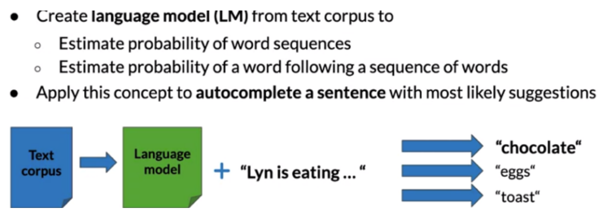
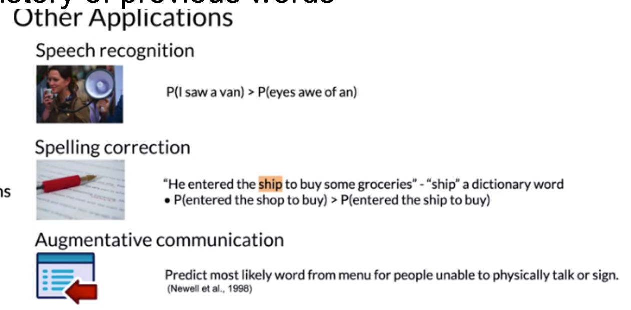
Autocomplete and Language Models

N_Grams: Overview

A language model is a tool that's calculates the probabilities of sentences. You can think of sentence as a sequence of words. Language models can also estimate the probability of an upcoming word given a history of previous words



- 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)

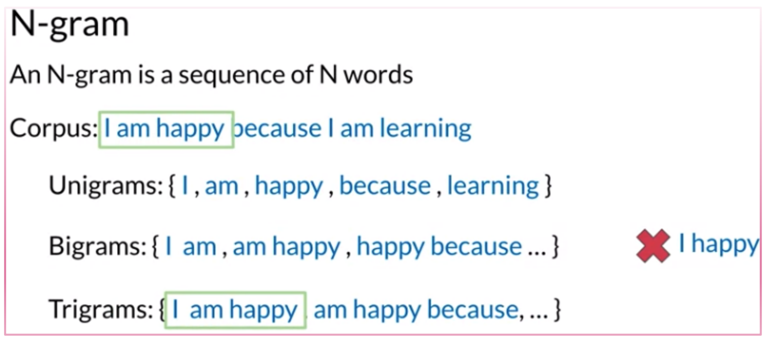
Spelling correction

"He entered the ship to buy some groceries" - "ship" a dictionary word
- P(entered the 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. (Newell et al., 1998)

N_grams and Probabilities



N-gram

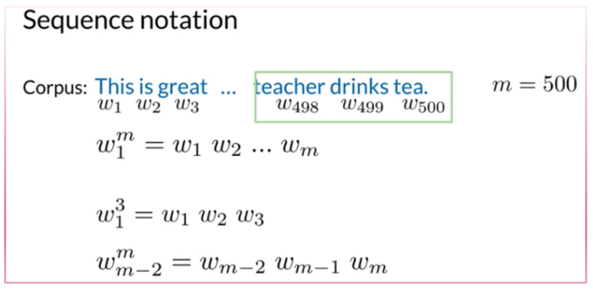
An N-gram is a sequence of N words

Corpus: I am happy because I am learning

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

Bigrams: { I am , am happy , happy because … } ✖ I happy

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



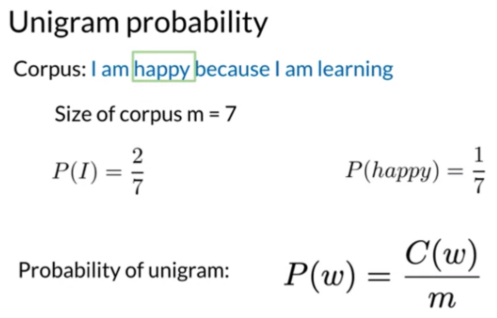
Sequence notation

Corpus: $\underset{w_1}{\text{This}}\ \underset{w_2}{\text{is}}\ \underset{w_3}{\text{great}}\ \ldots\ \underset{w_{498}}{\text{teacher}}\ \underset{w_{499}}{\text{drinks}}\ \underset{w_{500}}{\text{tea.}}$ $m = 500$

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

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

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



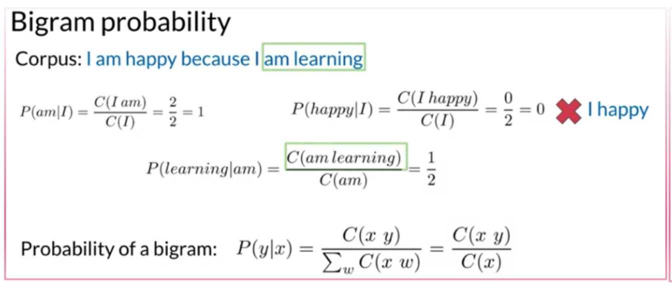
Unigram probability

Corpus: I am happy because I am learning

Size of corpus m = 7

$$P(I) = \frac{2}{7} \qquad P(happy) = \frac{1}{7}$$

Probability of unigram: $$P(w) = \frac{C(w)}{m}$$



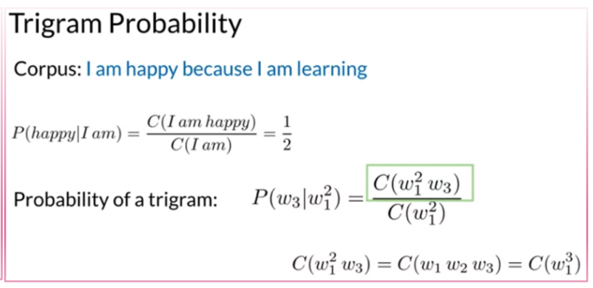
Bigram probability

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

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

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: $$P(w_3|w_1^2) = \frac{\boxed{C(w_1^2\ w_3)}}{C(w_1^2)}$$

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

N-gram probability

Probability of N-gram: $P(w_N|w_1^{N-1}) = \frac{\boxed{C(w_1^{N-1}w_N)}}{C(w_1^{N-1})}$

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

Sequence Probabilities

- Conditional probability and chain rule reminder

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

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

Sentence not in corpus

- Problem: Corpus almost never contains the exact sentence we're interested in or even its longer subsequences!

Input: the teacher drinks tea

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

$$\boxed{P(tea|the\ teacher\ drinks)} = \frac{C(the\ teacher\ drinks\ tea)}{C(the\ teacher\ drinks)} \begin{array}{l}\leftarrow \text{Both}\\ \leftarrow \text{likely 0}\end{array}$$

Approximation of sequence probability

the teacher drinks tea

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

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

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

$$P(the)P(teacher|the)P(drinks|the\ teacher)\boxed{P(tea|the\ teacher\ drinks)}$$

$$\Downarrow$$

$$P(the)P(teacher|the)P(drinks|teacher)\boxed{P(tea|drinks)}$$

We approximate because of **Markov's assumption – only last N matters matter**

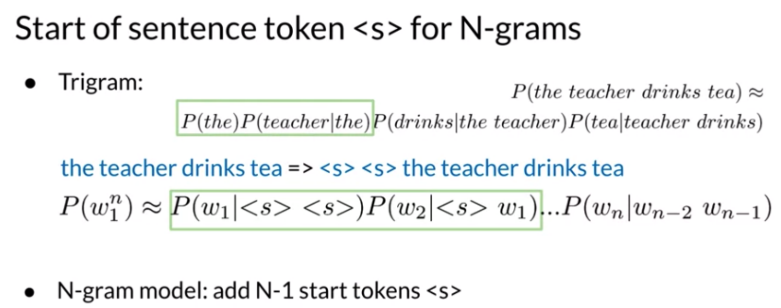
- Bigram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$
- N-gram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$
- Entire sentence modeled with bigram $P(w_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$

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

Starting and Ending Sentences

Start of sentence symbols <s>

End of sentence symbol </s>



Start of sentence token <s> for N-grams

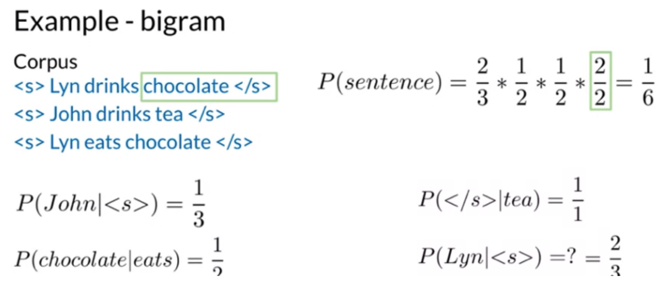
- Trigram:

$$P(the\ teacher\ drinks\ tea) \approx \boxed{P(the)P(teacher|the)}P(drinks|the\ teacher)P(tea|teacher\ drinks)$$

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

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

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



Example - bigram

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

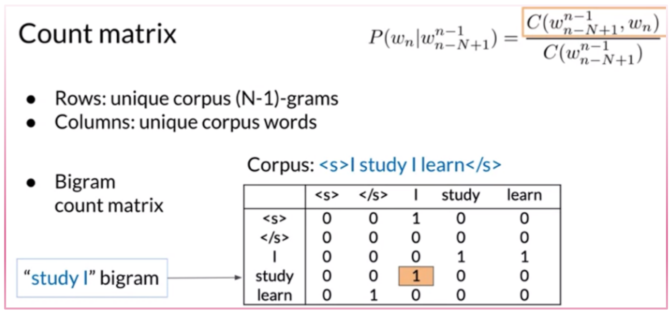
$$P(sentence) = \frac{2}{3} * \frac{1}{2} * \frac{1}{2} * \boxed{\frac{2}{2}} = \frac{1}{6}$$

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

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

The N-gram Language Model

- Count matrix



Count matrix

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

- Rows: unique corpus (N-1)-grams
- Columns: unique corpus words

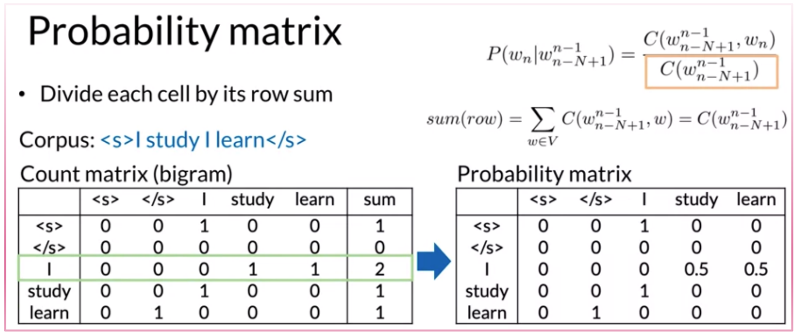
- Bigram count matrix

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

	<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

"study I" bigram

- Probability matrix
  - Divide each cell by row sum



Probability matrix

- Divide each cell by its row sum

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

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

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

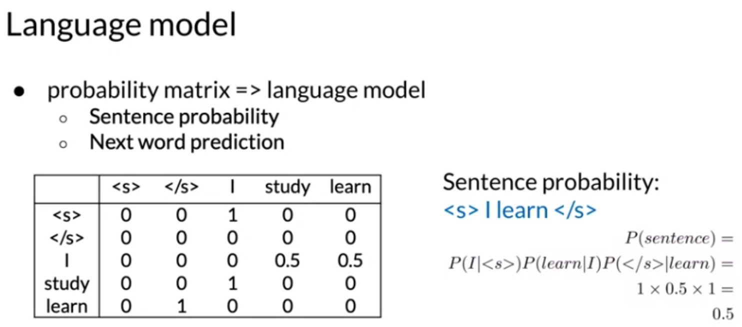
Count matrix (bigram)

	<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



Language model

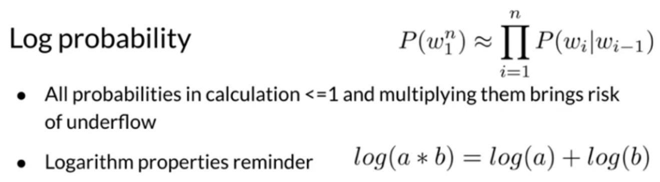
- probability matrix => language model
  - Sentence probability
  - Next word prediction

	<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>

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

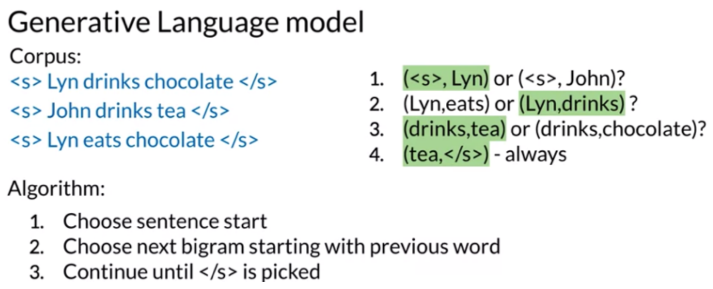
- Log probability to avoid overflow



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

- All probabilities in calculation <=1 and multiplying them brings risk of underflow
- Logarithm properties reminder $log(a * b) = log(a) + log(b)$

- Generative Language model



Generative Language model

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

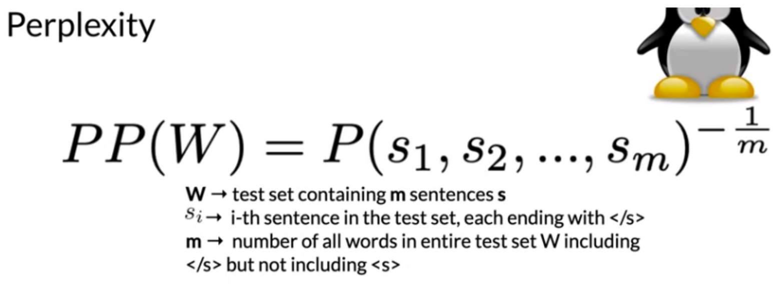
1. (<s>, Lyn) or (<s>, John)?
2. (Lyn,eats) or (Lyn,drinks) ?
3. (drinks,tea) or (drinks,chocolate)?
4. (tea,</s>) - always

Algorithm:
1. Choose sentence start
2. Choose next bigram starting with previous word
3. Continue until </s> is picked

Language Model Evaluation

Split corpus to train/validation/test

Split either as continuous text or random short sequences



Perplexity

$$PP(W) = P(s_1, s_2, ..., s_m)^{-\frac{1}{m}}$$

W → test set containing **m** sentences **s**
s_i → i-th sentence in the test set, each ending with </s>
m → number of all words in entire test set W including </s> but not including <s>

Perplexity is basically the inverse probability of the test sets normalized by the number of words in the test set. So the higher the language model estimates the probability of your test set the lower the perplexity is going to be. perplexity is closely related to entropy which measures uncertainty.

Smaller perplexity better model
Character level models PP < word based models PP
Good language models have perplexity between 60 and 20

Perplexity for bigram models

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

$w_j^{(i)} \rightarrow$ j-th word in i-th sentence

- 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

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

$$\Downarrow$$

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

Out of Vocabulary

- Unknown words
- Closed vocabulary – e.g. chatbot only answering from a fixed list of questions
- Open vocabulary – outside of vocabulary (OOV) which will have words unknown. Use special tag <UNK> in corpus and in input

Using <UNK> in corpus

- Create vocabulary V
- Replace any word in corpus and not in V by <UNK>
- Count the probabilities with <UNK> as with any other word

Example

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>

How to create vocabulary V

- Criteria:
  - Min word frequency f
  - Max |V|, include words by frequency
- Use <UNK> sparingly
- Perplexity - only compare LMs with the same V

Smoothing

Smoothing

- Add-one smoothing (Laplacian smoothing)

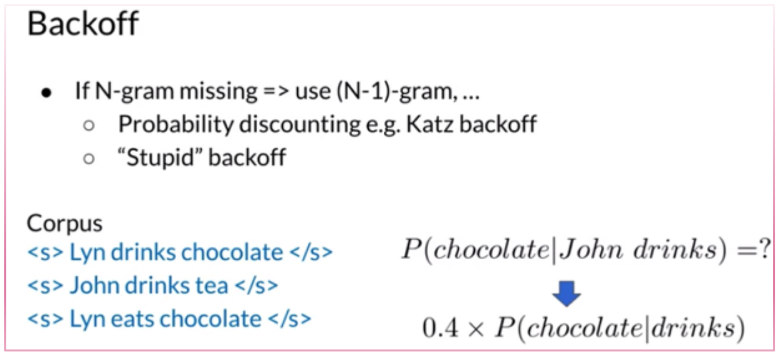
$$P(w_n|w_{n-1}) = \frac{C(w_{n-1},w_n)+1}{\sum_{\boxed{w\in V}}(C(w_{n-1},w)+\boxed{1})} = \frac{C(w_{n-1},w_n)+1}{C(w_{n-1})+\boxed{V}}$$

- Add-k smoothing

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

Advanced smoothing

- Kneser-Ney
- Good-Turing



Backoff

- If N-gram missing => use (N-1)-gram, ...
  - Probability discounting e.g. Katz backoff
  - “Stupid” backoff

Corpus

<s> Lyn drinks chocolate </s>

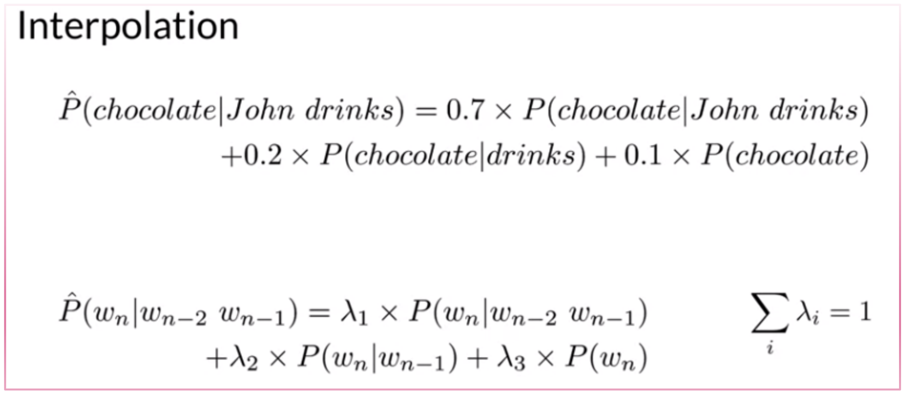
<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

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

$$\Downarrow$$

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



Interpolation

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

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

Lambdas need to add up to one. The Lambdas are learned from the validation parts of the corpus. You can get them by maximizing the probability of sentences from the validation set. Use a fixed language model trained from the training parts of the corpus to calculate n-gram probabilities and optimize the Lambdas. The interpolation can be applied to general n-gram by using more Lambdas.

Here are the steps of this assignment:

1. Load and preprocess data
   - Load and tokenize data.
   - Split the sentences into train and test sets.
   - Replace words with a low frequency by an unknown marker `<unk>`.
2. Develop N-gram based language models
   - Compute the count of n-grams from a given data set.
   - Estimate the conditional probability of a next word with k-smoothing.
3. Evaluate the N-gram models by computing the perplexity score.
4. Use your own model to suggest an upcoming word given your sentence.