Introduction to Probabilistic Language Model

Ramaseshan Ramachandran

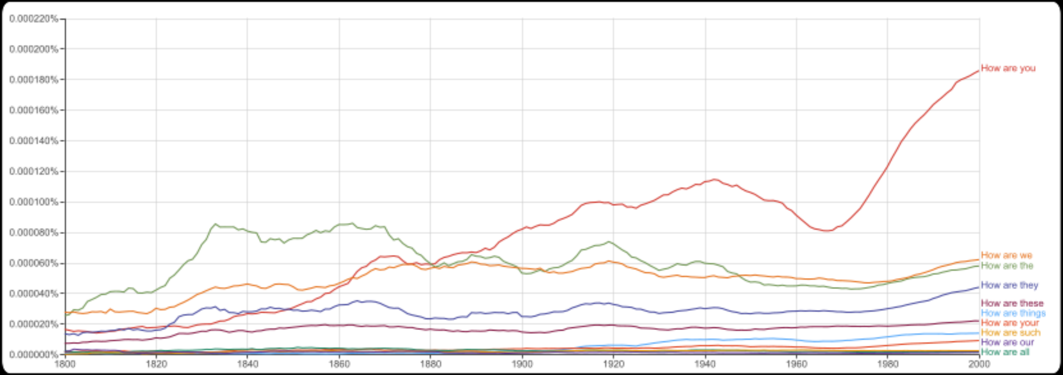
[illegible]

INTRODUCTION

How are _____? Can you guess the missing word?

INTRODUCTION

How are _____? Can you guess the missing word?



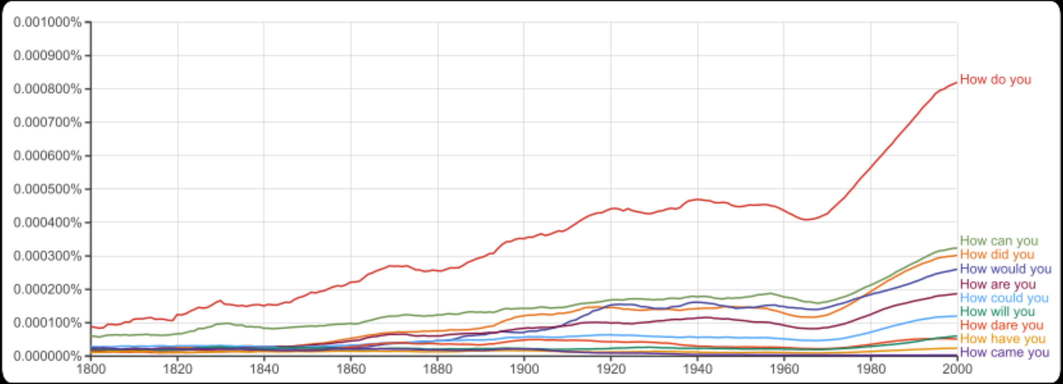
Source:Google NGram Viewer

INTRODUCTION

How _____ you? Can you guess the missing word?

INTRODUCTION

How _____ you? Can you guess the missing word?



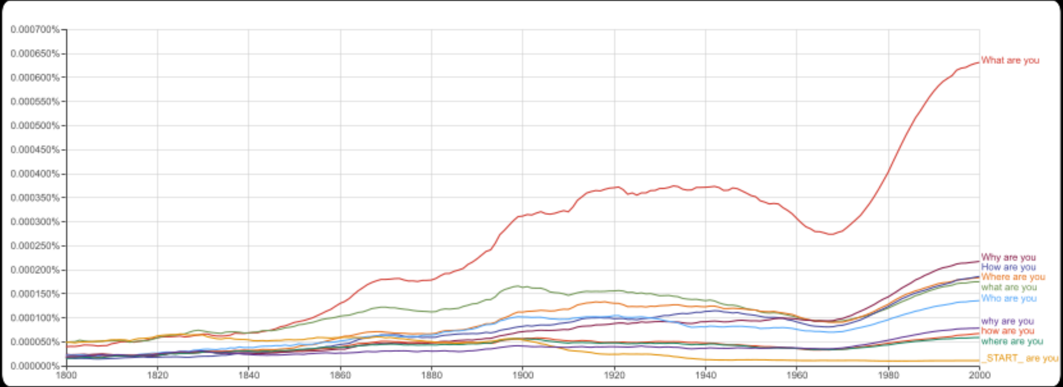
Source:Google NGram Viewer

INTRODUCTION

_____ are you?

INTRODUCTION

_____ are you?



Source:Google NGram Viewer

INTRODUCTION

How do humans predict the next word?

INTRODUCTION

How do humans predict the next word?

- Domain knowledge

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge
- Lexical knowledge
- Knowledge about the sentence structure
-
- Some words are hard to find. Why?

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge
- Lexical knowledge
- Knowledge about the sentence structure
-
- Some words are hard to find. Why?
- Natural language is not deterministic in general

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge
- Lexical knowledge
- Knowledge about the sentence structure
-
- Some words are hard to find. Why?
- Natural language is not deterministic in general
- Some sentences are familiar or had been heard/seen/used several times

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge
- Lexical knowledge
- Knowledge about the sentence structure
-
- Some words are hard to find. Why?
- Natural language is not deterministic in general
- Some sentences are familiar or had been heard/seen/used several times
- They are more likely to happen than others, hence we could guess

INTRODUCTION

How do humans predict the next word?

- Domain knowledge
- Syntactic knowledge
- Lexical knowledge
- Knowledge about the sentence structure
-
- Some words are hard to find. Why?
- Natural language is not deterministic in general
- Some sentences are familiar or had been heard/seen/used several times
- They are more likely to happen than others, hence we could guess

WHY PROBABILITY?

WHY PROBABILITY?

- Provides methods to predict or make decisions to pick the next word in the sequence based on sampled data

WHY PROBABILITY?

- Provides methods to predict or make decisions to pick the next word in the sequence based on sampled data
- Make the informed decision when there a certain degree of uncertainty and some observed data

WHY PROBABILITY?

- Provides methods to predict or make decisions to pick the next word in the sequence based on sampled data
- Make the informed decision when there a certain degree of uncertainty and some observed data
- It provides a quantitative description of the chances or likelihoods associated with various outcomes

WHY PROBABILITY?

- Provides methods to predict or make decisions to pick the next word in the sequence based on sampled data
- Make the informed decision when there a certain degree of uncertainty and some observed data
- It provides a quantitative description of the chances or likelihoods associated with various outcomes
- Probability of a sentence
- Probability of the next word in a sentence - how likely to predict "**you**" as the next word

WHY PROBABILITY?

- Provides methods to predict or make decisions to pick the next word in the sequence based on sampled data
- Make the informed decision when there a certain degree of uncertainty and some observed data
- It provides a quantitative description of the chances or likelihoods associated with various outcomes
- Probability of a sentence
- Probability of the next word in a sentence - how likely to predict "**you**" as the next word
- Likelihood of the next word is formalized through an observation by conducting experiment - counting the words in a document

PROBABILITY - DEFINITION AND PROPERTY

- The Probability is defined as the likelihood that an event will occur
- Let us use the most popular example - a flip of a coin - there is a 50% chance or probability that heads will come up for any given toss of a fair coin
- Probabilities can be expressed as percentage (60%), in decimal form (0.6) or in fractions (6/10)

DISCRETE SAMPLE SPACE

Consider following bag of words ($count = 52$)

Experiment - Extracting tokens from a document
Outcome - Every token/word x in the document
'a', 'weather', 'balloon', 'is', 'floating', 'at', 'a', 'constant', 'height', 'above', 'earth', 'when', 'it', 'releases', 'a', 'pack', 'of', 'instruments', 'level', 'a', 'if', 'the', 'pack', 'hits', 'the', 'ground', 'with', 'a', 'downward', 'velocity', 'of', 'm', 's', 'how', 'far', 'did', 'the', 'pack', 'fall', 'b', 'calculate', 'the', 'distance', 'the', 'ball', 'has', 'rolled', 'at', 'the', 'end', 'of', 's'

The outcome of the experiment - 52 samples (words). They constitute the ***sample space***,Ω or the set of all possible ***outcomes***
Each word in this sample belongs to Ω, represented by $x \in \Omega$
Each sample $x \in \Omega$ is assigned a probability score $[0, 1]$
A *probability function or probability distribution function* distributes the probability mass of 1 to the all the samples in the sample space Ω

SAMPLE SPACE - CONSTRAINTS

All the words in the Ω, must satisfy the following constraints:

1. $P(x) \in [0, 1], \forall x \in \Omega$ and
2. $\sum_{x \in \Omega} P(x) = 1$

EXAMPLE - 1

Bag of words $Count = 52$

'a', 'weather', 'balloon', 'is', 'floating', 'at', 'a', 'constant', 'height', 'above', 'earth', 'when', 'it', 'releases', 'a', 'pack', 'of', 'instruments', 'level', 'a', 'if', 'the', 'pack', 'hits', 'the', 'ground', 'with', 'a', 'downward', 'velocity', 'of', 'm', 's', 'how', 'far', 'did', 'the', 'pack', 'fall', 'b', 'calculate', 'the', 'distance', 'the', 'ball', 'has', 'rolled', 'at', 'the', 'end', 'of', 's'

If we are equally likely to pick any word from the BOW, then the probability for any word is
$P(x) = 1/52, \forall x \in \Omega$ so that
$P(\Omega) = 1$
$P('weather') = 1/52 = 0.01923076923$

EVENTS

'a', 'weather', 'balloon', 'is', 'floating', 'at', 'a', 'constant', 'height', 'above', 'earth', 'when', 'it', 'releases', 'a', 'pack', 'of', 'instruments', 'level', 'a', 'if', 'the', 'pack', 'hits', 'the', 'ground', 'with', 'a', 'downward', 'velocity', 'of', 'm', 's', 'how', 'far', 'did', 'the', 'pack', 'fall', 'b', 'calculate', 'the', 'distance', 'the', 'ball', 'has', 'rolled', 'at', 'the', 'end', 'of', 's'

Total number of words = 52. The number of unique words = 37 or there are 37 ***types*** of words in this BOW. 15 words have frequencies > 1.

An ***event*** is a collection of samples of the same type, $E \subseteq \Omega$

$$P(E) = \sum_{x \in E} P(x) \tag{1}$$

Events can be described as a variable taking a certain value

EVENTS - EXAMPLE

'a', 'weather', 'balloon', 'is', 'floating', 'at', 'a', 'constant', 'height', 'above', 'earth', 'when', 'it', 'releases', 'a', '**pack**', 'of', 'instruments', 'level', 'a', 'if', '**the**', '**pack**', 'hits', '**the**', 'ground', 'with', 'a', 'downward', 'velocity', 'of', 'm', 's', 'how', 'far', 'did', '**the**', '**pack**', 'fall', 'b', 'calculate', '**the**', 'distance', '**the**', 'ball', 'has', 'rolled', 'at', '**the**', 'end', 'of', 's'

In the BOW, the word type **the** occurs 6 times. Then

$$E_{the} = 6$$

$$P(E_{the}) = 6 \times \frac{1}{52} = 0.115$$

In the BOW, the word type **pack** occurs 3 times. Then

$$E_{pack} = 3$$

$$P(E_{pack}) = 3 \times \frac{1}{52} = 0.058$$

RANDOM VARIABLE

- A **random variable**,[1] is a variable whose possible values are numerical outcomes of a random phenomenon
- Two types - continuous and discrete - for NLP, they are discrete

To capture the type-token distinction, we use random variable W. $W(x)$ maps to the sample $x \in \Omega$.

V is the set of types and the value is represented by a variable v.

Given a random variable V and a value v, $P(V = v)$ is the probability of the event that V takes the value v, i.e.: $P(V = v) = P(x \in \Omega : V(x) = v)$

$P(V =' the') = P('the') = 0.115$

Random variables are useful in describing/constructing various events

[1] Random Variable - http://www.stats.gla.ac.uk/steps/glossary/probability_distributions.html#randvar

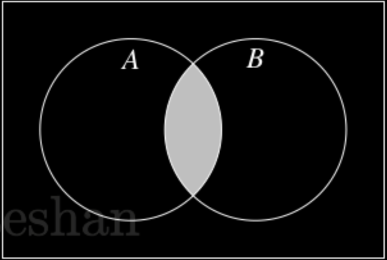
JOINT PROBABILITY

Given any two events E_1 and E_2, the probability of their conjunction

$$P(E_1, E_2) = P(E_1 \cap E_2) \qquad (2)$$

is called the ***joint probability***[2] of E_1 and E_2. This probability, E_1 and E_2, occurs simultaneously.

Example The probability of the the first letter of 't' and the second letter 'h' is $P(F =' t', S =' h')$. The joint probability should be as large as the probability of $P('the')$



$P(A) =$ size of A relative to Ω

$P(A, B) =$ size of $A \cap B$ relative to Ω

[2] https://cs.brown.edu/courses/csci1460/assets/files/langmod.pdf

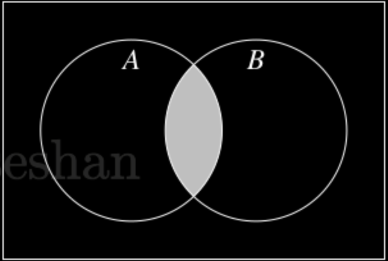
CONDITIONAL PROBABILITY

When we have partial knowledge influencing the outcome of an experiment, we use it to update the outcome.

The ***conditional probability*** $P(E_2|E_1)$ is the probability of event E_2 given that event E_1 has occurred. $P(E_2|E_1)$ is defined as:

$$P(E_2|E_1) = \frac{P(E_1,E_2)}{P(E_1)}, \text{ if } P(E_1) > 0 \quad (3)$$

$$= \frac{P(E_1 \cap E_2)}{P(E_1)} \quad (4)$$



$P(A) =$ size of A relative to Ω

$P(A,B) =$ size of $A \cap B$ relative to Ω

$P(A|B) =$ size of $A \cap B$ relative to B

CONDITIONAL PROBABILITY - BIGRAM EXAMPLE

Let consider a corpus of Kinematics problems in physics that contains about 280+ problems (*very small corpus*).

- Bigram Sample Space - $\{w_1, w_2\} \in \Omega = 3767$
- $A = \{w_1, w_2\} = \{$**average**$, *\}$ - bigram starting with *average*
- $B = \{w_1, w_2\} = \{*, $**speed**$\}$ - bigram ending with *speed*
- $P(average) = 0.036$
- $P(speed) = 0.114$
- $P(average, speed) = 0.004$
- $P(speed|average) = \frac{0.004}{0.036} = 0.111$
- $P(average|speed) = \frac{0.004}{0.114} = 0.035$

CONDITIONAL PROBABILITY - TRIGRAM EXAMPLE

Let consider a corpus of Kinematics problems in physics that contains about 280+ problems (*very small corpus*).

- Trigram Sample Space - $\{(w_1, w_2), w_3\} \in \Omega = 5902$
- $A = \{(w_1, w_2), w_3\} = \{\textbf{average,speed}, of\}$ - trigram starting with $(average, speed)$
- $B = \{(w_1, w_2), w_3\} = \{average, speed\}, \textbf{of}\}$ - trigram ending with of
- $C = \{(w_1, w_2), w_3\} = \{average, speed\}, \textbf{for}\}$ - trigram ending with for
- $D = \{(w_1, w_2), w_3\} = \{average, speed\}, \textbf{during}\}$ - bigrams ending with $during$
- $P(average, speed) = 0.0032; \quad P(average, speed, of) = 0.0007$
- $P(average, speed, for) = 0.0005; \quad P(average, speed, during) = 0.0002$

$$P(of|average, speed) = \frac{0.0007}{0.0032} = 0.21875$$
$$P(for|average, speed) = \frac{0.0005}{0.0032} = 0.15576$$
$$P(during|average, speed) = \frac{0.0002}{0.0032} = 0.0625$$

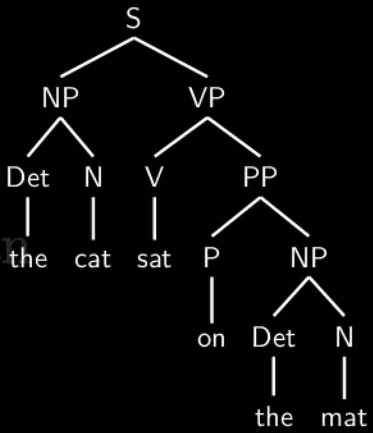
INDEPENDENCE

- Two events are dependent if the probability of one relies on occurrence of the other; if there is no such interaction, then the events are independent
- Two events E_1 and E_2 are independent if and only if $P(E_1, E_2) = P(E_1)P(E_2)$
- OR
    - $P(E_1) = P(E_1|E_2)$
    - $P(E_2) = P(E_2|E_1)$

- Example
    - $P(average) = 0.036$
    - $P(speed) = 0.114$
    - $P(average, speed`) = 0.004$
- The bigram $\{average, speed\}$ did not happen by chance. The words $average, speed$ are **NOT** independent

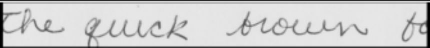
THE LANGUAGE MODEL

- Natural language sentences can be described by parse trees which use the morphology of words, syntax and semantics
- Probabilistic thinking - finding how likely a sentence occurs or formed, given the word sequence.
- In probabilistic world, the Language model is used to assign a probability $P(W)$ to every possible word sequence W.

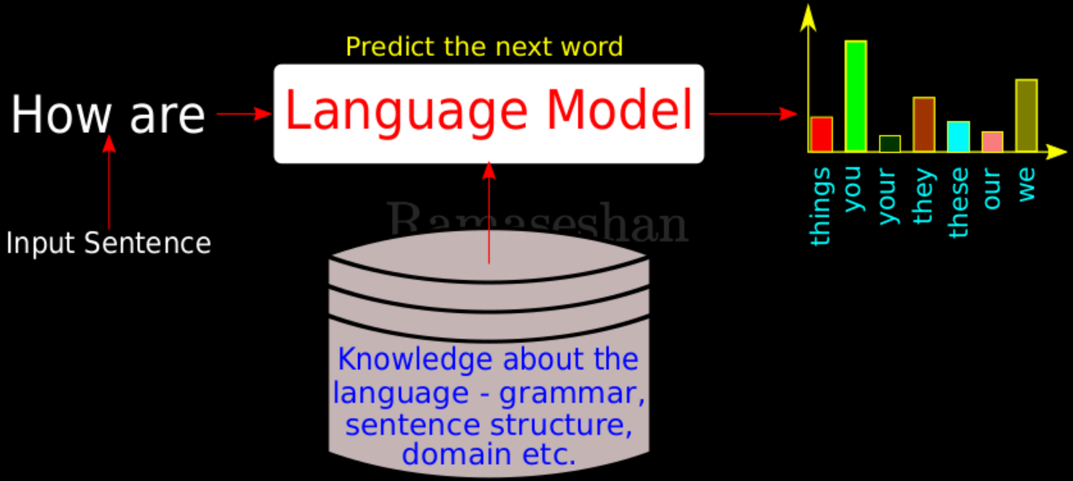


The current research in Language models focuses more on building the model from the huge corpus of text

APPLICATIONS

Application	Sample Sentences
Speech Recognition	Did you hear ***Recognize speech*** or Wreck a nice beach?
Context sensitive Spelling	One upon a ***tie***, ***Their*** lived aking
Machine translation	artwork is good $\rightarrow$ l'oeuvre est bonne
Sentence Completion	Complete a sentence as the previous word is given - GMail
OCR and Hand-written recognition	 The quick brown fo

A SIMPLE LANGUAGE MODEL IMPLEMENTATION



WHY PROBABILISTIC MODEL

- Speech recognition systems cannot depend on the processed speech signals. It may require the help of a language model and context recognizer to convert a speech to correct text format.
- As there are multiple combinations for a word to be in the next slot in a sentence, it is important for language modeling to be probabilistic in nature - judgment about the fluency of a sequence of words returns the probability of the sequence
- The probability of the next word in a sequence is real number $[0, 1]$
- The combination of words with high-probability in a sentence are more likely to occur than low-probability ones
- A probabilistic model continuously estimates the rank of the words in a sequence or phrase or sentence in terms of frequency of occurrence

PROBABILISTIC LANGUAGE MODEL

Goal: Compute the probability of a sequence of words

$$P(W) = P(w_1, w_2, w_3, ...w_n) \tag{5}$$

Task: To predict the next word using probability. Given the context, find the next word using

$$P(w_n | w_1, w_2, w_3, \ldots, w_{n-1}) \tag{6}$$

A model which computes the probability for (5) or predicting the next word (6) or complete the partial sentence is called as Probabilistic Language Model.
The goal is to learn the joint probability function of sequences of words in a language.
The probability of $P(\text{The cat roars })$ is less likely to happen than $P(\text{The cat meows})$

CHAIN RULE

It is difficult to compute the probability of the entire sequence $P(w_1, w_2, w_3, \ldots, w_n)$?
Chain rule is used to decompose the joint probability of a sequence into a product of conditional probability

$$\begin{aligned} P(W) &= P(w_1, w_2, w_3, \ldots, w_n) = P(w_1^n) && (7) \\ &= P(w_1)P(w2|w_1)P(w3|w_2, w_1)\ldots P(w_n|w_{n-1}, w_{n-2}, w_{n-3}, \ldots, w_1) && (8) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) && (9) \end{aligned}$$

- It is possible to $P(w|h)$, but it does not really help in reducing the computational complexity
- We use innovative ways to string words to form new sentences
- Finding the probability for a long sentence may not yield good outcome as the context may never occur in the corpus
- Short sequences may provide better results

MARKOV ASSUMPTION

Markov Assumption:The future behavior of a dynamic system depends on its recent history and not on the entire history

The product of the conditional probabilities can be written approximately for a bigram as

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-1}) \tag{10}$$

Equation (10) can be generalized for an *n-gram* as

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-K+1}^{k-1}) \tag{11}$$

Now, the joint probability of a sequence can be re-written as

$$\begin{align} P(W) &= P(w_1, w_2, w_3, \ldots, w_n) = P(w_1^n) \tag{12} \\ &= P(w_1)P(w2|w_1)P(w3|w_2, w_1)\ldots P(w_n|w_{n-1}, w_{n-2}, w_{n-3}, \ldots, w_1) \tag{13} \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \tag{14} \\ &\approx \prod_{k=1}^{n} P(w_k|w_{k-K+1}^{k-1}) \tag{15} \end{align}$$

TARGET AND CONTEXT WORDS

Next word in the sentence depends on its immediate past words, known as context words

$$P(w_{k+1} | \underbrace{w_{i-k}, w_{i-k+1}, \dots, w_k}_{\text{Context words}})$$

n-grams

unigram	-	$P(w_{k+1})$
bigram	-	$P(w_{k+1} \mid w_k)$
trigram	-	$P(w_{k+1} \mid w_{k-1}, w_k)$
4-gram	-	$P(w_{k+1} \mid w_{k-2}, w_{k-1}, w_k)$

LANGUAGE MODELING USING UNIGRAMS

- A unigram language model all words are generated independently $W_, W_2, W_3, \ldots..W_n$ and none of them depend on the other
- This is not a good model for language generation
- It may generate ***the the the the*** as a sentence

GENERATIVE MODEL

- Generates a document containing N words using n-gram
- A good model assigns higher probability to the word that actually occurs

$$P(\mathbf{W}) = P(N)\prod_{i=1}^{N} P(W_i) \tag{16}$$

- The location of the word in the document is not important
- P(N) is the distribution over N and is same for all documents. Hence it is ignored
- W_i, to be estimated in this model is $P(W_i)$ and it must satisfy $\sum_{i=1}^{N} P(w_i) = 1$

MAXIMUM LIKELIHOOD ESTIMATE

- One of the methods to find the unknown parameter(s) is the use of Maximum Likelihood Estimate
- Estimate the parameter value for which the observed data have the highest probability
- Training data may not have all the words in the vocabulary
- If a sentence with an unknown word is presented, then the MLE is zero.
- Add a smoothing parameter to the equation without affecting the overall probability requirements

$$P(\mathbf{W}) = \frac{C_{w_i} + \alpha}{C_W + \alpha |W|} \tag{17}$$

BIGRAM LANGUAGE MODEL

- Bigram language model generates a sequence one word at a time, starting with the first word and then generating each succeeding word conditioned on the previous one[3]
- A bigram language model is defined as follows:

$$P(\mathbf{W}) = \prod_{i=1}^{n+1} P(w_i|w_{i-1}), \quad (18)$$

where $\mathbf{W} = w_1, w_2, w_3, \ldots, w_n$

- Estimate the parameter $P(w_i|w_{i-1})$ for all bigrams
- The parameter estimation does not depend on the location of the word
- If we consider the sentence as a sequence in time, they are time-invariant MLE picks up the word that is $\frac{n_{w,w'}}{n_{w,o}}$ where nw, w' is the number of times the words w_1, w' occur together and $n_{w,o}$ is the number of times the word w appears in the bigram sequence

[3] https://cs.brown.edu/courses/csci1460/assets/files/langmod.pdf

PROBABILISTIC LANGUAGE MODEL - EXAMPLE

Peter Piper picked a peck of pickled peppers
A peck of pickled peppers Peter Piper picked
If Peter Piper picked a peck of pickled peppers
Where's the peck of pickled peppers Peter Piper picked?

—

The joint probability of a sentence formed with n words can be expressed as a product conditional probabilities - we use immediate context and not the entire history

$P(w_1|\langle S\rangle) \times P(w_2|w_1) \times \ldots P(\langle E\rangle|w_n)$

and $P(w_{i+1}|w_i) = \frac{|w_i.w_{i+1}|}{|w_i|}$

—

What is the probability of these sentences?
P(Peter Piper picked)
P(Peter Piper picked peppers)

Bigram	**Frequency**
$\langle S\rangle$Peter	1
Peter Piper	4
Piper picked	4
picked a	2
a peck	2
peck of	4
pickled peppers	4
peppers $\langle E\rangle$	1
$\langle S\rangle A$	1
A peck	1
of pickled	4
peppers Peter	2
...	..
$\langle S\rangle$ ...	1

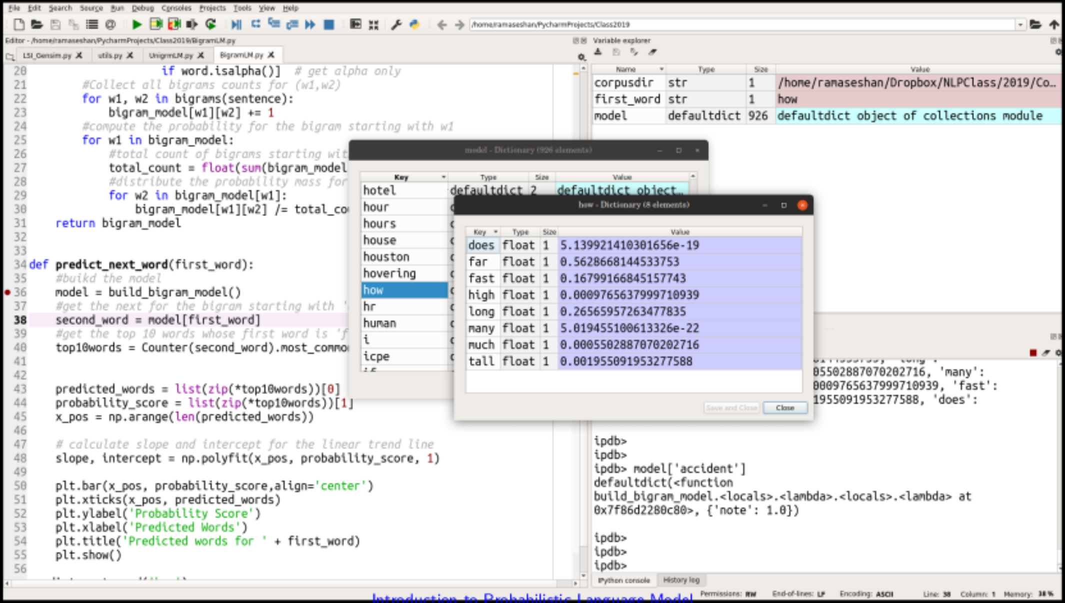
BUILDING A BIGRAM MODEL - CODE

```
#compute the bigram model
def build_bigram_model():
    bigram_model = collections.defaultdict(
        lambda: collections.defaultdict(lambda: 0))
    for sentence in kinematics_corpus.sents():
        sentence = [word.lower() for word in sentence
                    if word.isalpha()]  # get alpha only
        #Collect all bigrams counts for (w1,w2)
        for w1, w2 in bigrams(sentence):
            bigram_model[w1][w2] += 1
        #compute the probability for the bigram containing w1
        for w1 in bigram_model:
            #total count of bigrams conaining w1
            total_count = float(sum(bigram_model[w1].values()))
            #distribute the probability mass for all bigrams starting with w1
            for w2 in bigram_model[w1]:
                bigram_model[w1][w2] /= total_count
    return bigram_model
```

BUILDING A BIGRAM MODEL - CODE

```
def predict_next_word(first_word):
    #buikd the model
    model = build_bigram_model()
    #get the next for the bigram starting with 'word'
    second_word = model[first_word]
    #get the top 10 words whose first word is 'first_word'
    top10words = Counter(second_word).most_common(10)

    predicted_words = list(zip(*top10words))[0]
    probability_score = list(zip(*top10words))[1]
    x_pos = np.arange(len(predicted_words))

    plt.bar(x_pos, probability_score,align='center')
    plt.xticks(x_pos, predicted_words)
    plt.ylabel('Probability Score')
    plt.xlabel('Predicted Words')
    plt.title('Predicted words for ' + first_word)
    plt.show()

predict_next_word('how')
```

MODEL PARAMETERS - BIGRAM EXAMPLE



File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/ramaseshan/PycharmProjects/Class2019
Editor - /home/ramaseshan/PycharmProjects/Class2019/BigramLM.py
LSI_Gensim.py
utils.py
UnigrmLM.py
BigramLM.py
Variable explorer
model - Dictionary (926 elements)
how - Dictionary (8 elements)
Save and Close
Close
Python console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 38
Column: 1
Memory: 38 %

```python
                    if word.isalpha()]  # get alpha only
        #Collect all bigrams counts for (w1,w2)
        for w1, w2 in bigrams(sentence):
            bigram_model[w1][w2] += 1
        #compute the probability for the bigram starting with w1
        for w1 in bigram_model:
            #total count of bigrams starting wit
            total_count = float(sum(bigram_model
            #distribute the probability mass for
            for w2 in bigram_model[w1]:
                bigram_model[w1][w2] /= total_co
    return bigram_model


def predict_next_word(first_word):
    #buikd the model
    model = build_bigram_model()
    #get the next for the bigram starting with '
    second_word = model[first_word]
    #get the top 10 words whose first word is 'f
    top10words = Counter(second_word).most_commo


    predicted_words = list(zip(*top10words))[0]
    probability_score = list(zip(*top10words))[1]
    x_pos = np.arange(len(predicted_words))

    # calculate slope and intercept for the linear trend line
    slope, intercept = np.polyfit(x_pos, probability_score, 1)

    plt.bar(x_pos, probability_score,align='center')
    plt.xticks(x_pos, predicted_words)
    plt.ylabel('Probability Score')
    plt.xlabel('Predicted Words')
    plt.title('Predicted words for ' + first_word)
    plt.show()
```

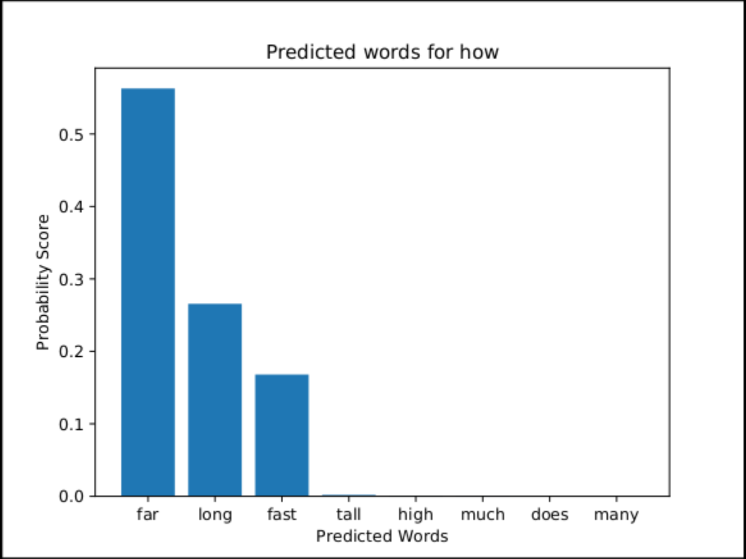
Name	Type	Size	Value
corpusdir	str	1	/home/ramaseshan/Dropbox/NLPClass/2019/Co…
first_word	str	1	how
model	defaultdict	926	defaultdict object of collections module

Key	Type	Size	Value
hotel	defaultdict	2	defaultdict object…
hour			
hours			
house			
houston			
hovering			
how			
hr			
human			
i			
icpe			

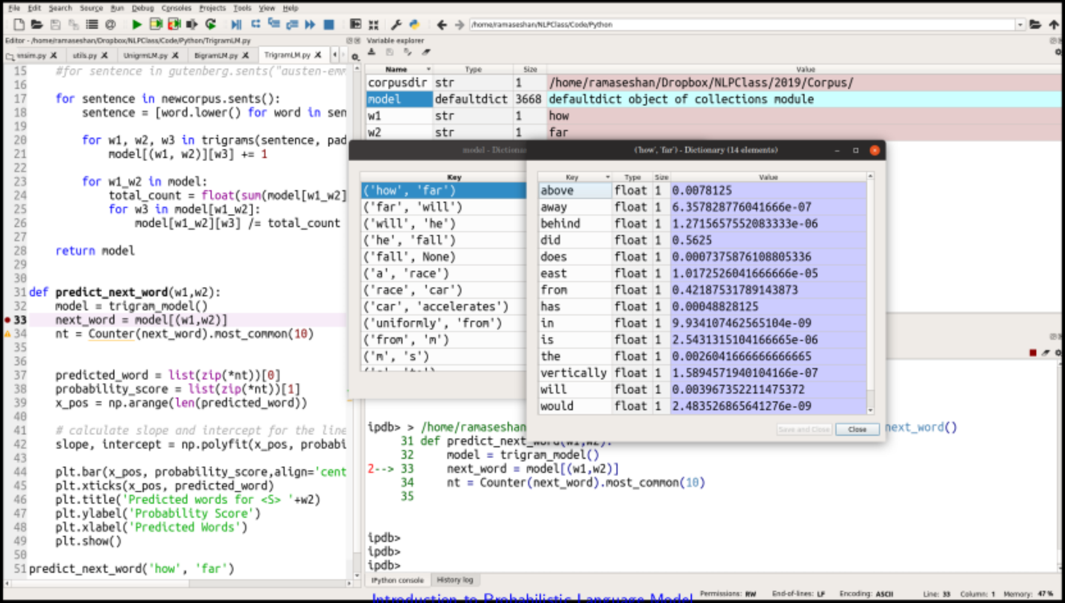
Key	Type	Size	Value
does	float	1	5.139921410301656e-19
far	float	1	0.5628668144533753
fast	float	1	0.16799166845157743
high	float	1	0.0009765637999710939
long	float	1	0.26565957263477835
many	float	1	5.019455100613326e-22
much	float	1	0.0005502887070202716
tall	float	1	0.001955091953277588

```
05502887070202716, 'many':
0009765637999710939, 'fast':
1955091953277588, 'does':

ipdb>
ipdb>
ipdb> model['accident']
defaultdict(<function
build_bigram_model.<locals>.<lambda>.<locals>.<lambda> at
0x7f86d2280c80>, {'note': 1.0})

ipdb>
ipdb>
ipdb>
```

BIGRAM MODEL - NEXT WORD PREDICTION



MODEL PARAMETERS - TRIGRAM EXAMPLE


File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/ramaseshan/NLPClass/Code/Python
Editor - /home/ramaseshan/Dropbox/NLPClass/Code/Python/TrigramLM.py
ensim.py utils.py UnigrmLM.py BigramLM.py TrigramLM.py
Variable explorer
model - Dictiona
('how', 'far') - Dictionary (14 elements)
Save and Close
Close
IPython console
History log
Permissions: RW End-of-lines: LF Encoding: ASCII Line: 33 Column: 1 Memory: 47 %

```python
    #for sentence in gutenberg.sents("austen-emm

    for sentence in newcorpus.sents():
        sentence = [word.lower() for word in sen

        for w1, w2, w3 in trigrams(sentence, pad
            model[(w1, w2)][w3] += 1

        for w1_w2 in model:
            total_count = float(sum(model[w1_w2]
            for w3 in model[w1_w2]:
                model[w1_w2][w3] /= total_count

    return model

def predict_next_word(w1,w2):
    model = trigram_model()
    next_word = model[(w1,w2)]
    nt = Counter(next_word).most_common(10)


    predicted_word = list(zip(*nt))[0]
    probability_score = list(zip(*nt))[1]
    x_pos = np.arange(len(predicted_word))

    # calculate slope and intercept for the line
    slope, intercept = np.polyfit(x_pos, probabi

    plt.bar(x_pos, probability_score,align='cent
    plt.xticks(x_pos, predicted_word)
    plt.title('Predicted words for <S> '+w2)
    plt.ylabel('Probability Score')
    plt.xlabel('Predicted Words')
    plt.show()

predict_next_word('how', 'far')
```

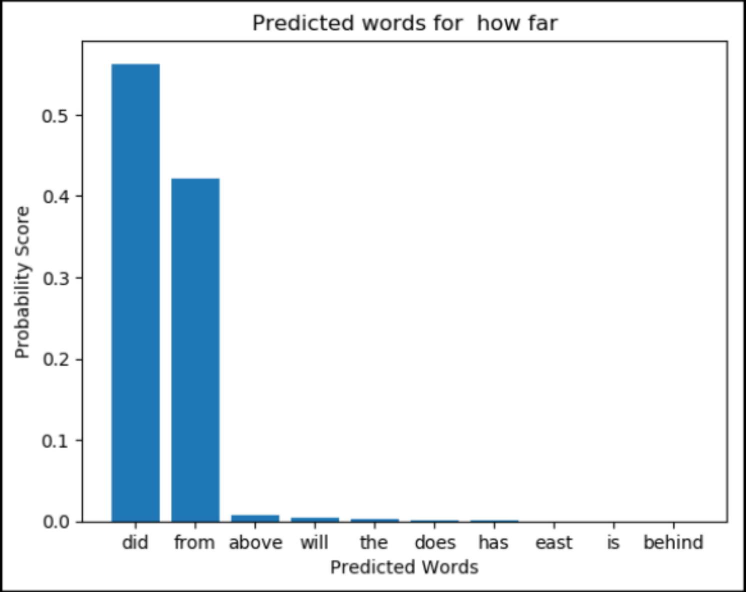
Name	Type	Size	Value
corpusdir	str	1	/home/ramaseshan/Dropbox/NLPClass/2019/Corpus/
model	defaultdict	3668	defaultdict object of collections module
w1	str	1	how
w2	str	1	far

Key
('how', 'far')
('far', 'will')
('will', 'he')
('he', 'fall')
('fall', None)
('a', 'race')
('race', 'car')
('car', 'accelerates')
('uniformly', 'from')
('from', 'm')
('m', 's')

Key	Type	Size	Value
above	float	1	0.0078125
away	float	1	6.357828776041666e-07
behind	float	1	1.2715657552083333e-06
did	float	1	0.5625
does	float	1	0.0007375876108805336
east	float	1	1.0172526041666666e-05
from	float	1	0.42187531789143873
has	float	1	0.00048828125
in	float	1	9.934107462565104e-09
is	float	1	2.5431315104166665e-06
the	float	1	0.0026041666666666665
vertically	float	1	1.5894571940104166e-07
will	float	1	0.003967352211475372
would	float	1	2.483526865641276e-09

```
ipdb> > /home/ramaseshan                                    next_word()
     31 def predict_next_word(w1,w2):
     32     model = trigram_model()
2--> 33     next_word = model[(w1,w2)]
     34     nt = Counter(next_word).most_common(10)
     35


ipdb>
ipdb>
ipdb>
```

TRIGRAM MODEL - NEXT WORD PREDICTION



PERPLEXITY

Perplexity is a measurement of how well a probability model predicts a sample. Perplexity is defined as

$$\text{For bigram model, } PP(W_N) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \tag{19}$$

$$\text{For trigram model } PP(W_N) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1}w_{i-2})}} \tag{20}$$

A good model gives maximum probability to a sentence or minimum perplexity to a sentence

UNKNOWN WORDS

- In a closed vocabulary language model, there is no unknown words or ***out of vocabulary words (OOV)***
- In an open vocabulary system, you will find new words that are not present in the trained model
- Pick words below certain frequency and replace them as OOV.
- Treat every OOV as a regular word
- During testing, the new words would be treated as OOV and the corresponding frequency will be used for computation
- This eliminates zero probability for sentences containing OOV

CURSE OF DIMENSIONALITY

- A fundamental problem that makes language modeling and other learning problems difficult is the curse of dimensionality
- It is particularly obvious in the case when one wants to model the joint distribution between many discrete random variable
- If one wants to estimate the joint probability distribution of 10 words in a language with a million words as vocabulary, then we need to estimate $1000000^{10} - 1 = 10^{60} - 1$ free parameters

.

EXERCISE

- Extend the program[4,5] that predicts the word into another program that computes the probability and perplexity of a test sentence
  1. Build the language model using corpus
     - Big corpus $\rightarrow$ Long time to build a model
     - Development $\rightarrow$ Choose a smaller corpus
     - Testing/Production $\rightarrow$ use a bigger corpus to build your model
  2. Check the model parameters using the debugger
  3. Once satisfied with the learned model, test your sentences using the model
- **Input**:
  1. A sentence consisting of words in the corpus that you have used for creating the language model
  2. A sentence with one more words that are OOV
- **Output**:Probability and perplexity of the input sentence
- Try this exercise for trigram and 4-gram language models

[4] https://github.com/Ramaseshanr/anlp/blob/master/BigramLM.ipynb
[5] https://github.com/Ramaseshanr/anlp/blob/master/TrigramLM.ipynb