A (Very Very) Brief Introduction to Language Models

Aniello De Santo
he/him

aniellodesanto.github.io
aniello.desanto@utah.edu

CUNY GC

March 27, 2023

Get the slides!

A Definition

Language Models assign probabilities to sequences of words.

A Definition

Language Models assign probabilities to sequences of words.

$$
\begin{aligned}
P(X_1 ... X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\
&= \prod_{k=1}^{n} P(X_k|X_{1:k-1})
\end{aligned}
$$

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

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

$$
\begin{aligned}
&P(\texttt{<s> i want english food </s>}) \\
&\quad = P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&\quad = .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&\quad = .000031
\end{aligned}
$$

A Definition

Language Models assign probabilities to sequences of words.

$$
\begin{aligned}
P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\
&= \prod_{k=1}^{n} P(X_k|X_{1:k-1})
\end{aligned}
$$

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

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

$$
\begin{aligned}
&P(\texttt{<s> i want english food </s>}) \\
&\quad = P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad\quad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&\quad = .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&\quad = .000031
\end{aligned}
$$

Word Prediction Everywhere

An experiment:

- Open any chat/messaging app you use frequently
- Start typing

  I wish this lecture was ____

- What do you get after **was**?
- The same idea also applies also to full sentences!

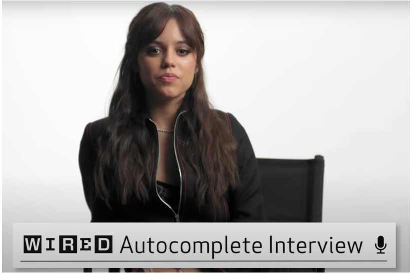
Word Prediction Everywhere

An experiment:

- Open any chat/messaging app you use frequently
- Start typing

I wish this lecture was ____

- What do you get after **was**?
- The same idea also applies also to full sentences!



Word Prediction Everywhere [cont.]

- Humans do it too!

Please turn your homework ____

Why is automatizing this useful?

- speech recognition
- spell-checking/grammatical error correction
- machine translation
- maybe even more direct linguistics research ...

This is where Language Models (LMs) enter the picture!

Word Prediction Everywhere [cont.]

- Humans do it too!

Please turn your homework ____

Why is automatizing this useful?

- speech recognition
- spell-checking/grammatical error correction
- machine translation
- maybe even more direct linguistics research . . .

This is where Language Models (LMs) enter the picture!

Word Prediction Everywhere [cont.]

- Humans do it too!

Please turn your homework ____

Why is automatizing this useful?

- speech recognition
- spell-checking/grammatical error correction
- machine translation
- maybe even more direct linguistics research ...

This is where Language Models (LMs) enter the picture!

Tackling Next Word Prediction

We want the **most likely completion(s)**...

Uhm, how do we figure out what is most likely?

- A naive idea:
  $\rightarrow$ Most likely = Most frequent word
- **Approach:**
  1. Collect sufficiently large sample of texts (**corpus**)
  2. For each word (**type**), count how often it occurs in the entire sample (= its number of **tokens**).
  3. Calculate the **frequency** of the word in the sample:

$$\text{freq}(\textit{word}, \textit{sample}) = \frac{\text{number of tokens of } \textit{word}}{\text{word length of whole } \textit{sample}}$$

Tackling Next Word Prediction

We want the **most likely completion(s)**...

Uhm, how do we figure out what is most likely?

- A naive idea:
  $\rightarrow$ Most likely = Most frequent word
- **Approach:**
  1. Collect sufficiently large sample of texts (**corpus**)
  2. For each word (**type**), count how often it occurs in the entire sample (= its number of **tokens**).
  3. Calculate the **frequency** of the word in the sample:

$$\text{freq}(\textit{word}, \textit{sample}) = \frac{\text{number of tokens of } \textit{word}}{\text{word length of whole } \textit{sample}}$$

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$$\text{freq}(\text{be}) = \frac{\mathbf{13}}{1000} = 1.3\%$$

$$\text{freq}(\text{bee}) = \frac{\mathbf{0}}{1000} = 0.0\%$$

$$\text{freq}(\text{bed}) = \frac{\mathbf{2}}{1000} = 0.2\%$$

$$\text{freq}(\text{bell}) = \frac{\mathbf{3}}{1000} = 0.3\%$$

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$$\text{freq}(\text{be}) = \frac{\mathbf{13}}{1000} = 1.3\%$$

$$\text{freq}(\text{bee}) = \frac{\mathbf{0}}{1000} = 0.0\%$$

$$\text{freq}(\text{bed}) = \frac{\mathbf{2}}{1000} = 0.2\%$$

$$\text{freq}(\text{bell}) = \frac{\mathbf{3}}{1000} = 0.3\%$$

Ordered predictions:

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$\text{freq}(\text{be}) = \frac{\mathbf{13}}{1000} = 1.3\%$

$\text{freq}(\text{bed}) = \frac{\mathbf{2}}{1000} = 0.2\%$

$\text{freq}(\text{bee}) = \frac{\mathbf{0}}{1000} = 0.0\%$

$\text{freq}(\text{bell}) = \frac{\mathbf{3}}{1000} = 0.3\%$

Ordered predictions: be

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$\text{freq(be)} = \frac{\mathbf{13}}{1000} = 1.3\%$

$\text{freq(bee)} = \frac{\mathbf{0}}{1000} = 0.0\%$

$\text{freq(bed)} = \frac{\mathbf{2}}{1000} = 0.2\%$

$\text{freq(bell)} = \frac{\mathbf{3}}{1000} = 0.3\%$

Ordered predictions: be, bell

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$$\text{freq(be)} = \frac{\mathbf{13}}{1000} = 1.3\% \qquad \text{freq(bee)} = \frac{\mathbf{0}}{1000} = 0.0\%$$

$$\text{freq(bed)} = \frac{\mathbf{2}}{1000} = 0.2\% \qquad \text{freq(bell)} = \frac{\mathbf{3}}{1000} = 0.3\%$$

Ordered predictions: be, bell, bed

Example calculation

Sample: 1000 words long

Words: be, bed, bee, bell

Type	be	bed	bee	bell
Tokens	**13**	**2**	**0**	**3**

$$\text{freq(be)} = \frac{\mathbf{13}}{1000} = 1.3\% \qquad \text{freq(bee)} = \frac{\mathbf{0}}{1000} = 0.0\%$$

$$\text{freq(bed)} = \frac{\mathbf{2}}{1000} = 0.2\% \qquad \text{freq(bell)} = \frac{\mathbf{3}}{1000} = 0.3\%$$

Ordered predictions: be, bell, bed, bee

Tackling Next Word Prediction [cont.]

We want the **most likely completion(s)**...

- Most likely = Most frequent word?
- BUT! Word usage varies by **context**!

Example

	tested	**testing**	**testimony**
I have			
I have been			
I have the			

- The frequency of words is not enough,
  we need frequencies of sequences of words ⇒ **n-gram LMs**

Tackling Next Word Prediction [cont.]

We want the **most likely completion(s)**...

- Most likely = Most frequent word?
- BUT! Word usage varies by **context**!

Example

	tested	**testing**	**testimony**
I have	hi	low	mid
I have been			
I have the			

- The frequency of words is not enough,
  we need frequencies of sequences of words ⇒ **n-gram LMs**

Tackling Next Word Prediction [cont.]

We want the **most likely completion(s)**...

- Most likely = Most frequent word?
- BUT! Word usage varies by **context**!

Example

	tested	**testing**	**testimony**
I have	hi	low	mid
I have been	hi	hi	low
I have the	low	low	hi

- The frequency of words is not enough,
  we need frequencies of sequences of words ⇒ **n-gram LMs**

Tackling Next Word Prediction [cont.]

We want the **most likely completion(s)**...

- Most likely = Most frequent word?
- BUT! Word usage varies by **context**!

Example

	tested	**testing**	**testimony**
I have	hi	low	mid
I have been	hi	hi	low
I have the	low	low	hi

- The frequency of words is not enough,
  we need frequencies of sequences of words ⇒ **n-gram LMs**

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and $\boxed{\text{Marie are}}$ not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are $\boxed{\text{not Bill}}$ and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not $\boxed{\text{Bill and}}$ Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill $\boxed{\text{and Sue}}$

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

$\boxed{\text{John and Marie}}$ are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John $\boxed{\text{and Marie are}}$ not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are $\boxed{\text{not Bill and}}$ Sue

Defining n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not $\boxed{\text{Bill and Sue}}$

N-Gram LMs for Next Word Prediction

Frequencies can be computed and used for n-grams, too.
→ we still need a representative corpus...

Example

- **Trigram frequencies**

bus is late	30%	train is late	15%
bus is lovely	25%	train is lovely	8%
bus is lazy	10%	train is lazy	2%

- **Input**
  I will text you if the train is __
- **Sorted completions**

- To predict a word w:
  1. **Needed resources:** corpus
  2. Compute frequencies for all n-grams
  3. Look at previous $n-1$ words
  4. Find completions that maximize n-gram probability

N-Gram LMs for Next Word Prediction

Frequencies can be computed and used for n-grams, too.
$\rightarrow$ we still need a representative corpus...

Example

- **Trigram frequencies**

bus is late	30%	train is late	15%
bus is lovely	25%	train is lovely	8%
bus is lazy	10%	train is lazy	2%

- **Input**
  I will text you if the train is __
- **Sorted completions**
  late

- To predict a word w:
  1. **Needed resources:** corpus
  2. Compute frequencies for all n-grams
  3. Look at previous $n-1$ words
  4. Find completions that maximize n-gram probability

N-Gram LMs for Next Word Prediction

Frequencies can be computed and used for n-grams, too.
→ we still need a representative corpus...

Example

- **Trigram frequencies**

bus is late	30%	train is late	15%
bus is lovely	25%	train is lovely	8%
bus is lazy	10%	train is lazy	2%

- **Input**
  I will text you if the train is __

- **Sorted completions**
  late, lovely

- To predict a word w:
  1. **Needed resources:** corpus
  2. Compute frequencies for all n-grams
  3. Look at previous $n-1$ words
  4. Find completions that maximize n-gram probability

N-Gram LMs for Next Word Prediction

Frequencies can be computed and used for n-grams, too.
$\rightarrow$ we still need a representative corpus...

Example

- **Trigram frequencies**

bus is late	30%	train is late	15%
bus is lovely	25%	train is lovely	8%
bus is lazy	10%	train is lazy	2%

- **Input**
  I will text you if the train is __
- **Sorted completions**
  late, lovely, lazy

- To predict a word w:
  1. **Needed resources:** corpus
  2. Compute frequencies for all n-grams
  3. Look at previous $n-1$ words
  4. Find completions that maximize n-gram probability

Linguistic Evaluation

The n-Gram Hypothesis (aka Markov Assumption)

The **preceding $n-1$ words** reliably predict the next word.

$$P(w_n|w_1w_2\dots w_{n-2}w_{n-1}) \approx P(w_n|w_{n-2}w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

- The n-gram hypothesis is **not quite satisfying**, though.
  1. We are not going to see all possible words in all contexts
  2. Many dependencies in language are not local

Linguistic Evaluation

The n-Gram Hypothesis (aka Markov Assumption)

The **preceding $n-1$ words** reliably predict the next word.

$$P(w_n|w_1w_2\dots w_{n-2}w_{n-1}) \approx P(w_n|w_{n-2}w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

- The n-gram hypothesis is **not quite satisfying**, though.
  1. We are not going to see all possible words in all contexts
  2. Many dependencies in language are not local

The n-Gram Hypothesis (aka Markov Assumption)

The **preceding** $n-1$ **words** reliably predict the next word.

$$P(w_n|w_1 w_2 \dots w_{n-2} w_{n-1}) \approx P(w_n|w_{n-2} w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

- The n-gram hypothesis is **not quite satisfying**, though.
  1. We are not going to see all possible words in all contexts
  2. Many dependencies in language are not local

Linguistic Evaluation

The n-Gram Hypothesis (aka Markov Assumption)

The **preceding $n-1$ words** reliably predict the next word.

$$P(w_n|w_1 w_2 \dots w_{n-2} w_{n-1}) \approx P(w_n|w_{n-2} w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

- The n-gram hypothesis is **not quite satisfying**, though.
  1. We are not going to see all possible words in all contexts
  2. Many dependencies in language are not local

This is where Neural Networks LMs come in handy...

Ok but, who cares?

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction
- Spoilers: **Neural LMs**...

And?

- speech recognition
- spell-checking/grammatical error correction
- text generation (think chatbots)
- machine translation
- maybe even (less application-oriented) linguistic research . . .

Ok but, who cares?

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction
- Spoilers: **Neural LMs**...

And?

- speech recognition
- spell-checking/grammatical error correction
- text generation (think chatbots)
- machine translation
- maybe even (less application-oriented) linguistic research ...

Ok but, who cares?

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction
- Spoilers: **Neural LMs**...

And?

- speech recognition
- spell-checking/grammatical error correction
- text generation (think chatbots)
- machine translation
- maybe even (less application-oriented) linguistic research ...

LMs as Tools for Psycholinguistics

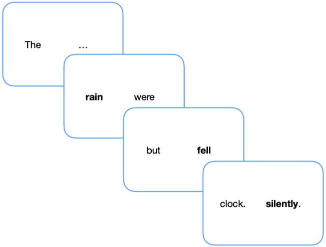
Jacobs, De Santo, and Grobol (2023)

Zeugma	The architect bit the lime and **the dust**
Literal	The architect bit the lime and the apple

- We can use LMs to generate literal continuations

  The architect bit the ____

- Maze Task (Boyce & Levy, 2021):
  Use LMs to generate low probability foils



LMs as Tools for Psycholinguistics

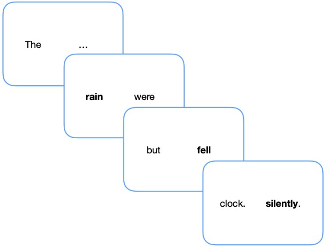
Jacobs, De Santo, and Grobol (2023)

Zeugma	The architect bit the lime and **the dust**
Literal	The architect bit the lime and the apple

- We can use LMs to generate literal continuations

  The architect bit the ____

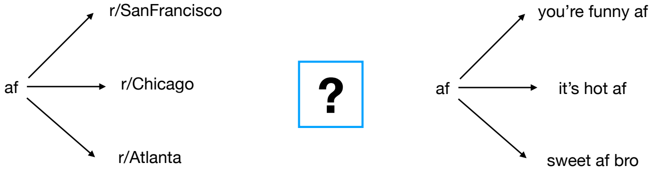
- Maze Task (Boyce & Levy, 2021):
  Use LMs to generate low probability foils



LMs as Tools for Sociolinguistics

Making "fetch" happen: The influence of social and linguistic context on nonstandard word growth and decline

Ian Stewart and Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology



- Does the social context of a word influence its adoption more than its linguistic context?
- Use *unique* n-gram counts to measure *dissemination*: the diversity of linguistic contexts in which a word appears
- How do communities (e.g. *r/x,y,z*) predict word usage? (Lucy & Bamman, 2021)

LMs as Psycholinguistic Subjects

"Wait...Maybe I find **the models** interesting?"

- Can we use linguistic tests to understand them better?

LMs as Psycholinguistic Subjects

"Wait...Maybe I find **the models** interesting?"

- Can we use linguistic tests to understand them better?

Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies

Tal Linzen[1,2] **Emmanuel Dupoux**[1]
LSCP[1] & IJN[2], CNRS,
EHESS and ENS, PSL Research University
{tal.linzen,
emmanuel.dupoux}@ens.fr

Yoav Goldberg
Computer Science Department
Bar Ilan University
yoav.goldberg@gmail.com

Agreement attraction errors

Training objective	Sample input	Training signal	Prediction task	Correct answer
Number prediction	*The keys to the cabinet*	PLURAL	SINGULAR/PLURAL?	PLURAL
Verb inflection	*The keys to the cabinet [is/are]*	PLURAL	SINGULAR/PLURAL?	PLURAL
Grammaticality	*The keys to the cabinet are here.*	GRAMMATICAL	GRAMMATICAL/UNGRAMMATICAL?	GRAMMATICAL
Language model	*The keys to the cabinet*	are	$P(are) > P(is)$?	True

Table 1: Examples of the four training objectives and corresponding prediction tasks.

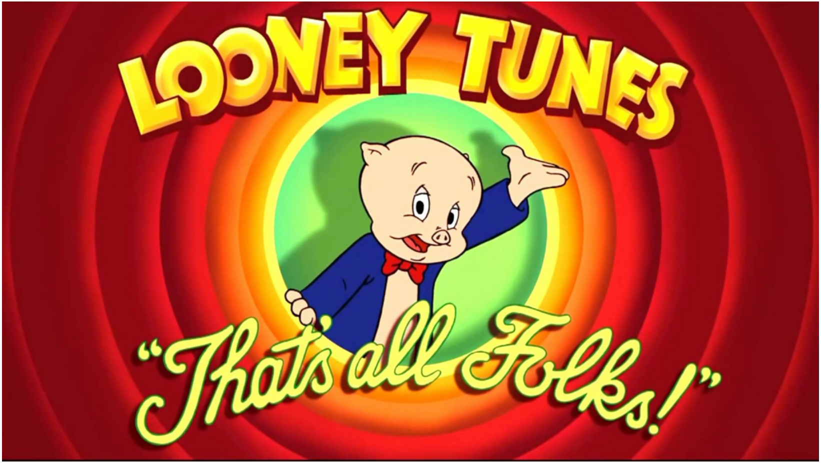
A Final Note: A Word of Caution

- LMs are sensitive to statistical regularities in language data...
  - Bias: treating language behavior as ground truth (Bolukbasi et al. 2016)
  - Exclusion/discrimination: what kind of data is included? (Bender et al. 2019)
  - Privacy: whose data and how do we get it? (Huang & Paul 2019)
  - Environmental and financial cost (Strubell et al. 2019)
  - And more!
- Reflect on **social impact** while conducting research!

A Final Note: A Word of Caution

- LMs are sensitive to statistical regularities in language data...
  - Bias: treating language behavior as ground truth (Bolukbasi et al. 2016)
  - Exclusion/discrimination: what kind of data is included? (Bender et al. 2019)
  - Privacy: whose data and how do we get it? (Huang & Paul 2019)
  - Environmental and financial cost (Strubell et al. 2019)
  - And more!
- Reflect on **social impact** while conducting research!

The End (?)



Appendix

N-Grams Limits

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction.
  - Struggle to generalize to novel contexts
  - Struggle with long distance relations (Markov assumption)
- Spoilers: **Neural LMs**...
  - ...might help with these issues
    - Incorporate word similarity based on distributional information
    - More complex approximation of sentential dependencies

N-Grams Limits

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction.
  - Struggle to generalize to novel contexts
  - Struggle with long distance relations (Markov assumption)
- Spoilers: **Neural LMs**...
  - ...might help with these issues
    - Incorporate word similarity based on distributional information
    - More complex approximation of sentential dependencies

N-Grams Limits

LMs assign probabilities to sequences of words.

- **n-Gram LMs**: use local contexts for sequence prediction.
  - Struggle to generalize to novel contexts
  - Struggle with long distance relations (Markov assumption)
- Spoilers: **Neural LMs**...
  - ...might help with these issues
    - Incorporate word similarity based on distributional information
    - More complex approximation of sentential dependencies

Generalizing to Novel Contexts

Imagine our model has seen sequences like:

I have to make sure that the cat gets fed.
Pearl's parrot gets fed every day.

Then we want to complete the following:

I forgot to make sure that the dog gets ____

- It would be great if the model could take advantage of the similarity between *dog*,*cat*,*parrot* to predict *fed*!

Generalizing to Novel Contexts

Imagine our model has seen sequences like:

I have to make sure that the cat gets fed.
Pearl's parrot gets fed every day.

Then we want to complete the following:

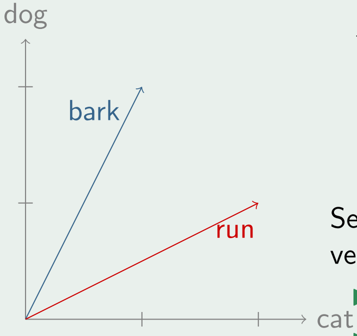
I forgot to make sure that the dog gets ____

- It would be great if the model could take advantage of the similarity between *dog*,*cat*,*parrot* to predict *fed*!

From Counts to Vector Spaces

The dog barked at the cat. The cat ran away. The dog ran after the cat. The dog kept barking. He also kept running.

2-Dimensional Vector Space with *dog* and *cat*



	dog	cat	bark	run
dog	-	2	2	1
cat	2	-	1	2
bark	2	1	-	0
run	1	2	0	-

Semantic similarity as angle between vectors:

- *bark* more closely related to *dog*
- *run* more closely related to *cat*

Long-distance Dependencies in Language

- Word choice can be influenced by words that are very far away.

Subject-verb agreement

- The key to the cabinet **is** on the table.
- The keys to the cabinets **are** on the table.
- The key to the cabinets **is**/**are** on the table.
- The keys to the cabinet **is**/**are** on the table.

- Observation: humans get those "wrong" sometimes...
- It's not just about complex "syntactic" dependencies

  I spread like strawberries, I climb like peas and beans

  I've been sucking it in so long, That I'm busting at the seams

Long-distance Dependencies in Language

- Word choice can be influenced by words that are very far away.

Subject-verb agreement

- The key to the cabinet **is** on the table.
- The keys to the cabinets **are** on the table.
- The key to the cabinets **is**/**are** on the table.
- The keys to the cabinet **is**/**are** on the table.

- Observation: humans get those “wrong” sometimes...
- It’s not just about complex “syntactic” dependencies

  I spread like strawberries, I climb like peas and beans
  I’ve been sucking it in so long, That I’m busting at the seams

Long-distance Dependencies in Language

- Word choice can be influenced by words that are very far away.

Subject-verb agreement

- The key to the cabinet **is** on the table.
- The keys to the cabinets **are** on the table.
- The key to the cabinets **is**/**are** on the table.
- The keys to the cabinet **is**/**are** on the table.

- Observation: humans get those "wrong" sometimes...
- It's not just about complex "syntactic" dependencies

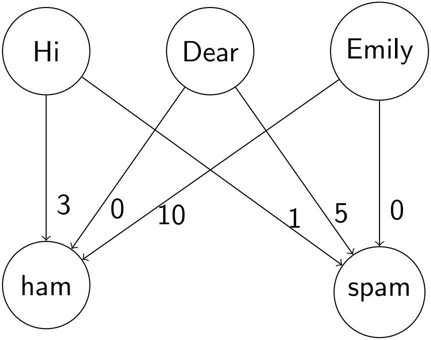
  I spread like strawberries, I climb like peas and beans

  I've been sucking it in so long, That I'm busting at the seams

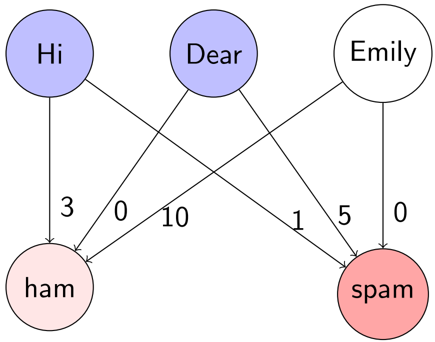
A Quick Excursus: The Perceptron

The Perceptron: A Mini-Version of a Neural Network

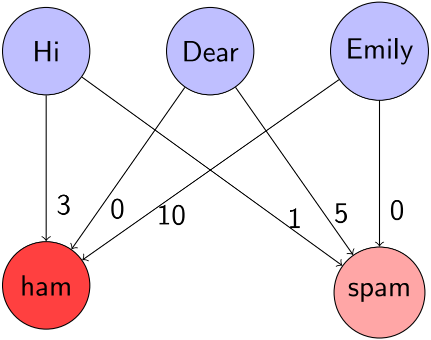
- **input layer:** neurons that are sensitive to input
- **output layer:** neurons that represent output values
- **connections:** weighted links between input and output layer
- most activated output neuron represents decision



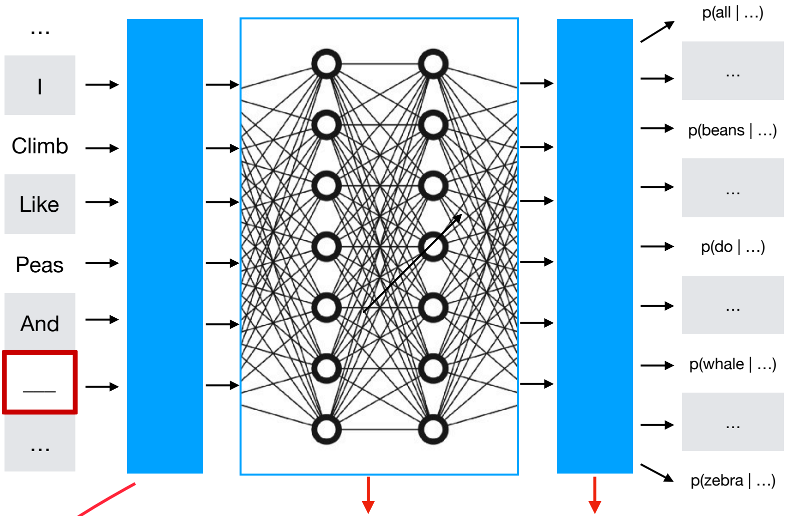
Perceptron Activation for *Hi Dear*



Perceptron Activation for *Hi Dear Emily*



Putting Things Together: Neural LMs



Embeddings

The input is now not words but their distributional representations

Hidden Layer

Derives approximations from more complex "contexts"

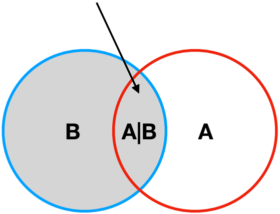
Output Layer

This makes probability predictions about upcoming word(s)

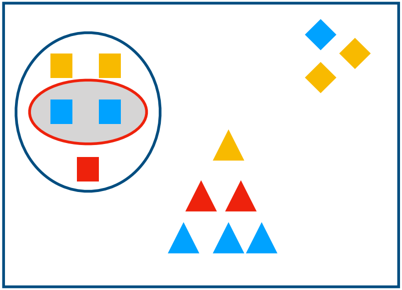
A Bit More on Conditional Probability

- We said we are interested in $P(\textbf{late}|\text{is})$

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



- E.g. $P(\textbf{blue}|\blacksquare) = 2/5$



Estimating Bigram Probabilities: MLE

Ok but where do we get probabilities from?

- One possibility: Counts (Maximum Likelihood Estimate)!
  - For a unigram:

$$P(w_n) = \frac{count(w_n)}{\sum_{w \in V} count(w)}$$

- MLE of conditional probability for bigrams:

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

- Note that the normalization factor is different than what we did for pure bigram frequency counts (which gave us an estimate of joint probability for each bigram)!

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  $\boxed{\text{when buffalo}}$ buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**

  when buffallo,

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**

  when buffallo, buffalo buffalo,

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**
  when buffallo, buffalo buffalo, buffalo buffalo,

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**
  when buffallo, buffalo buffalo, buffalo buffalo, buffalo buffalo,

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**
  when buffallo, buffalo buffalo, buffalo buffalo, buffalo buffalo, buffalo buffalo,

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo $\boxed{\text{buffalo buffalo}}$

- **Bigram token list**

  when buffallo, buffalo buffalo, buffalo buffalo, buffalo buffalo, buffalo buffalo, buffalo buffalo

- **Bigram counts and frequencies**

Frequencies for n-grams

Frequencies can be computed for n-grams, too.

Example: Calculating Bigram Frequencies

- **String**

  when buffalo buffalo buffalo buffalo buffalo buffalo

- **Bigram token list**
  when buffallo, buffalo buffalo, buffalo buffalo, buffalo buffalo, buffalo buffalo, buffalo buffalo
- **Bigram counts and frequencies**
  1. when buffalo: $1 \Rightarrow \frac{1}{6} = 16.7\%$
  2. buffalo buffalo: $5 \Rightarrow \frac{5}{6} = 83.3\%$

Ok, NOW Some Formulas

$$P(w_n | w_1 w_2 \cdots w_{n-1})$$

$P(\textbf{late} | \textit{I will text you if the train is})$ $P(\textbf{lazy} | \textit{I will text you if the train is})$

- Lots of possible sentences!
- Simplifying assumption:

$$P(w_n | w_1 w_2 \ldots w_{n-2} w_{n-1}) \approx P(w_n | w_{n-2} w_{n-1})$$

$$P(\textbf{late} | \textit{I will text you if the train is}) \approx P(\textbf{late} | \textit{train is})$$

The n-Gram Hypothesis (aka Markov Assumption)

One can reliably predict the next word based on the **preceding** $n-1$ **words**.

Ok, NOW Some Formulas

$$P(w_n|w_1w_2\cdots w_{n-1})$$

$P(\textbf{late}|\textit{I will text you if the train is})$ $P(\textbf{lazy}|\textit{I will text you if the train is})$

- Lots of possible sentences!
- Simplifying assumption:

$$P(w_n|w_1w_2\ldots w_{n-2}w_{n-1}) \approx P(w_n|w_{n-2}w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

The n-Gram Hypothesis (aka Markov Assumption)

One can reliably predict the next word based on the **preceding** $n-1$ **words**.

Ok, NOW Some Formulas

$$P(w_n|w_1 w_2 \cdots w_{n-1})$$

$P($**late**$|$*I will text you if the train is*$)$ $P($**lazy**$|$*I will text you if the train is*$)$

- Lots of possible sentences!
- Simplifying assumption:

$$P(w_n|w_1 w_2 \ldots w_{n-2} w_{n-1}) \approx P(w_n|w_{n-2} w_{n-1})$$

$P($**late**$|$*I will text you if the train is*$) \approx P($**late**$|$*train is*$)$

The n-Gram Hypothesis (aka Markov Assumption)

One can reliably predict the next word based on the **preceding** $n-1$ **words**.

Ok, NOW Some Formulas

$$P(w_n | w_1 w_2 \cdots w_{n-1})$$

$P(\textbf{late}|\textit{I will text you if the train is})$ $P(\textbf{lazy}|\textit{I will text you if the train is})$

- Lots of possible sentences!
- Simplifying assumption:

$$P(w_n | w_1 w_2 \ldots w_{n-2} w_{n-1}) \approx P(w_n | w_{n-2} w_{n-1})$$

$$P(\textbf{late}|\textit{I will text you if the train is}) \approx P(\textbf{late}|\textit{train is})$$

The n-Gram Hypothesis (aka Markov Assumption)

One can reliably predict the next word based on the **preceding $n-1$ words**.

An Observation on Frequencies: Zipf's Law

- Word models care about word frequency.
- But there is a problem...

Zipf's Law

The frequency of a type is inversely proportional to its rank.

In Plain English

The most frequent word is

- 2 times as common as the 2nd most frequent word,
- 3 times as common as the 3rd most frequent word,
- and so on.

An Observation on Frequencies: Zipf's Law

- Word models care about word frequency.
- But there is a problem. . .

Zipf's Law

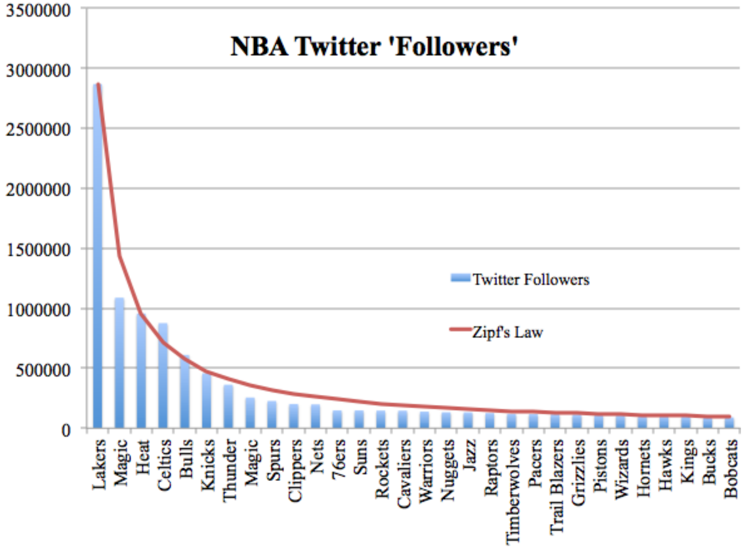
The frequency of a type is inversely proportional to its rank.

In Plain English

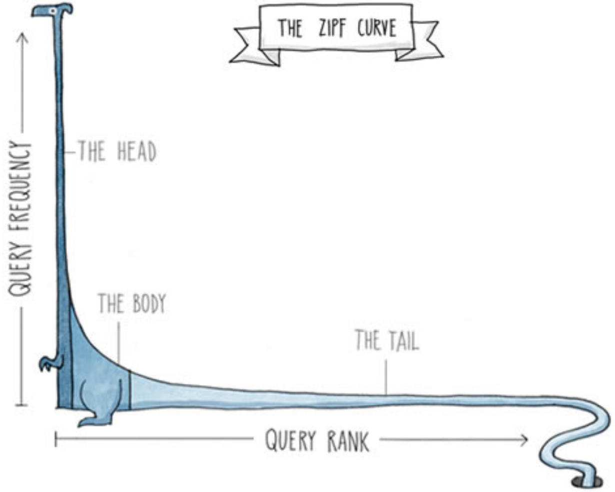
The most frequent word is

- 2 times as common as the 2nd most frequent word,
- 3 times as common as the 3rd most frequent word,
- and so on.

An Example from... the NBA?



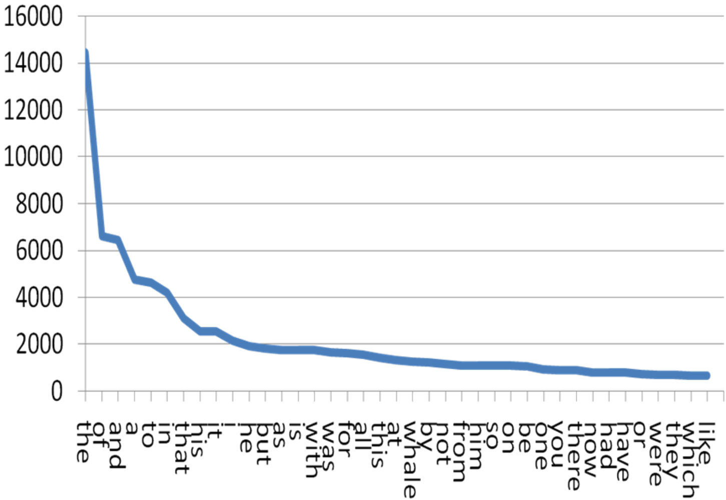
Visualizing Zipf Distributions



Zipf's Law is Everywhere...

- A distribution is probably Zipfian if
  - there is a **long neck**:
    a few types make up the majority of tokens,
  - there is a **long tail**:
    most types only have 1 token (**hapax legomenon**)
- Surprisingly, Zipf's Law shows up in tons of places:
  - size of large cities in a country
  - citations for academic papers
  - frequencies of last names
  - frequencies of weekdays in text

... Even in Language!



An Important Consequence of Zipf's Law

- Texts mostly consist of stop words.
- Hence it can be difficult to get representative counts for non-stop words.

Sparse Data Problem

- Most of the data is not informative.
- You need tons of data to have enough useful data.

An Important Consequence of Zipf's Law

- Texts mostly consist of stop words.
- Hence it can be difficult to get representative counts for non-stop words.

Sparse Data Problem

- Most of the data is not informative.
- You need tons of data to have enough useful data.

Example

- Most models require corpora with at least a few million sentences.
- Really good models (e.g. Google translate) use billions of data points.

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

$\boxed{\text{John and}}$ Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John $\boxed{\text{and Marie}}$ are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and $\boxed{\text{Marie are}}$ not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie $\boxed{\text{are not}}$ Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are $\boxed{\text{not Bill}}$ and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill $\boxed{\text{and Sue}}$

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

$\boxed{\text{John and Marie}}$ are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John $\boxed{\text{and Marie are}}$ not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are $\boxed{\text{not Bill and}}$ Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not $\boxed{\text{Bill and Sue}}$

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are not Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John $\boxed{\text{and Marie are not}}$ Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and $\boxed{\text{Marie are not Bill}}$ and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie $\boxed{\text{are not Bill and}}$ Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie are $\boxed{\text{not Bill and Sue}}$

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

$\boxed{\text{John and Marie are not}}$ Bill and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John $\boxed{\text{and Marie are not Bill}}$ and Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and $\boxed{\text{Marie are not Bill and}}$ Sue

Defining Larger n-Grams

n-gram a contiguous sequence of n words

n	Name	Example
1	unigram	John
2	bigram	John to
3	trigram	John to be
4	4-gram	John to be in
5	5-gram	John to be in the

Example

String

John and Marie $\boxed{\text{are not Bill and Sue}}$

How long can n-grams be?

- It is tempting to move to longer and longer n-grams in order to handle long-distance dependencies.
- But this has **two problems**:

  data sparsity: longer n-grams require too much data

  storage needs: longer n-grams require lots of storage
- Data sparsity is much more severe than storage needs.

Sparse data: A simple calculation

Words	bigrams	trigrams	5-grams	6-grams
10	100	1000	10,000	100,000
100	10,000	1,000,000	10,000,000,000	**1,000,000,000,000**
10,000	10^8	$\mathbf{10^{12}}$	10^{20}	10^{24}
25,000	6.3×10^8	1.6×10^{13}	9.7×10^{21}	2.4×10^{26}

Some comparison values

4.3×10^{17}	number of seconds since the Big Bang
5×10^{22}	number of stars in observable universe
10^{24}	milliliters of water in the Earth's oceans
8.8×10^{26}	diameter of observable universe, in meters
10^{80}	number of atoms in observable universe

- **Conclusion:** with large n, most n-grams are **never encountered** in a corpus $\Rightarrow$ frequency 0

Things get worse: A more realistic estimate

- The Linux dictionary american-english-insane has 650,000 entries.
- This makes the numbers much worse.
  Can you guess how many 5-grams there are then?

Things get worse: A more realistic estimate

- The Linux dictionary american-english-insane has 650,000 entries.
- This makes the numbers much worse.
  Can you guess how many 5-grams there are then?

$$\text{116 octillion} \approx \mathbf{10^{29}}$$

Things get worse: A more realistic estimate

- The Linux dictionary american-english-insane has 650,000 entries.
- This makes the numbers much worse.
  Can you guess how many 5-grams there are then?

$$116 \text{ octillion} \approx \mathbf{10^{29}}$$

10^{29} is larger than the number of shotglasses it takes to drain the Earth's oceans over 2000 times.

Evaluating Language Models: Perplexity

The **perplexity** of a language model is defined as the inverse of the probability of the test set, normalized by the number of tokens (N) in the test set.

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

A LM with lower perplexity is better because it assigns a higher probability to the unseen test corpus. But note that two LMs can be compared wrt to perplexity iff they use the same vocabulary!

- Trigram models have lower perplexity than bigram models, etc.

Intrinsic vs. Extrinsic Evaluation

Perplexity tells us which LM assigns a higher probability to unseen text.

This doesn't necessarily tell us which LM is better for a specific task.

Task-based evaluation:

- Train model A, plug it into your system for performing task T
- Evaluate performance of system A on task T
- Train model B, plug it in, evaluate system B on same task T
- Compare scores of system A and system B on task T.

Extrinsic Evaluation: Word Error rate [1]

Originally developed for speech recognition.

How much does the *predicted* sequence of words differ from the *actual* sequence of words in the correct transcript?

$$\mathrm{WER} = \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Actual words in transcript}}$$

Insertions: "eat lunch" → "eat **a** lunch"
Deletions: "see **a** movie" → "see movie"
Substitutions: "drink **ice** tea" → "drink **nice** tea"

[1] slide adapted from J. Hockenmaier