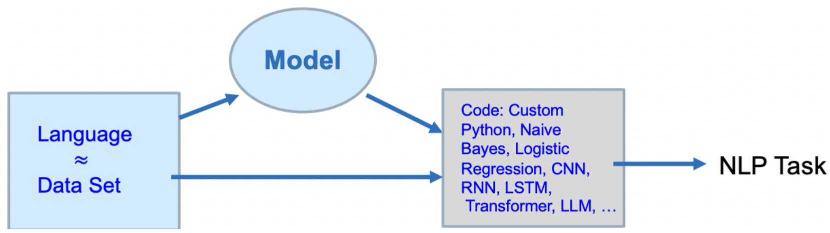
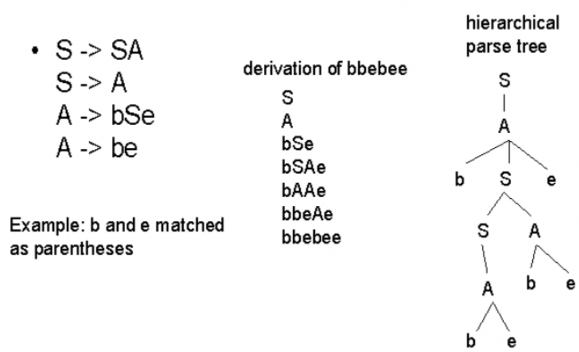
CAS CS 505
Lec 3

Language Models, Bag-of-Words, N-grams, Skip-Grams

1. Language Models
    a. A Language Model is a simplified representation of a language which facilitates an NLP task, where
        i. A language (potentially infinite) is approximated by a (finite) data set; and
        ii. The model is a set of (simplified) assumptions about the language, embodied by the algorithms and data structures of your program.
        iii. 

    b. NLP systems rely on models to capture knowledge of about a language, and as a representation for texts which facilitate an NLP task.
        i. Example: Context-Free Grammars (Chomsky, Backus-Naur)
        ii. 

2. Language Modeling: Word Representations
    a. Before diving into the subject of Language Models, let's prepare a bit by talking about an essential component of language modeling…
    b. Last time we discussed how to represent characters, as integer ASCII codes in the range [0 .. 127]. But how do we represent words?
    c. Bad idea: word = sequence of ASCII codes
    d. Why is this bad?
        i. Variable length (every word length has to be the same, which leads to having lots of _ after the word) → ex) dog____________________
        ii. Information contains confusing correspondences:
            1. "to" very similar to "too" "dog" same chars as "god"
    e. (Neural networks will find these difficult to learn.)

f. There are two principal representations, both based on constant-length arrays (or vectors).

g. One-Hot Encoding

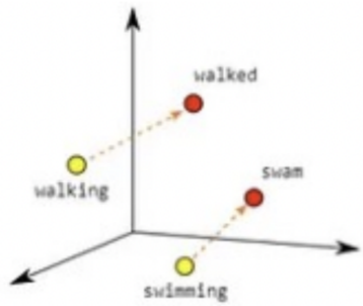
i.

Color
Red
Green
Blue
Green

→

Red	Green	Blue
1	0	0
0	1	0
0	0	1
0	1	0

h. Word Embedding (we'll do these in a few weeks) – putting words that are similar close to each other

i. 

i. We generically call sequences/lists/arrays by the standard term vector. For simplicity, we foten write them as Python lists here

3. Language Modeling: One-Hot Encoding for Words
    a. Basic Idea of One-Hot Encoding:
        i. Create a vocabulary list of length N of all distinct words in the text (the ordering is fixed but arbitrary).
        ii. The representation of the kth word in the list is an array/vector of N integers, with a 0 in every position except for a single1 in position k.
        iii. Example: "John likes to watch movies. Mary likes movies too."
        iv. Vocabulary list: ["John", "likes", "Mary", "movies", "to", "too", "watch"]
        0 1 2 3 4 5 6
        v. One-Hot Encodings:
        "movies" ⇨ [0, 0, 0, 1, 0, 0, 0]
        "likes" ⇨ [0, 1, 0, 0, 0, 0, 0]
        vi. Plus:
            1. Vectors have same length
            2. Spelling is completely irrelevant
        vii. Minus
            1. Very long vectors (typically 10,000 or more) → can eliminate many words that do not matter much for later

4. Language Models: Bag-of-Words
    a. Examples of Models: Bag of Words (BOW)
        i. The BOW model represents a text (sentence, sequence of words, entire corpus) as a multiset (bag) of all words in the text, i.e, just the vocabulary, no information about order of words!
            1. Useful in classification (Spam Vs Ham)
        ii. Text: “John likes to watch movies. Mary likes movies too.”

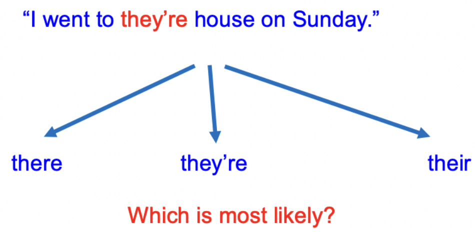
Vocabulary list:	[“John”,	“likes”,	“Mary”,	”movies”,	“to”,	“too”,	”watch”]
	0	1	2	3	4	5	6

        iii. **BOW model of text:** [1, 2, 1, 2, 1, 1, 1]
        iv. The multiplicity (frequency) of words do not really matter
            1. Ex) “The movie is bad bad bad” is similar to “The movie is bad”
        v. Do this for large number of data
        vi. Alternate BOW representations
            1. We might only consider the presence (0/1) of a word, not its frequency (as if “Set of Words”)
            2. Since most BOW vectors are sparce, we might want to store them as a dictionary
            3. { "John" : 1, "likes" : 2, "to" : 1, "watch" : 1, "movies" : 2, "Mary" : 1, "too" : 1 }
        vii. What is the relationship between One-Hot Encodings and a BOW model?
            1. The BOW model of a text is the array sum of the one-hot encodings
            2.

“John”	[1, 0, 0, 0, 0, 0, 0]
“likes”	[0, 1, 0, 0, 0, 0, 0]
“to”	[0, 0, 0, 0, 1, 0, 0]
“watch”	[0, 0, 0, 0, 0, 0, 1]
“movies”	[0, 0, 0, 1, 0, 0, 0]
“Mary”	[0, 0, 1, 0, 0, 0, 0]
“likes”	[0, 1, 0, 0, 0, 0, 0]
“movies”	[0, 0, 0, 1, 0, 0, 0]
“too”	[0, 0, 0, 0, 0, 1, 0]
+	
	[1, 2, 1, 2, 1, 1, 1]

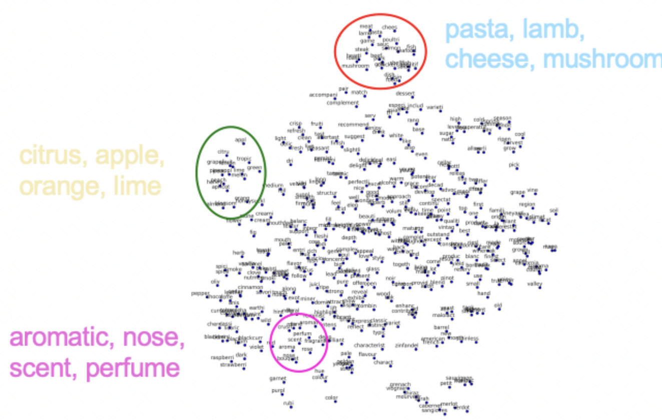
    b. Reducing words
        i. Very common words such as “the” “an”
        ii. Very uncommon words
5. Language Models: Probabilistic Language Models
    a. Probabilistic Language Model: Assign a probability to text components (letters, words, sentences, ...)

b. This is very useful to work with the ambiguous nature of human languages, and very amenable to computation:
    i. “Given K choices for some ambiguous input, choose the most probable one.”



c.
    i. Which one appears the most in the corpus? If you feed wrong corpus, you will get wrong

6. Language Models: Vector Space Language Models
    a. Vector space models use a vector in M-dimensional space to represent
        i. Words
        ii. Sentences
        iii. Texts



b.
    c. This is the current SOTA (“Stage Of The Art”) for language modeling. → very useful computation since words that are similar are close to each other

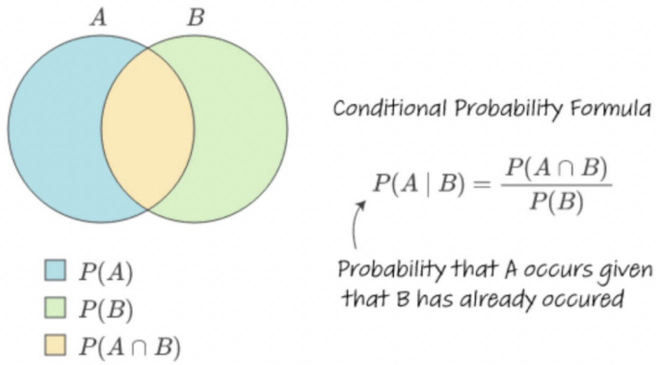
7. Probabilistic Language Models
    a. Main Idea of PLMs: Assign a probability to a sequence of words. Why?
        i. Some words are more probable than others
    b. Machine Translation:
        i. EX)Translating from Japanese to English and there are two options
        ii. P(high winds tonight) > P(large winds tonight)
            1. Look for every occurrence of those words in the corpus
    c. Spelling Correction:
        i. The office is about fifteen minuets from my house
        ii. P(about fifteen minutes from) > P(about fifteen minuets from)

d. Speech Recognition:
    i. P(I saw a van) >> P(eyes awe of an)
    ii. Could be any sequence and length of words (corpus might not have enough examples to capture all the data)
e. Summarization, Q&A, etc.
f. The main task: compute the probability of a sentence or sequence of words:
    i. P(W) = P(w1,w2,w3,w4,w5…wn)
    ii. Words might not occur or occur → handle later
g. Subtask: compute the conditional probability of the next word
    i. P(w5 | w1,w2,w3,w4) “The weather is ? “
h. A model that computes either of these: P(W) or P(wn | w1,w2…wn-1) is called a probabilistic language model (often, just “language model”).

i.
j. It is possible to apply this framework to any sequence (predicting the next sequence given a sequence)
    i. Letters in a word
    ii. Pitches in a melody
    iii. Phonemes in a voice signal
    iv. Sentences in a paragraph
    v. Topics in a discourse.
k. The amount of data is very significant because we need to consider all the likelihoods
l. How to compute P(W)?
    i. How to compute this joint probability
        1. P(I went to their house on Sunday)
    ii. Intuition: let’s rely on the Chain Rule of Probability

8. Calculating Probabilities
    a. Recall the definition of conditional probabilities
        i. P(A|B) = P(B,A) / P(B)
        ii. Rewriting: P(B,A) = P(B) * P(A|B)

iii. For this LM, we will think of B,A as a sequence: B happens, then A happens. Thus, P(A|B) = “given that B has happened, what is the probablity that A happens”?

b. 

c. More variables:
   i. P(A,B,C,D) = P(A) × P(B|A) × P(C|A,B) × P(D|A,B,C)

d. General Chain Rule:
   i. P(x1,x2,x3,…,xn) = P(x1) × P(x2|x1) × P(x3|x1,x2) × … × P(xn|x1,…,xn-1)

e. The Chain Rule applied to compute joint probability of words in sentence:
   i. “I went to their house on Sunday.”
   ii.

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

   iii. P(I went to their house on Sunday.) =
   P(I) × P(went | I) × P(to | I went) × P(their | I went to)
   × P(house | I went to their) × P(on | I went to their house)
   x P(Sunday | I went to their house on)

f. How to estimate these probabilities?

g. Could we just count and divide?
   i.

$$\text{P(Sunday | I went to their house on)} = \frac{\text{Count(I went to their house on Sunday)}}{\text{Count(I went to their house on)}}$$

   ii. Not realistic
   iii. In an infinite set of sentences, the probability of any distinct sequence of words is 0. So a data set is a small sample which hopefully represents the essential features of the language.
   iv. But realistic data sets never have enough sample sequences, and sequences might be very long or simply not exist in your data.

h. Markov Assumption: Finite history!

i. Only consider N-1 words of left context, for some fixed N.

j. So, if N = 2 , “I went to their house on Sunday” becomes

   i. I went

      went to

      to their

      their house

      house on

      on Sunday

k. If N = 3, “I went to their house on Sunday” becomes

   i. I went to

      went to their

      to their house

      their house on

      house on Sunday

l. Terminology: An N-Gram is a sequence of N contiguous words from the data set.

   i. unigram = 1-gram

   ii. bigram = 2-gram

   iii. trigram = 3-gram, etc

m. Markov Assumption: Only consider N-1 words of left context.

n. Thus, for a sequence of length M,

$$P(w_1 w_2 \ldots w_M) \approx \prod_{N \leq i \leq M-N} P(w_i | w_{i-N+1} \ldots w_{i-1})$$

Bigram Example (N = 2)

P(<s> I went to their house on Sunday </s>)=

P(I |<s>) × P(went | I) × P(to | went) × P(their | to)

× P(house | their) × P(on | house)

x P(Sunday | on) x P(</s> | Sunday)

o.

   i. <s> is where the sentence begins, so you lose the information that the start of the sentence is “I”

$$P(\text{ I } | \text{ <s> }) \approx \frac{C(\text{ <s> I })}{C(\text{ <s> })}$$

$$P(\text{ went } | \text{ I }) \approx \frac{C(\text{ I went })}{C(\text{ I })}$$

p.

q. Note that this calculation involves finding the number of occurrences of an N-gram and of an (N-1)-gram (the prefix)!

Trigram Example (N = 3)

P(<s> I went to their house on Sunday </s>)=
P(I |<s>) × P(went | I) × P(to | went) × P(their | to)
× P(house | their) × P(on | house)
x P(Sunday | on) x P(</s> | Sunday)

$$P(\text{went} \mid \text{<s> I}) = \frac{C(\text{<s> I went})}{C(\text{<s> I})}$$

$$P(\text{to} \mid \text{I went}) = \frac{C(\text{I went to})}{C(\text{I went})}$$

r.

s. In trigram, the beginning of the sentence has no length
    i. Start with bigram for once → go trigram

t. Similarly, when doing pentagram,
    i. Unigram → bigram → trigram → etc.. → pentagram

u. Remarks: This is almost trivial to code after you have separated your text into words and sentences.

v. For small N, it will be reasonably efficient.

w. BUT, it does NOT capture the recursive/nesting structure inherent in complex sentences:
    i. My friend Bill, who went to the same high school that I did –Pennsbury, which is in Fairless Hills in PA—lives in his car, and he called me yesterday (or the day before, I forget).

x. Example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

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

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

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

    i.

9. Generative Language Models

a. A clever feature of this model is that it can easily generate sentences.

b. For bigrams, after calculating the probability of all bigrams appearing in the data.
    i. Pick a probable <s> w1
    ii. Pick a probable bigram w1 w2
    iii. … etc. …
    iv. End when you generate a bigram wk </s>

c. You may not want to always choose the most likely, or you will not be able to generate many sentences! → it will always generate the same sentence

d. So choose randomly from the probability distribution of next words.