Lecture 22: Language Models

NLP Lectures: Part 1 of 4

Harvard IACS

CS109B

Pavlos Protopapas, Mark Glickman, and Chris Tanner

FOREWORD

I'll teach an NLP course next year!

The goals of the next four NLP lectures are to:

- convey the ubiquity and importance of text data/NLP
- build a foundation of the most important concepts
- illustrate how some state-of-the-art models (SOTA) work
- provide experience with these SOTA models (e.g., BERT, GPT-2)
- instill when to use which models, based on your data
- provide an overview and platform from which to dive deeper

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Our digital world is inundated with text.

How can we leverage it for useful tasks?

62B pages

500M tweets/day

360M user pages

13M articles

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

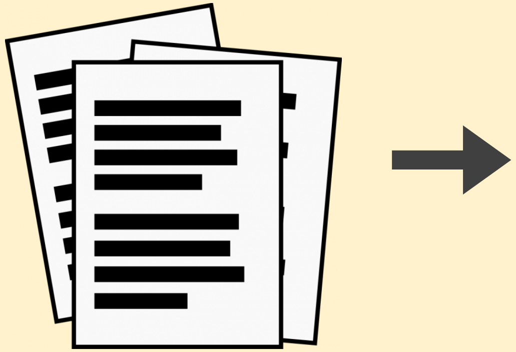
Semantics

Sentiment Analysis

Topic Modelling

Question Answering

Language Modelling



"Overall, Pfizer's COVID-19 vaccine is very safe and one of the most effective vaccines ever produced"

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

Constituenc

Dependenc

Discourse

Summarization

Coreference Re

Semantics

Sentiment Analysis

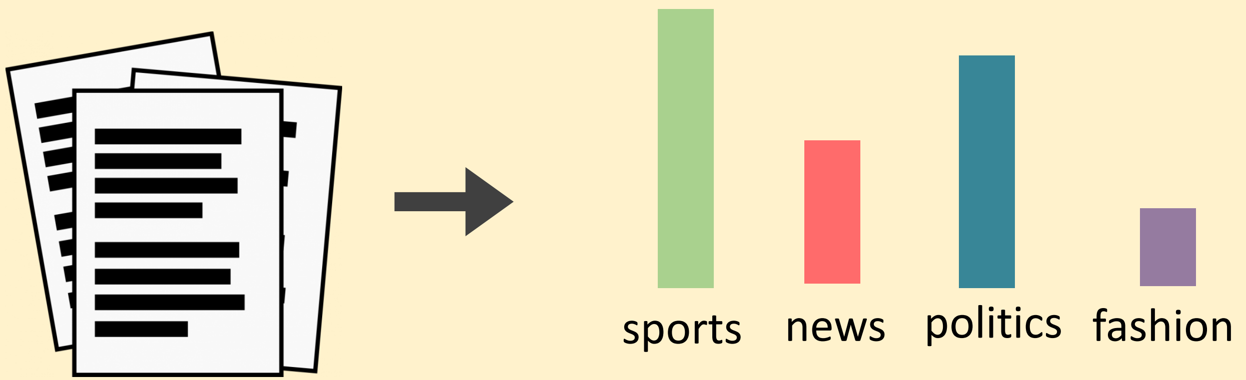
Topic Modelling

Named E y Recognition (NER)

(NLU)

G)

Language Modelling



Common NLP Tasks (aka problems)

“Alexa, play Drivers License by Olivia Rodrigo”

“Alexa, play Drivers License by Olivia Rodrigo”

INTENT SONG ARTIST

Syntax

Morpholog

Word Segm

Part-of-Spe

Parsing

Constit

Dependency

Semantics

ER)

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Discourse

Summarization

Coreference Resolution

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

ng (NLU)

Natural Lang e Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

El perro marrón → The brown dog

SPANISH → ENGLISH

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Can help with every other task!

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Language Modelling

A Language Model represents the language used by a given entity (e.g., a particular person, genre, or other well-defined class of text)

Language Modelling

A Language Model represents the language used by a given entity (e.g., a particular person, genre, or other well-defined class of text)

Spam

Not Spam

Language Modelling

A Language Model represents the language used by a given entity (e.g., a particular person, genre, or other well-defined class of text)

Language Modelling

FORMAL DEFINITION

A Language Model estimates the probability of any sequence of words

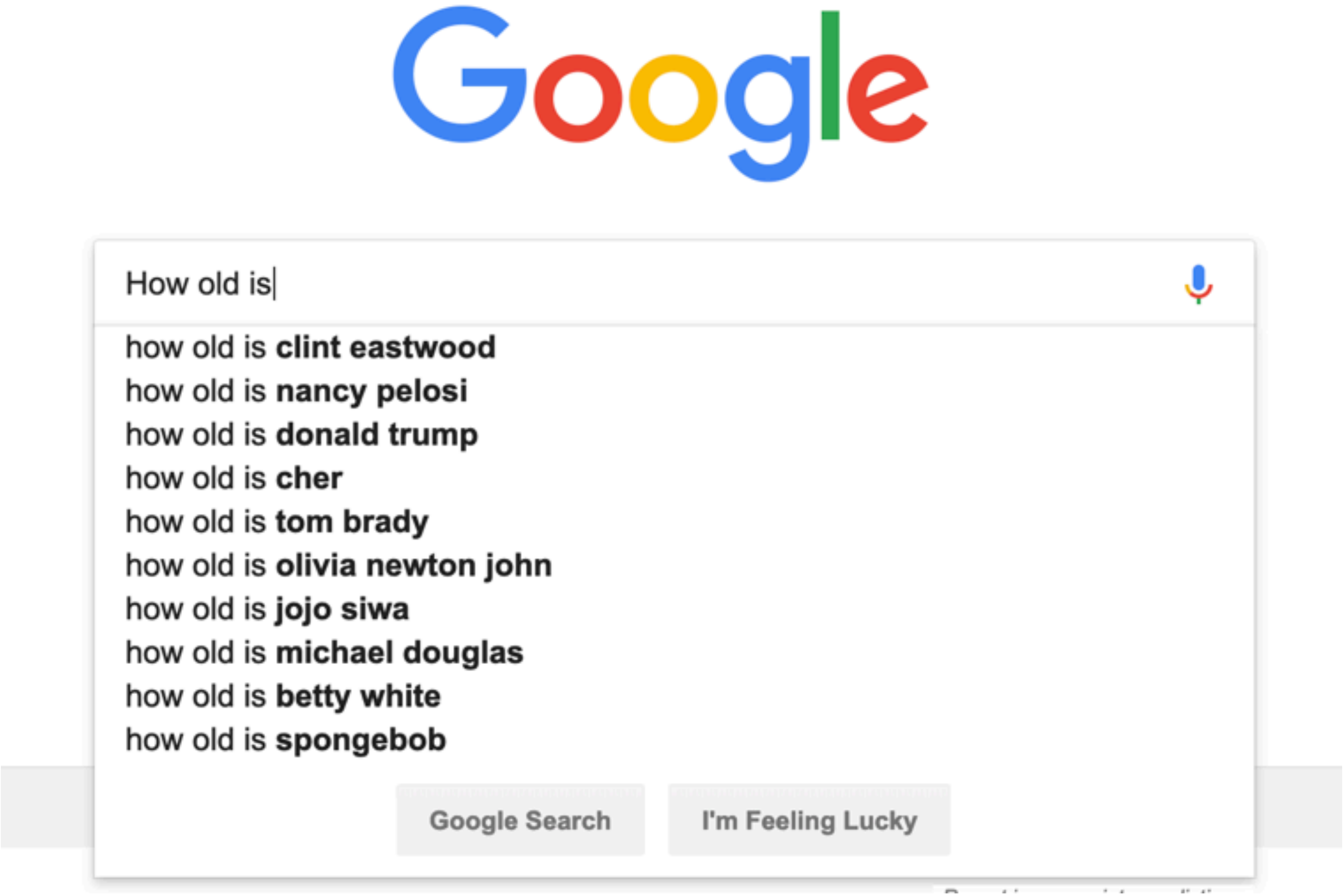
Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$\mathrm{P}(\boldsymbol{X}) = P($"Anqi was late for class"$)$

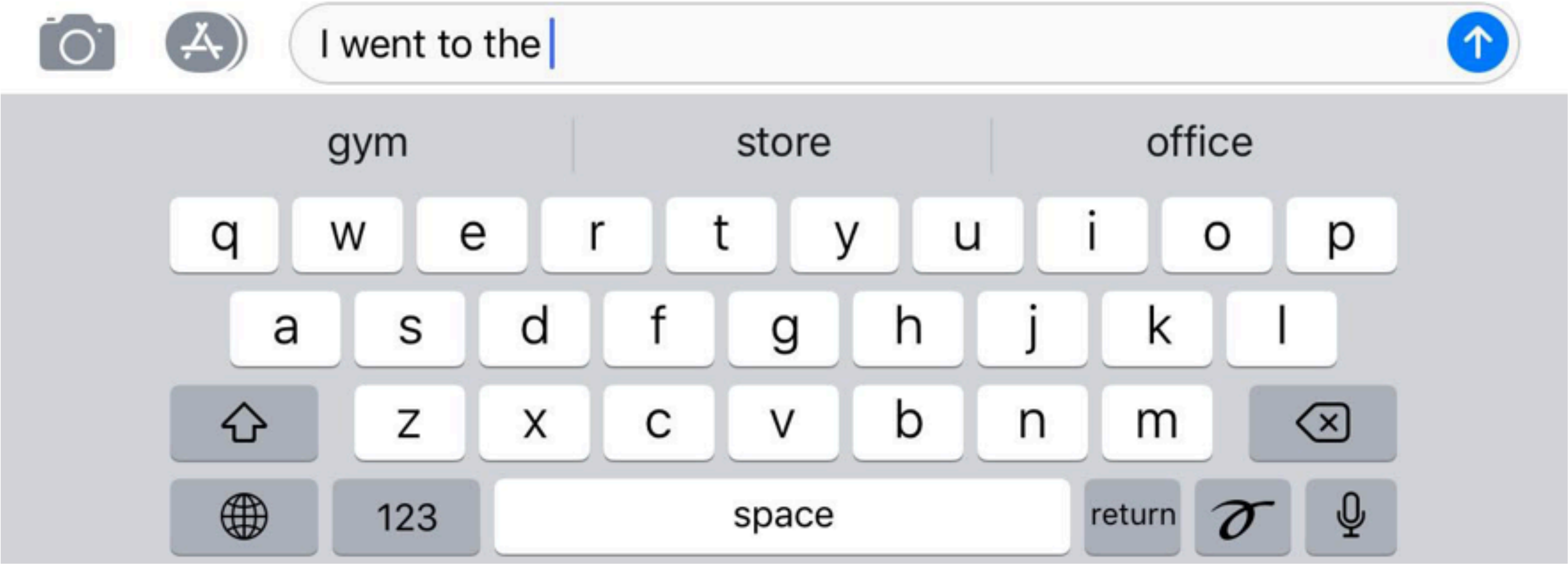
Language Modelling

Generate Text



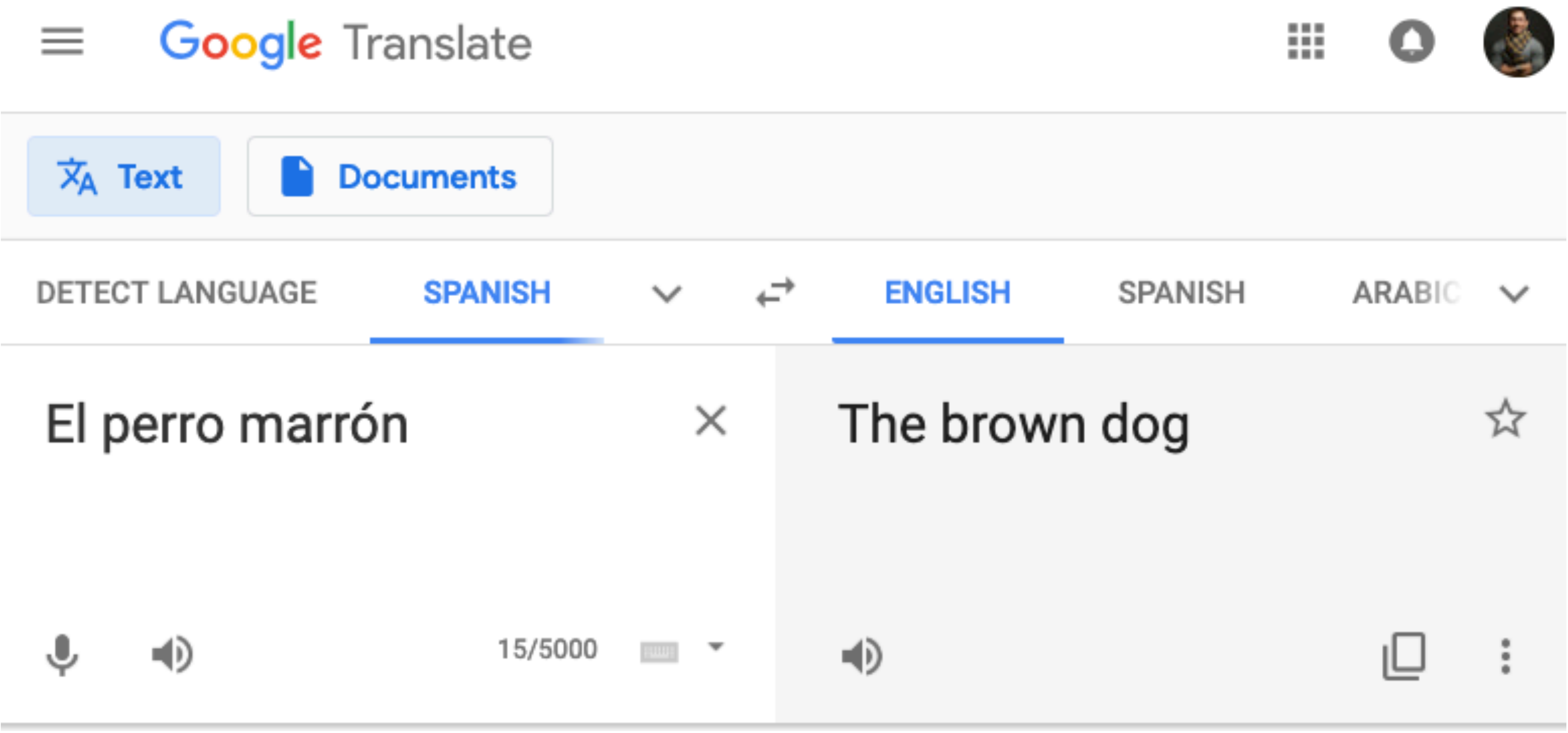
Language Modelling

Generate Text



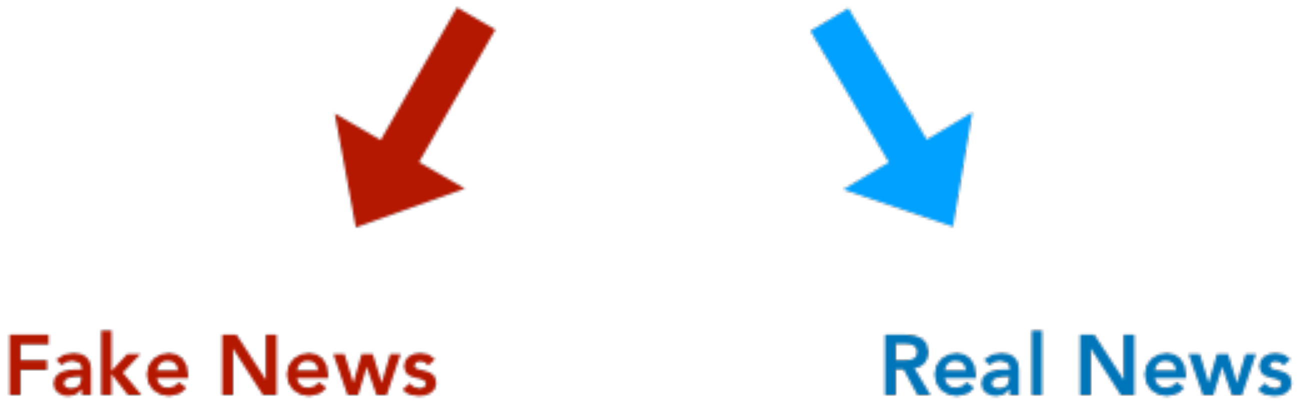
Language Modelling

Generate Text



Language Modelling

"Drug kingpin El Chapo testified that he gave MILLIONS to Pelosi, Schiff & Killary. The Feds then closed the courtroom doors."



Language Modelling

A Language Model is useful for:

Generating Text

- Auto-complete
- Speech-to-text
- Question-answering / chatbots
- Machine translation

Classifying Text

- Authorship attribution
- Detecting spam vs not spam

And much more!

Language Modelling

Slide adapted from Luke Zettlemoyer @ UW 2018

Scenario: assume we have a finite vocabulary V

V^* represents the **infinite set** of strings/sentences that we could construct

e.g., V^* = {a, a dog, a frog, dog a, dog dog, frog dog, frog a dog, …}

Data: we have a training set of sentences $x \in V^*$

Problem: estimate a probability distribution:

$$\sum_{x \in V^*} p(x) = 1$$

$$p(the) = 10^{-2}$$

$$p(the, sun, okay) = 2x10^{-13}$$

$$p(waterfall, the, icecream) = 2x10^{-18}$$

Motivation

"Wreck a nice beach" vs "Recognize speech"

"I ate a cherry" vs "Eye eight uh Jerry!"

"What is the weather today?"

"What is the whether two day?"

"What is the whether too day?"

"What is the Wrether today?"

How can we build a language model?

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Important Terminology

a word **token** is a specific occurrence of a word in a text

a word **type** refers to the general form of the word, defined by its lexical representation

If our corpus were just "I ran and ran and ran", you'd say we have:

- 6 word **tokens** [I, ran , and , ran , and , ran]
- 3 word **types**: {I, ran, and}

Language Modelling

Naive Approach: unigram model

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t)$$

Assumes each word is independent of all others.

Language Modelling

Naive Approach: unigram model

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t)$$

Assumes each word is independent of all others.

$$\mathrm{P}(\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}, \boldsymbol{w_5}) = \mathrm{P}(\boldsymbol{w_1}), \boldsymbol{P}(\boldsymbol{w_2}), \boldsymbol{P}(\boldsymbol{w_3})\boldsymbol{P}(\boldsymbol{w_4})\boldsymbol{P}(\boldsymbol{w_5})$$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

w_1 w_2 w_3 w_4 w_5

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = 100{,}000$$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{w_i}) = \frac{n_{w_i}(\boldsymbol{d})}{n_{w_*}(\boldsymbol{d})}$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w_i}(\boldsymbol{d})$ = # of times word $\boldsymbol{w_i}$ appears in $\boldsymbol{d}$

$n_{w_*}(\boldsymbol{d})$ = # of times any word $\boldsymbol{w}$ appears in $\boldsymbol{d}$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{w_i}) = \frac{n_{w_i}(\boldsymbol{d})}{n_{w_*}(\boldsymbol{d})}$$

$$\mathrm{P}(\mathrm{Anqi}) = \frac{15}{100{,}000} = 0.00015$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w_i}(\boldsymbol{d})$ = # of times word $\boldsymbol{w_i}$ appears in $\boldsymbol{d}$

$n_{w_*}(\boldsymbol{d})$ = # of times any word $\boldsymbol{w}$ appears in $\boldsymbol{d}$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{w_i}) = \frac{n_{w_i}(\boldsymbol{d})}{n_{w_*}(\boldsymbol{d})}$$

$$\mathrm{P}(\mathrm{Anqi}) = \frac{15}{100{,}000} = 0.00015$$

$$\mathrm{P}(\mathrm{was}) = \frac{1{,}000}{100{,}000} = 0.01$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w_i}(\boldsymbol{d})$ = # of times word $\boldsymbol{w_i}$ appears in $\boldsymbol{d}$

$n_{w_*}(\boldsymbol{d})$ = # of times any word $\boldsymbol{w}$ appears in $\boldsymbol{d}$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P(w_i)} = \frac{n_{w_i}(\boldsymbol{d})}{n_{w_*}(\boldsymbol{d})}$$

$$\mathrm{P(Anqi)} = \frac{15}{100{,}000} = 0.00015$$

$$\mathrm{P(was)} = \frac{1{,}000}{100{,}000} = 0.01$$

⋮

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w_i}(\boldsymbol{d})$ = # of times word $\boldsymbol{w_i}$ appears in $\boldsymbol{d}$

$n_{w_*}(\boldsymbol{d})$ = # of times any word $\boldsymbol{w}$ appears in $\boldsymbol{d}$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\text{Anqi, was, late, for, class}) = \mathrm{P}(\text{Anqi})\mathrm{P}(\text{was})\ \mathrm{P}(\text{late})\ \mathrm{P}(\text{for})\ \mathrm{P}(\text{class})$$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\text{Anqi}, \text{was}, \text{late}, \text{for}, \text{class}) = \mathrm{P}(\text{Anqi})\mathrm{P}(\text{was})\,\mathrm{P}(\text{late})\,\mathrm{P}(\text{for})\,\mathrm{P}(\text{class})$$

$$= 0.00015 * 0.01 * 0.004 * 0.03 * 0.0035$$

$$= 6.3 * 10^{13}$$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{Anqi, was, late, for, class}) = \mathrm{P}(\mathrm{Anqi})\mathrm{P}(\mathrm{was})\ \mathrm{P}(\mathrm{late})\ \mathrm{P}(\mathrm{for})\ \mathrm{P}(\mathrm{class})$$

$$= 0.00015 * 0.01 * 0.004 * 0.03 * 0.0035$$

$$= 6.3 * 10^{13}$$

This iterative approach is much more efficient than dividing by all possible sequences of length 5

Unigram Model

$$\mathrm{P}(\text{Anqi}, \text{was}, \text{late}, \text{for}, \text{class}) > \mathrm{P}(\text{Anqi}, \text{was}, \text{late}, \text{for}, \text{asdfjkl;})$$

$$\mathrm{P}(\text{Anqi}, \text{was}, \text{late}, \text{for}, \text{the}) = ?$$

UNIGRAM ISSUES?

?

UNIGRAM ISSUES?

1. Probabilities become too small

2. Out-of-vocabulary words <UNK>

3. Context doesn't play a role at all

$$P(\text{"Anqi was late for class"}) = P(\text{"class for was late Anqi"})$$

4. Sequence generation: What's the most likely next word?

Anqi was late for class _____

Anqi was late for class the

Anqi was late for class the the

UNIGRAM ISSUES?

Problem 1: Probabilities become too small

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t)$$

UNIGRAM ISSUES?

Problem 1: Probabilities become too small

$$P(w_1, \dots, w_T) = \prod_{t=1}^{T} p(w_t)$$

Solution:

$$\log \prod_{t=1}^{T} p(w_t) = \sum_{t=1}^{T} \log(p(w_i))$$

even $\log(10^{-100}) = -230.26$ is manageable

UNIGRAM ISSUES?

Problem 2: Out-of-vocabulary words <UNK>

$$p(COVID19) = 0$$

UNIGRAM ISSUES?

Problem 2: Out-of-vocabulary words <UNK>

$$p(COVID19) = 0$$

Solution: Smoothing

(give every word's count some inflation)

$$\mathrm{P}(w) = \frac{n_w(\boldsymbol{d})}{n_{w_*}}$$

UNIGRAM ISSUES?

Problem 2: Out-of-vocabulary words <UNK>

$$p(COVID19) = 0$$

Solution: Smoothing

(give every word's count some inflation)

$$\mathrm{P}(\mathrm{w}) = \frac{n_w(\boldsymbol{d})+\alpha}{n_{w_*}+\alpha|V|}$$

$$\mathrm{P}(\mathrm{Anqi}) = \frac{15+\alpha}{100{,}000+\alpha|V|}$$

$|V|$ = the # of unique words types in vocabulary (including an extra 1 for <UNK>)

$$\mathrm{P}(\mathrm{COVID19}) = \frac{0+\alpha}{100{,}000+\alpha|V|}$$

Two important notes:

1. Generally, α values are small (e.g., 0.5 – 2)

2. When a word w isn't found within the training corpus $\boldsymbol{d}$ you should replace it with <UNK> (or *U*)

$|V|$ = the # of unique words types in vocabulary (including an extra 1 for <UNK>)

$$P(\text{COVID19}) = \frac{0+\alpha}{100{,}000 + \alpha|V|}$$

UNIGRAM ISSUES?

Problems 3 and 4: Context doesn't play a role at all

$$P(\text{"Anqi was late for class"}) = P(\text{"class for was late Anqi"})$$

Question: How can we factor in context?

UNIGRAM ISSUES?

Easiest Approach:

Instead of words being completely independent, condition each word on its immediate predecessor

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Bigram LM

Look at *pairs* of consecutive words

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

Bigram LM

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = "Anqi was late for class"

w_1 w_2 w_3 w_4 w_5

$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})$

Bigram LM

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = "Anqi was late for class"

w_1 w_2 w_3 w_4 w_5

$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})P(\text{late}|\text{was})$

Bigram LM

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = "Anqi was $\boxed{\text{late for}}$ class"

w_1 w_2 w_3 w_4 w_5

$$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})P(\text{late}|\text{was})P(\text{for}|\text{late})$$

Bigram LM

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = "Anqi was late $\boxed{\text{for class}}$"

w_1 w_2 w_3 w_4 w_5

$$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})P(\text{late}|\text{was})P(\text{for}|\text{late})P(\text{class}|\text{for})$$

Bigram LM

You calculate each of these probabilities by simply counting the occurrences

probability

Let $\boldsymbol{X}$ = "Anqi was late $\boxed{\text{for class}}$"

w_1 w_2 w_3 w_4 w_5

$$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})P(\text{late}|\text{was})P(\text{for}|\text{late})P(\text{class}|\text{for})$$

Bigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(w'|w) = \mathrm{P}(w, w') = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,w*}(\boldsymbol{d})}$$

$n_{w,w'}(\boldsymbol{d})$ = # of times words $\boldsymbol{w}$ and $\boldsymbol{w}'$ appear together as a bigram in $\boldsymbol{d}$

$n_{w,w*}(\boldsymbol{d})$ = # of times word $\boldsymbol{w}$ is the first token of a bigram in $\boldsymbol{d}$

Bigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(w'|w) = \mathrm{P}(w, w') = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,w*}(\boldsymbol{d})}$$

$$\mathrm{P}(\text{class}|\text{for}) = \mathrm{P}(\text{for}, \text{class}) = \frac{12}{3{,}000}$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$|W| = n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w,w'}(\boldsymbol{d})$ = # of times words $\boldsymbol{w}$ and $\boldsymbol{w}'$ appear together as a bigram in $\boldsymbol{d}$

$n_{w,w*}(\boldsymbol{d})$ = # of times word $\boldsymbol{w}$ is the first token of a bigram in $\boldsymbol{d}$

BIGRAM ISSUES?

?

BIGRAM ISSUES?

1. Out-of-vocabulary bigrams are 0 → kills the overall probability

2. Could always benefit from more context but sparsity is an issue (e.g., rarely seen 5-grams)

3. Storage becomes a problem as we increase the window size

4. No semantic information conveyed by counts (e.g., vehicle vs car)

BIGRAM ISSUES?

Problem 1: Out-of-vocabulary bigrams

Our current bigram probabilities:

$$P(\mathrm{w}, \mathrm{w}') = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,w*}(\boldsymbol{d})}$$

Q: What should we do?

How we smoothed unigrams:

$$P(\mathrm{w}) = \frac{n_w(\boldsymbol{d}) + \alpha}{n_{w_*} + \alpha|V|}$$

$|V|$ = the # of unique words types in vocabulary (including an extra 1 for <UNK>)

BIGRAM ISSUES?

Problem 1: Out-of-vocabulary bigrams

Imagine our current string x includes "COVID19 harms ribofliptonik …"

In our training corpus d, we've never seen:

"COVID19 harms" or "harms ribofliptonik"

But we've seen the unigram "harms", which provides useful information:

BIGRAM ISSUES?

Problem 1: Out-of-vocabulary bigrams

Solution: unigram-backoff for smoothing

$$P(\mathrm{w},\mathrm{w}') = \frac{n_{w,w'}(\boldsymbol{d}) + \beta * P(\mathrm{w}')}{n_{w,w*}(\boldsymbol{d}) + \beta}$$

$$P(\mathrm{w}') = \frac{n_{w'}(\boldsymbol{d}) + \alpha}{n_{w_*} + \alpha|V|}$$

$|V|$ = the # of unique words types in vocabulary (including an extra 1 for <UNK>)

BIGRAM ISSUES?

Problem 1: Out-of-vocabulary bigrams

Solution: unigram-backoff for smoothing

Our model is properly parameterized with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

So, instead of calculating the probability of text, we are actually interested in fixing the parameters at particular values and determining the likelihood of the data.

$|V|$ = the # of unique words types in vocabulary
(including an extra 1 for <UNK>)

BIGRAM ISSUES?

For a fixed $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\theta(\mathrm{w}, \mathrm{w}') = \frac{n_{w,w'}(\boldsymbol{d}) + \beta * \theta(\mathrm{w}')}{n_{w,w*}(\boldsymbol{d}) + \beta}$$

$$\theta(\mathrm{w}') = \frac{n_{w'}(\boldsymbol{d}) + \alpha}{n_{w_*} + \alpha|V|}$$

$|V|$ = the # of unique words types in vocabulary
(including an extra 1 for <UNK>)

IMPORTANT:

It is common to pad sentences with <S> tokens on each side, which serve as boundary markers. This helps LMs learn the transitions between sentences.

Let $\boldsymbol{X}$ = "I ate. Did you?" → $\boldsymbol{X}$ = "<S> I ate <S> Did you? <S>"

$w_1\ w_2\ \ w_3\ \ w_4$ $\qquad$ $w_1\ w_2\ w_3\ \ w_4\ \ w_5\ \ w_6\ \ w_7$

Generation

- We can also use these LMs to **generate** text
- Generate the very first token manually by making it be <S>
- Then, generate the next token by sampling from the probability distribution of possible next tokens (the set of possible *next* tokens sums to 1)
- When you generate be <S> again, that represents the end of the current sentence

Example of Bigram generation

- Force a <S> as the first token
- Of the bigrams that start with <S>, probabilistically pick one based on their likelihoods
- Let's say the chosen bigram was <S>_The
- Repeat the process, but now condition on "The". So, perhaps the next select Bigram is "The_dog"
- The sentence is complete when you generate a bigram whose second half is <S>

Imagine more context

Language Modelling

Better Approach: n-gram model

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1}, \ldots, x_1)$$

Let's factor in context (in practice, a window of size **n**)

Language Modelling

Better Approach: n-gram model

$$P(x_1, \dots, x_T) = \prod_{t=1}^{T} \underbrace{p(x_t | x_{t-1}, \dots, x_1)}_{\text{The likelihood of any event occurring hinges upon all prior events occurring}}$$

Language Modelling

Better Approach: n-gram model

$$P(x_1, \dots, x_T) = \overbrace{\prod_{t=1}^{T} \underbrace{p(x_t | x_{t-1}, \dots, x_1)}_{\text{The likelihood of any event occurring hinges upon all prior events occurring}}}^{\text{This compounds for all subsequent events, too}}$$

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Outline

- Recap where we are
- NLP Introduction
- Language Models
  - Unigrams
  - Bigrams
  - Perplexity

Perplexity

N-gram models seem useful, but how can we measure how good they are?

Can we just use the likelihood values?

Perplexity

Almost!

The likelihood values aren’t adjusted for the length of sequences, so we would need to normalize by the sequence lengths.

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity, denoted as PP, is the inverse probability of the test set, normalized by the number of words.

$$PP(w_1, \ldots, w_T) = p(w_1, w_2, \ldots, w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{p(w_1, w_2, \ldots, w_N)}}$$

Perplexity

Perplexity is also equivalent to the exponentiated negative log-likelihood normalized:

$$PP(w_1, \dots, w_T) = p(w_1, w_2, \dots, w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{p(w_1, w_2, \dots, w_N)}}$$

$$= 2^{-l}, \text{ where l } = \frac{1}{N}\sum_{i=1}^{n} \log(p(w_i))$$

Perplexity

Very related to entropy, **perplexity** measures the **uncertainty** of the model for a particular dataset. So, very high perplexity scores correspond to having tons of uncertainty (which is bad).

Perplexity also represents the **average** number of bits needed to represent each word. You can view this as the branching factor at each step. That is, the more branches (aka bits) at each step, the more uncertainty there is.

Perplexity

Good models tend to have perplexity scores around 40-100 on large, popular corpora.

If our model assumed a uniform distribution of words, then our perplexity score would be:

$|V|$ = the # of unique word types

Perplexity

Example: let our corpus X have only 3 unique words {the, dog, ran} but have a length of N.

$$PP(X) = \sqrt[N]{\frac{1}{\left(\frac{1}{3}\right)^N}} = \sqrt[N]{3^N} = 3$$

Perplexity

More generally, if we have M unique words for a sequence of length N.

$$PP(X) = \sqrt[N]{\frac{1}{\left(\frac{1}{M}\right)^N}} = \sqrt[N]{M^N} = M$$

Perplexity

Example perplexity scores: when trained on a corpus of 38 million words and tested on 1.5 million words:

model	perplexity
unigram	962
bigram	170
trigram	109

SUMMARY

- Language models estimate the probability of sequences and can predict the most likely next word
- We can probabilistically generate sequences of words
- We can measure performance of any language model
- Unigrams provide no context and are not good
- Bi-grams and Tri-grams are better but still have serious weaknesses