

Deep Learning with Applications in Natural Language Processing

Mihaela Breabăn
Mădălina Răschip
Diana Trandabăț

Academic year 2025-2026

Language Models

- Speech recognition
 - “I ate a cherry” is a more likely sentence than “Eye eight uh Jerry”
- OCR & Handwriting recognition
 - More probable sentences are more likely correct readings.
- Machine translation
 - More likely sentences are probably better translations.
- Generation
 - More likely sentences are probably better NL generations.
- Context sensitive spelling correction
 - “Their are problems wit this sentence.”

Completion Prediction

- A language model also supports predicting the completion of a sentence.
 - Please turn off your cell _____
 - Your program does not _____
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
-

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- W

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- Wh

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- Wha

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What d

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
- What do

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
 - What do you think the next letter is?

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
-

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do

Letter-based Language Models

- Shannon's Game
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do you

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do you think

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do you think the

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do you think the next

Letter-based Language Models

- **Shannon's Game**
- Guess the next letter:
 - What do you think the next letter is?
- Guess the next word:
 - What do you think the next word is?

Real Word Spelling Errors

- They are leaving in about fifteen *minuets* to go to her house.
- The study was conducted mainly *be* John Black.
- Hopefully, all *with* continue smoothly in my absence.
- Can they *lave* him my messages?
- I need to *notified* the bank of....
- He is trying to *fine* out.

Corpora

- Corpora are online collections of text and speech
 - Brown Corpus
 - Wall Street Journal
 - AP newswire
 - Hansards
 - Timit
 - DARPA/NIST text/speech corpora (Call Home, Call Friend, ATIS, Switchboard, Broadcast News, Broadcast Conversation, TDT, Communicator)
 - TRAINS, Boston Radio News Corpus

Terminology

- **Sentence**: unit of written language
- **Utterance**: unit of spoken language
- **Word Form**: the inflected form as it actually appears in the corpus
- **Lemma**: an abstract form, shared by word forms having the same **stem**, part of speech, word sense – stands for the class of words with same **stem**
- **Types**: number of distinct words in a corpus (vocabulary size)
- **Tokens**: total number of words

Word-based Language Models

- A model that enables one to compute the probability, or likelihood, of a sentence S , $P(S)$.
- Simple: Every word follows every other word w/ equal probability (0-gram)
 - Assume $|V|$ is the size of the vocabulary V
 - Likelihood of sentence S of length n is $= 1/|V| \times 1/|V| \dots \times 1/|V|$
 - If English has 100,000 words, probability of each next word is $1/100000 = .00001$

Word Prediction: Simple vs. Smart

- Smarter: probability of each next word is related to word frequency (unigram)
 - Likelihood of sentence $S = P(w_1) \times P(w_2) \times \dots \times P(w_n)$
 - Assumes probability of each word is independent of probabilities of other words.
- Even smarter: Look at probability *given* previous words (N-gram)
 - Likelihood of sentence $S = P(w_1) \times P(w_2|w_1) \times \dots \times P(w_n|w_{n-1})$
 - Assumes probability of each word is dependent on probabilities of other words.

Chain Rule

- Conditional Probability
 - $P(w_1, w_2) = P(w_1) \cdot P(w_2 | w_1)$
- The **Chain Rule** generalizes to multiple events
 - $P(w_1, \dots, w_n) = P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1 \dots w_{n-1})$
- Examples:
 - $P(\text{the dog}) = P(\text{the}) P(\text{dog} | \text{the})$
 - $P(\text{the dog barks}) = P(\text{the}) P(\text{dog} | \text{the}) P(\text{barks} | \text{the dog})$

Relative Frequencies and Conditional Probabilities

- Relative word frequencies are better than equal probabilities for all words
 - In a corpus with 10K word types, each word would have $P(w) = 1/10K$
 - Does not match our intuitions that some words are more likely to occur (e.g. the) than others
- Conditional probability - more useful than individual relative word frequencies
 - **dog** may be relatively rare in a corpus
 - But if we see **barking**, $P(\text{dog} | \text{barking})$ may be very large

N-Gram Models

- Estimate probability of each word given prior context.
 - $P(\text{phone} \mid \text{Please turn off your cell})$
- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only $N-1$ words of prior context.
 - Unigram: $P(\text{phone})$
 - Bigram: $P(\text{phone} \mid \text{cell})$
 - Trigram: $P(\text{phone} \mid \text{your cell})$
- The ***Markov assumption*** is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a ***kth-order Markov model***, the next state only depends on the k most recent states, therefore an N-gram model is a $(N-1)$ -order Markov model.

N-Gram Model Formulas

- Word sequences

$$w_1^n = w_1 \dots w_n$$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 | w_1)P(w_3 | w_1^2) \dots P(w_n | w_1^{n-1}) = \prod_{k=1}^n P(w_k | w_1^{k-1})$$

- Bigram approximation

$$P(w_1^n) = \prod_{k=1}^n P(w_k | w_{k-1})$$

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^n P(w_k | w_{k-N+1}^{k-1})$$

Maximum Likelihood Estimate (MLE)

- N-gram conditional probabilities can be estimated from raw text based on the ***relative frequency*** of word sequences.

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

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

- To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.

Simple N-Grams

- An **N-gram** model uses the previous N-1 words to predict the next one:
 - $P(w_n \mid w_{n-N+1} w_{n-N+2} \dots w_{n-1})$
- unigrams: $P(\text{dog})$
- bigrams: $P(\text{dog} \mid \text{big})$
- trigrams: $P(\text{dog} \mid \text{the big})$
- quadrigrams: $P(\text{dog} \mid \text{chasing the big})$

Using N-Grams

- Recall that

- N-gram: $P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$

- Bigram: $P(w_1^n) \approx \prod_{k=1}^n P(w_k | w_{k-1})$

- For a bigram grammar

- $P(\text{sentence})$ can be approximated by multiplying all the bigram probabilities in the sequence

- Example:

$P(\text{I want to eat Chinese food}) =$

$P(\text{I} | \text{<start>}) P(\text{want} | \text{I}) P(\text{to} | \text{want}) P(\text{eat} | \text{to}) P(\text{Chinese} | \text{eat}) P(\text{food} | \text{Chinese})$

A Bigram Grammar Fragment

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Additional Grammar

<start> I	.25	Want some	.04
<start> I'd	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

Computing Sentence Probability

- $P(\text{I want to eat British food}) = P(\text{I} | \langle \text{start} \rangle) P(\text{want} | \text{I}) P(\text{to} | \text{want}) P(\text{eat} | \text{to}) P(\text{British} | \text{eat}) P(\text{food} | \text{British}) = .25 \times .32 \times .65 \times .26 \times .001 \times .60 = .000080$
- vs.
- $P(\text{I want to eat Chinese food}) = .00015$
- Probabilities seem to capture “syntactic” facts, “world knowledge”
 - eat is often followed by a NP
 - British food is not too popular
- N-gram models can be trained by counting and normalization

Exercise: Computing a Language Model

- Consider the following training corpus:

there is a big house

i buy a house

they buy the new house

- Create the bigram language model for this corpus

- Compute the probability of the new sentence

they buy a big house

Exercise: Computing a Language Model

- Consider the following training corpus:

<s> there is a big house </s>

<s> i buy a house </s>

<s> they buy the new house </s>

- Create the bigram language model for this corpus

$$P(\text{there} | \text{<s>}) = 0.33$$

$$P(\text{is} | \text{there}) = 1$$

$$P(\text{a} | \text{is}) = 1$$

$$P(\text{big} | \text{a}) = 0.5$$

$$P(\text{house} | \text{big}) = 1$$

$$P(\text{</s>} | \text{house}) = 1$$

$$P(\text{i} | \text{<s>}) = 0.33$$

$$P(\text{buy} | \text{i}) = 1$$

$$P(\text{a} | \text{buy}) = 0.5$$

$$P(\text{house} | \text{a}) = 0.5$$

$$P(\text{they} | \text{<s>}) = 0.3$$

$$P(\text{buy} | \text{they}) = 1$$

$$P(\text{the} | \text{buy}) = 0.5$$

$$P(\text{new} | \text{the}) = 1$$

$$P(\text{house} | \text{new}) = 1$$

- Compute the probability of the new sentence

they buy a big house

$$P(S) = P(\text{they} | \text{<s>}) * P(\text{buy} | \text{they}) * P(\text{a} | \text{buy}) * P(\text{big} | \text{a}) * P(\text{house} | \text{big}) * P(\text{</s>} | \text{house}) =$$

$$= 0.33 * 1 * 0.5 * 0.5 * 1 * 1 = 0.082$$

N-grams Issues

- Sparse data
 - Not all N-grams are found in training data, need smoothing
- Change of domain
 - Train on Wall Street Journal, attempt to predict Shakespeare – won't work
- N-grams more reliable than (N-1)-grams
 - But even more sparse
 - Generating Shakespeare sentences with random unigrams...
 - Every enter now severally so, let
 - With bigrams...
 - What means, sir. I confess she? then all sorts, he is trim, captain.
 - Trigrams
 - Sweet prince, Falstaff shall die.

N-grams Issues

- Determine reliable sentence probability estimates
 - should have smoothing capabilities (avoid the zero-counts)
 - apply back-off strategies: if N-grams are not possible, back-off to (N-1) grams
- $P(\text{"And nothing but the truth"}) \approx 0.001$
- $P(\text{"And nuts sing on the roof"}) \approx 0$

Zipf's Law

- **Rank** (r): The numerical position of a word in a list sorted by decreasing frequency (f).
- Zipf (1949) “discovered” that:

$$f \cdot r = k \quad (\text{for constant } k)$$

- If probability of word of rank r is p_r and N is the total number of word occurrences:

$$p_r = \frac{f}{N} = \frac{A}{r} \quad \text{for corpus indep. const. } A \approx 0.1$$

Predicting Occurrence Frequencies

- By Zipf, a word appearing n times has rank $r_n = AN/n$
- If several words may occur n times, assume rank r_n applies to the last of these.
- Therefore, r_n words occur n or more times and r_{n+1} words occur $n+1$ or more times.
- So, the number of words appearing **exactly** n times is:

$$I_n = r_n - r_{n+1} = \frac{AN}{n} - \frac{AN}{n+1} = \frac{AN}{n(n+1)}$$

Fraction of words with frequency n is:

$$\frac{I_n}{D} = \frac{1}{n(n+1)}$$

Fraction of words appearing only once is therefore $1/2$.

Zipf's Law Impact on Language Analysis

- **Good News:** Stopwords will account for a large fraction of text so eliminating them greatly reduces size of vocabulary in a text
- **Bad News:** For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

Train and Test Corpora

- A language model must be trained on a large corpus of text to estimate good parameter values.
- Model can be evaluated based on its ability to predict a high probability for a disjoint (held-out) test corpus (testing on the training corpus would give an optimistically biased estimate).
- Ideally, the training (and test) corpus should be representative of the actual application data.
- May need to ***adapt*** a general model to a small amount of new (***in-domain***) data by adding highly weighted small corpus to original training data.

Evaluation of Language Models

- Ideally, evaluate use of model in end application (*extrinsic, in vivo*)
 - Realistic
 - Expensive
- Evaluate on ability to model test corpus (*intrinsic*).
 - Less realistic
 - Cheaper
- Verify at least once that intrinsic evaluation correlates with an extrinsic one.

Evaluation and Data Sparsity Questions

- **Perplexity** and **entropy**: how do you ***estimate*** how well your language model fits a corpus once you're done?
- **Smoothing and Backoff** : how do you handle unseen n-grams?

Perplexity and Entropy

- Information theoretic metrics
 - Useful in measuring how well a **grammar** or **language model (LM)** models a natural language or a corpus
- **Entropy**: With 2 LMs and a corpus, which LM is the better match for the corpus? How much information is in a grammar or LM about what the next word will be? More is better!
 - For a random variable X ranging over e.g. bigrams and a probability function $p(x)$, the entropy of X is the expected negative log probability

$$H(X) = - \sum_{x=1}^{x=n} p(x) \log_2 p(x)$$

- Perplexity

- At each choice point in a grammar

- What are the average number of choices that can be made, weighted by their probabilities of occurrence?
 - I.e., Weighted average branching factor

$$PP(W) = 2^{H(W)}$$

- How much probability does a grammar or language model (LM) assign to the sentences of a corpus, compared to another LM?
 - The more information, the lower the perplexity.

Perplexity

- Measure of how well a model “fits” the test data.
- Uses the probability that the model assigns to the test corpus.
- Normalizes for the number of words in the test corpus and takes the inverse.

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

- Measures the weighted average branching factor in predicting the next word (lower is better).

Sample Perplexity Evaluation

- Models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary.
- Evaluate on a disjoint set of 1.5 million WSJ words.

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Exercise: Computing a Language Model

- Consider the following training corpus:

<s> there is a big house </s>

<s> i buy a house </s>

<s> they buy the new house </s>

- Create the bigram language model for this corpus

$$P(\text{there} | \text{<s>}) = 0.33$$

$$P(\text{is} | \text{there}) = 1$$

$$P(\text{a} | \text{is}) = 1$$

$$P(\text{big} | \text{a}) = 0.5$$

$$P(\text{house} | \text{big}) = 1$$

$$P(\text{</s>} | \text{house}) = 1$$

$$P(\text{i} | \text{<s>}) = 0.33$$

$$P(\text{buy} | \text{i}) = 1$$

$$P(\text{a} | \text{buy}) = 0.5$$

$$P(\text{house} | \text{a}) = 0.5$$

$$P(\text{they} | \text{<s>}) = 0.3$$

$$P(\text{buy} | \text{they}) = 1$$

$$P(\text{the} | \text{buy}) = 0.5$$

$$P(\text{new} | \text{the}) = 1$$

$$P(\text{house} | \text{new}) = 1$$

- Compute the probability of the new sentence

they buy a red house

Smoothing

- Since there are a combinatorial number of possible word sequences, many rare (but not impossible) combinations never occur in training, so zero is incorrectly assigned to many parameters (a.k.a. ***sparse data***).
- If a new combination occurs during testing, it is given a probability of zero and the entire sequence gets a probability of zero (i.e. infinite perplexity).
- In practice, parameters are ***smoothed*** (a.k.a. ***regularized***) to reassign some probability mass to unseen events.
 - Adding probability mass to unseen events requires removing it from seen ones (***discounting***) in order to maintain a joint distribution that sums to 1.

Unknown Words

- How to handle words in the test corpus that did not occur in the training data, i.e. ***out of vocabulary*** (OOV) words?
- Train a model that includes an explicit symbol for an unknown word (<UNK>).
 - Choose a vocabulary in advance and replace other words in the training corpus with <UNK>.
 - Replace the first occurrence of each word in the training data with <UNK>.

Laplace (Add-One) Smoothing

- “Hallucinate” additional training data in which each possible N-gram occurs exactly once and adjust estimates accordingly.

$$\textbf{Bigram:} \quad P(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

$$\textbf{N-gram:} \quad P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n) + 1}{C(w_{n-N+1}^{n-1}) + V}$$

where V is the total number of possible $(N-1)$ -grams (i.e. the vocabulary size for a bigram model).

- Tends to reassign too much mass to unseen events, so can be adjusted to add $0 < \delta < 1$ (normalized by δV instead of V).

Other smoothing methods

- Absolute discounting:
 - Subtract a constant from all counts and redistribute this to unseen ones using N-1 gram probs and back-off (normalization) weights
- Witten-Bell smoothing
 - Use the count of things seen once to help to estimate the count of unseen things
- Good Turing
 - Estimate the rare n-grams based on counts of more frequent counts

Other smoothing methods

- Kneser-Ney smoothing (best results so far)
 - Instead of the number of occurrences, weigh the back-offs by the number of contexts the word appears in
- Interpolation
 - Instead of only back-off cases, interpolate all N-gram counts with N-1 counts

Exercise: Computing a Language Model

- Consider the following training corpus:

there is a big house

i buy a house

they buy the new house

- Create the bigram language model for this corpus

- Compute the probability of the new sentence

they buy a red house

Exercise: Computing a Language Model + Smoothing (Laplace)

- Consider the following training corpus:

<s> there is a big house </s>

<s> i buy a house </s>

<s> they buy the new house </s>

- Create the bigram language model for this corpus

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

$$P(\text{there} | <s>) = \frac{C(<s>, \text{there}) + 1}{C(<s>) + 10} = \frac{2}{13} = 0,15$$

- Compute the probability of the new sentence

they buy a red house

$$P(\text{red} | a) = ?$$

$$P(\text{house} | \text{red}) = ?$$

Exercise: Computing a Language Model + Smoothing

- Consider the following training corpus:

<s> there is a big house </s>

<s> i buy a house </s>

<s> they buy the new house </s>

- Create the bigram language model for this corpus

$$P(\text{they} | < s >) = 0,15$$

$$P(\text{buy} | \text{they}) = 0,18$$

$$P(a | \text{buy}) = 0,16$$

$$P(</s> | \text{house}) = 0,30$$

$$P(\text{red} | a) = 0,08$$

$$P(\text{house} | \text{red}) = 0,1$$

- Compute the probability of the new sentence

*they buy a **red** house*

$$P(\text{they buy a red house}) = 0.15 * 0.18 * 0,16 * 0,08 * 0,01 * 0,3 = 0,0000010$$

Model Combination

- As N increases, the power (expressiveness) of an N -gram model increases, **but** the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).
- A general approach is to combine the results of multiple N -gram models of increasing complexity (i.e. increasing N).

Other tokenization method – character-based

- Split text at character level, not into words;
- ASCII/UNICODE codes can be used to convert text to numerical values;
- (+) Does not suffer from OOV (out of vocabulary) problem, since the same chars are used to represent all words, known or not;
- (-) The resulted vectors are very large (too many computations)
- (-) No generalization: words like *happy* or *happiness* or *unhappiness* have a common root *happ*. Character encodings do not capture this characteristic.

Byte Pair Encoding (BPE)

- Breaks words into relevant sub-components

Word	SubUnit	SubUnit	SubUnit	Sub-unit
happy		happ	y	
unhappy	un	happ	y	
happiness		happ	i	ness
unhappiness	un	happ	i	ness

BPE Intuition

1. Initialize the vocabulary V with individual characters from training.
2. Define the number of desired merge operations (M).
3. For i from 1 to M :
 - a. Create list of all character pairs with their frequency from training
 - b. Identify the most frequent character pair ($p1$, $p2$).
 - c. Merge $p1$ and $p2$ into a new subword unit.
 - d. Update the vocabulary V by adding the merged subword unit and removing $p1$ and $p2$.
4. End the loop when the specified number of merge operations is reached or when no more pairs can be merged.
5. The final vocabulary V contains subword units that represent common character sequences in the training data.

BPE example*



Seq	Count
"l" followed by "o"	3
"o" followed by "w"	3
"w" followed by " "	2
" " followed by "n"	2
"n" followed by "e"	2
"e" followed by "w"	2

Seq	Count
"w" followed by " "	2
" " followed by "l"	2
"l" followed by "o"	2
"o" followed by "w"	2
"w" followed by "e"	3
"e" followed by "s"	3

Seq	Count
"s" followed by "t"	3
"t" followed by " "	3
" " followed by "w"	1
"w" followed by "i"	1
"i" followed by "d"	1
..... and so on	

* example from *Low Level Large Language Models - a Deep Dive into how Large Language Models Work*, Gavrilit Dragos, EUROLAN Summer School, 2023

BPE example

lo w  n e w  lo w e s t  n e w e s t  w i d e s t  lo w e r 


low  n e w  low e s t  n e w e s t  w i d e s t  low e r 







› Next merge will be “e” with “s” (3 appearances)

low  n e w  low e s t  n e w e s t  w i d e s t  low e r 

› Next merge will be “es” with “t” (3 appearances)

low  n e w  low e s t  n e w e s t  w i d e s t  low e r 

› Next merge will be “esr” with “” (3 appearances) **[this is a suffix]**

low  n e w  low e s t  n e w e s t  w i d e s t  low e r 

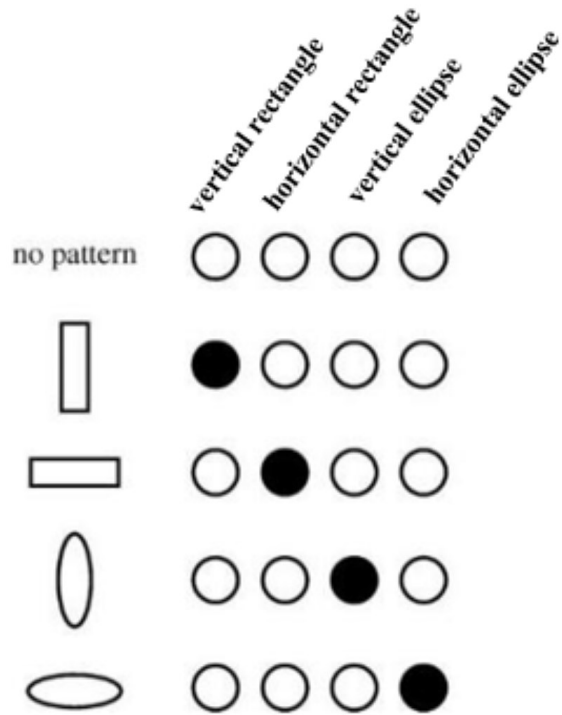
Problems of N-Grams Language Models

- Limited context
 - Usual n size is up to 4, but that yields a very limited context.
- Long dependencies are not considered
 - Syntactic dependencies
 - “The **man** next to the large oak tree near the grocery store on the corner **is** tall.”
 - “The **men** next to the large oak tree near the grocery store on the corner **are** tall.”
 - Semantic dependencies
 - “The **bird** next to the large oak tree near the grocery store on the corner **flies** rapidly.”
 - “The **man** next to the large oak tree near the grocery store on the corner **talks** rapidly.”

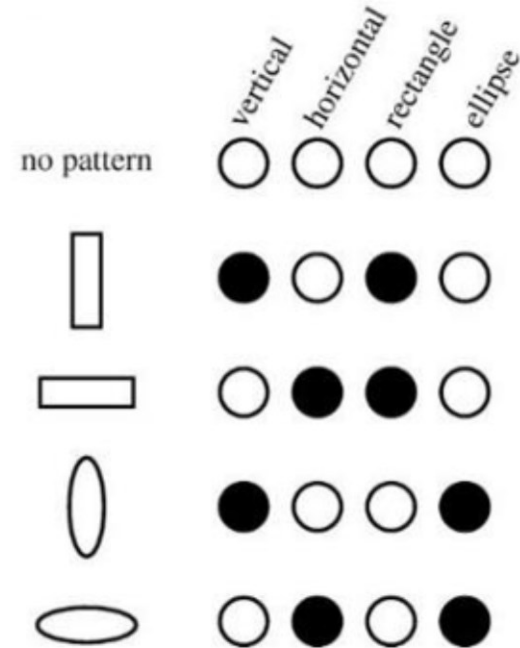
Problems of N-Grams Language Models

- No generalization to allow the discovery of **similarity** between words.
 - “The **cat** is **walking** in **the bedroom**”
 - “A **dog** was **running** in **a room**”
- almost as likely, simply because:
 - “dog” and “cat”
 - “the” and “a”
 - “room” and “bedroom”all have similar semantic and grammatical roles.

The curse of dimensionality



Sparse representation
(i.e. one hot encoding)



Dense representation
(distributed representation)

- ...or is it the blessing of dimensionality?

Example

- You're a real estate agent trying to analyze data about houses to estimate their prices

Scenario 1:

- You have 2 dimensions: m2 and location
- You can easily plot the data on a two-dimensional scatterplot, visualize trends, and make predictions based on these two features.

Example

Scenario 2: As your dataset expands, you start collecting more features to gain a better understanding of house values.

1. Number of bedrooms
2. Number of bathrooms
3. Presence of a garage
4. Presence of a swimming pool
5. Distance to the nearest school
6. Distance to the nearest grocery store
7. Distance do city center
8. Year built
9. Presence of a fireplace
10. Lot size

Example

1. Data Sparsity:

Houses with specific combinations of attributes become sparse in the space, making it challenging to find data points that are close to each other in all dimensions.

2. Interpreting Distances:

Measuring the "distance" between houses in this 10-dimensional space is no longer straightforward.

Two houses that are "close" in this space may not necessarily be similar in terms of the features that matter most to potential buyers.

Two houses with similar distances in distance to city/grocery might have vastly different lot sizes or ages.

Example

3. Increased Complexity:

Analyzing and making predictions is significantly more complex and computationally demanding.

Traditional visualization methods (like scatterplots) are no longer effective for high-dimensional data.

4. Overfitting:

With many dimensions, there's a higher risk of overfitting models because they can capture noise in the data instead of true patterns.

Sparse vs dense representation example

Feature List: {"bedrooms", "square footage", "garage", "swimming pool", "backyard"}

Property 1: 3 bedrooms, 1500 sq. ft., no garage, swimming pool, backyard.

- Sparse Representation: [1, 1, 0, 1, 1]
- Dense Representation: [3, 1500, 0, 1, 1.5]

Property 2: 4 bedrooms, 2000 sq. ft., garage, no swimming pool, backyard.

- Sparse Representation: [1, 1, 1, 0, 1]
- Dense Representation: [4, 2000, 1, 0, 0.8]

Neural language models

- Intuition:
 - sequence-to-sequence framework based on the encoder-decoder scheme
 - associate with each word in the vocabulary a **distributed word feature vector**
 - express the **joint probability** function of word sequences in terms of the feature vectors of these words in the sequence
 - **learn simultaneously** the word feature vectors and the parameters of that probability function.

Neural language models

- Works in 3 steps (step 1):
 - An encoder first maps the input sequence into fixed-sized low-dimensional feature vectors (called input embeddings)
 - The feature vector represents different aspects of the word
 - Each word is associated with a point in a vector space.
 - The number of features (e.g. $m = 30, 60$ or 100 in the experiments) is much smaller than the size of the vocabulary (usually more than 15.000 words).

Neural language models

- Works in 3 steps (step 2 and 3):
 - express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence
 - The probability function is expressed as a product of conditional probabilities of the next word given the previous ones
 - a multilayer neural network is used to predict the next word given the previous ones
 - learn both feature vectors and parameters for the probability function

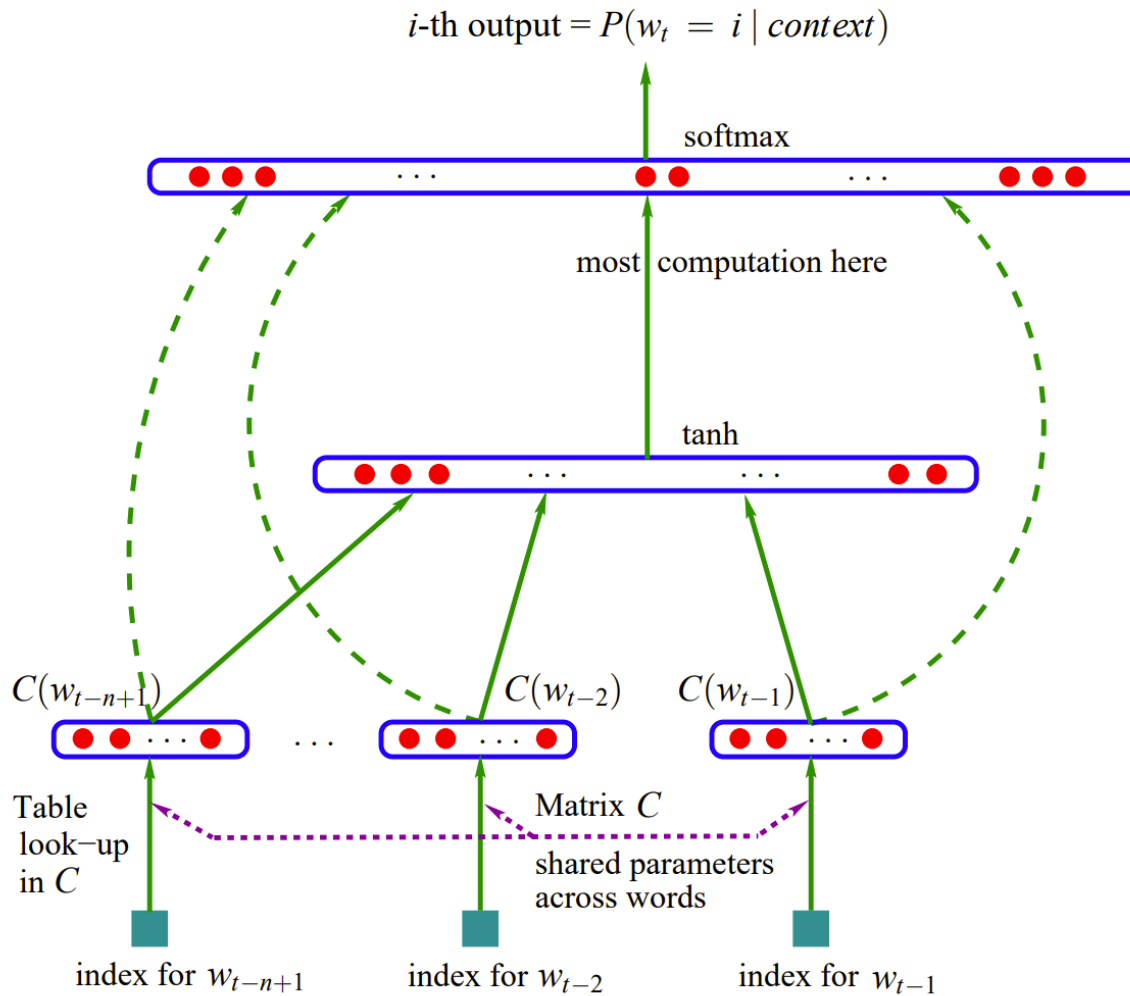
Neural language models

- The training set is a sequence $w_1 \cdots w_T$ of words $w_t \in V$, where the vocabulary V is a large but finite set.
- The objective is to learn a good model that gives high out-of-sample likelihood by employing a low-dimensional semantic representations to capture linguistic features of language (useful to alleviate data sparsity)

Neural language models

- Various neural models have been proposed for the encoder-decoder architecture:
 - graph neural networks (GNN) for encoding graph inputs
 - recurrent neural networks (RNN) for decoding texts.
- ... with different designs:
 - attention mechanism
 - copy mechanism

Neural language model



Neural language model representation

Bengio, Yoshua, et al. "A neural probabilistic language model." *The journal of machine learning research* 3 (2003): 1137-1155.

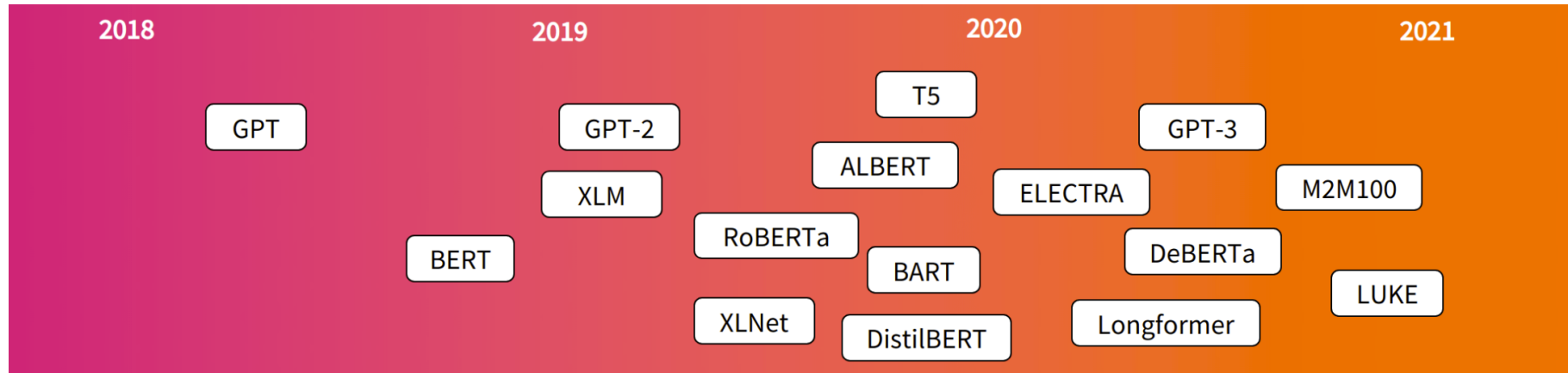
Neural language model

- A mapping C from any element i of V to a real vector $C(i) \in \mathbb{R}^m$.
- It represents the *distributed feature vectors* associated with each word in the vocabulary.
- In practice, C is represented by a $|V| \times m$ matrix of free parameters.

Neural language model

- The probability function over words, expressed with C :
 - a function g maps an input sequence of feature vectors for words in context, $C(w_{t-n+1}, \dots, C(w_{t-1}))$, to a conditional probability distribution over words in V for the next word w_t .
 - The output of g is a vector whose i -th element estimates the probability $\hat{P}(w_t = i | w_1^{t-1})$.

Large Pretrained Neural Language Models



- J.Devlin, M.-W. Chang, K. Lee and K. Toutanova. (2019). [BERT](#): Pre-training of Deep Bidirectional Transformers for Language Understanding.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever. (2019) Language models are unsupervised multitask learners. [OpenAI blog](#) (2019)
- [BLOOM](#)
- Li, J., Tang, T., Zhao, W. X., & Wen, J. R. (2021). [Pretrained language models for text generation: A survey.](#) *arXiv preprint arXiv:2105.10311*.

Summary

- Language models assign a probability that a sentence is a legal string in a language.
- They are useful as a component of many NLP systems, such as ASR, OCR, and MT.
- Simple N-gram models are easy to train on unsupervised corpora and can provide useful estimates of sentence likelihood.
- Smoothing techniques adjust parameter estimates to account for unseen (but not impossible) events.
- Neural language models are better at generalizing using learned semantic features of words, but need very large corpora.

Google NGrams

All Our N-gram are Belong to You

By Peter Norvig - 8/03/2006 11:26:00 AM

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word [n-gram models](#) for a variety of R&D projects, such as [statistical machine translation](#), speech recognition, [spelling correction](#), entity detection, information extraction, and others. While such models have usually been estimated from training to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.