be assigned to zero counts
- However, there are alternative approaches
  - If we're trying to compute $P(w_n \mid w_{n-2}, w_{n-1})$ with no examples of a particular trigram $w_{n-2}w_{n-1}w_n$, we can estimate its probability by using the bigram probability $P(w_n \mid w_{n-1})$
    - Similarly, if $P(w_n \mid w_{n-1}) = 0$ we can look to $P(w_n)$

Backoff

- With backoff if N-gram information is missing, you use N-1 gram
  - If that's also missing, you would use N-2 gram and so on until you find a non-zero probability
- It distorts the probability distribution, especially for smaller corpora
  - Some probability needs to be discounted from higher-level N-grams to use it for lower-level N-grams
    - The Katz backoff method uses discounting
  - "Stupid" backoff (large corpora)
    - The lower order N-gram probability is multiplied by a constant (e.g., 0.4)

"Stupid" backoff

Corpus

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

$$P(chocolate|John\ drinks) = ?$$

$$0.4 \times P(chocolate|drinks)$$

Interpolation

- An alternative approach to backoff is to use the linear interpolation of all orders of N-grams
  - That is, you would always combine the weighted probability of the N-gram, N-1 gram down to 1-grams
- Example
  - Trigrams

$$\hat{P}(chocolate|John\ drinks) = 0.7 \times P(chocolate|John\ drinks) + 0.2 \times P(chocolate|drinks) + 0.1 \times P(chocolate)$$

$$\hat{P}(w_n|w_{n-2}\ w_{n-1}) = \lambda_1 \times P(w_n|w_{n-2}\ w_{n-1}) + \lambda_2 \times P(w_n|w_{n-1}) + \lambda_3 \times P(w_n) \qquad \sum_i \lambda_i = 1$$

  - The lambdas are learned from the validation part of the corpus