The "Deep Learning for NLP" Lecture Roadmap

Lecture 5: Text Vectorization and Bag-of-Words

Lecture 6: Embeddings
Lecture 7: Transformers – Theory (1/2)
Lecture 8: Transformers – Applications (2/2)
Lecture 9: Gen AI: LLMs and RAG
Lecture 10: Gen AI: LLMs and Parameter Efficient Fine Tuning/ LORA
Lecture 11: Diffusion Models: Text to Image

MIT MANAGEMENT SLOAN SCHOOL

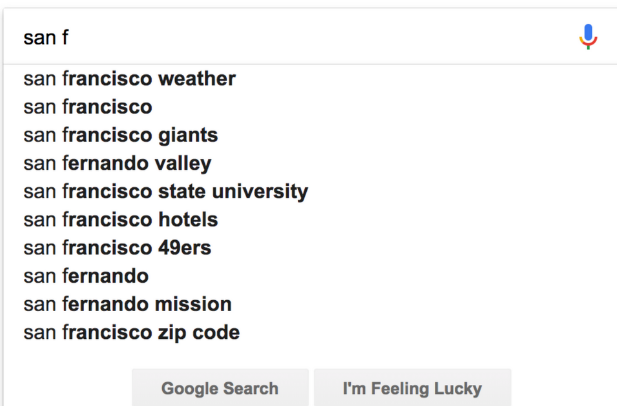
15.S04: Hands-on Deep Learning
Spring 2024
Farias, Ramakrishnan

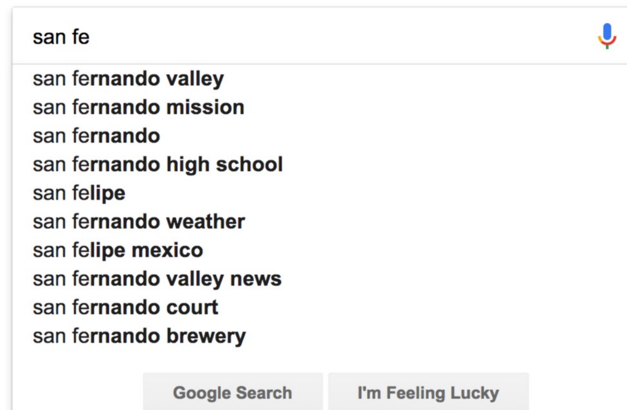
Why Natural Language Processing (NLP)?

- Human knowledge is (mostly) natural language text
- The Internet is (mostly) natural language text
- Human communication is (mostly) natural language text
- Cultural production is (mostly) natural language text

Imagine if a system could read and "understand" all this automatically

Why NLP? (15 years ago)

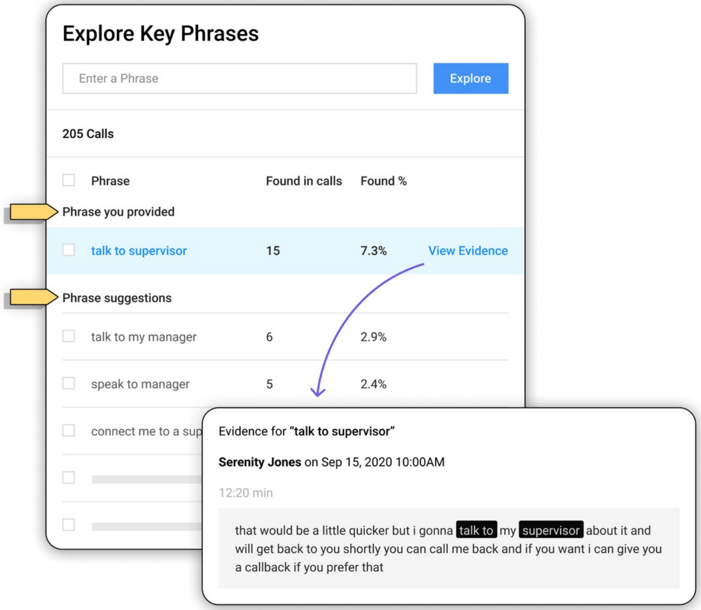


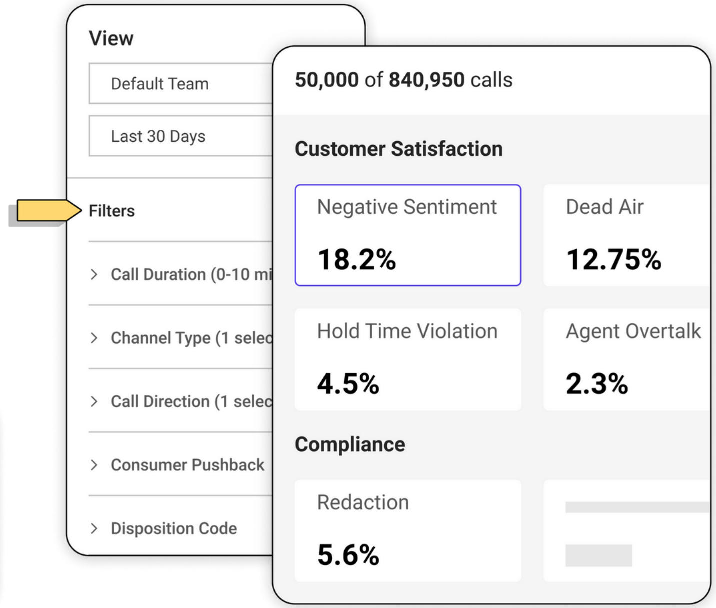


According to Google, Autocomplete

- Saves 200 *years* of typing time, every *day*
- Made mobile possible

Why NLP? (10 years ago)





OBSERVE-AI
replicant
CLARABRIDGE
CHORUS
CallMiner
cogito

AI Powered Speech Analytics

- Transforming sales and CX esp. post COVID
- Market growing at 20% CAGR

But the applicability today is substantially larger!

Non-Generative

- Classification of text (sentiment, abuse, support ticket RPA etc.)
- Summarization (productivity tools, commerce, etc.)
- Information Retrieval (RPA for invoicing, contract analysis, compliance, etc.)

Generative

- **Prediction/generation of text (chatbots, agents etc.)**
- **Co-pilots: code, ecommerce, data science, genera, productivity…**

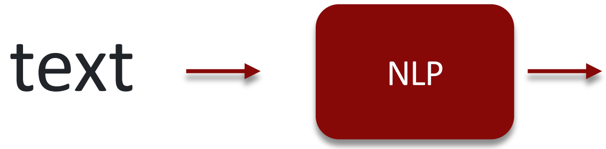
This seemingly simple **generative** capability covers a vast range of applications

Example applications: *Text Classification*

text → LLM → Classification of text for

- Sentiment
- Routing
- Intent
- Filtering
- …

Example applications: Text *Extraction*



Extract data out from free-form text

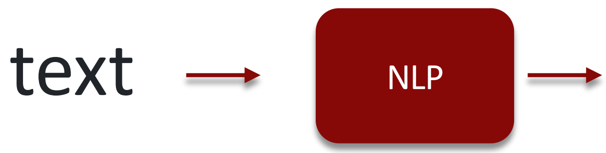
- Company financials from news article
- Customer name and contact info from chat
- Disease and medication codes from doctor's notes
- …

Example applications: Text *Summarization*

text → NLP → Summarize long-form text into

- Bullet points
- Abstracts
- Titles
- …

Example applications: Text *Generation*



- Marketing copy
- Sales emails
- Market summaries
- Job descriptions
- Social media posts
- College application essays 🤔
- …

Example applications: *Code Generation*

text → NLP → Code that implements the input text

Example applications: *Question-Answering*

Question text

\+

Documents

→ NLP →

Chatbots for:

- Medical/legal
- Call centers
- Compliance
- Form filling
- Workflow automation
- …

Example applications: *Question-Answering*

Question text

\+

Documents

→ NLP →

Chatbots for:

- Medical/legal
- Call centers
- Compliance
- Form filling
- Workflow automation
- …

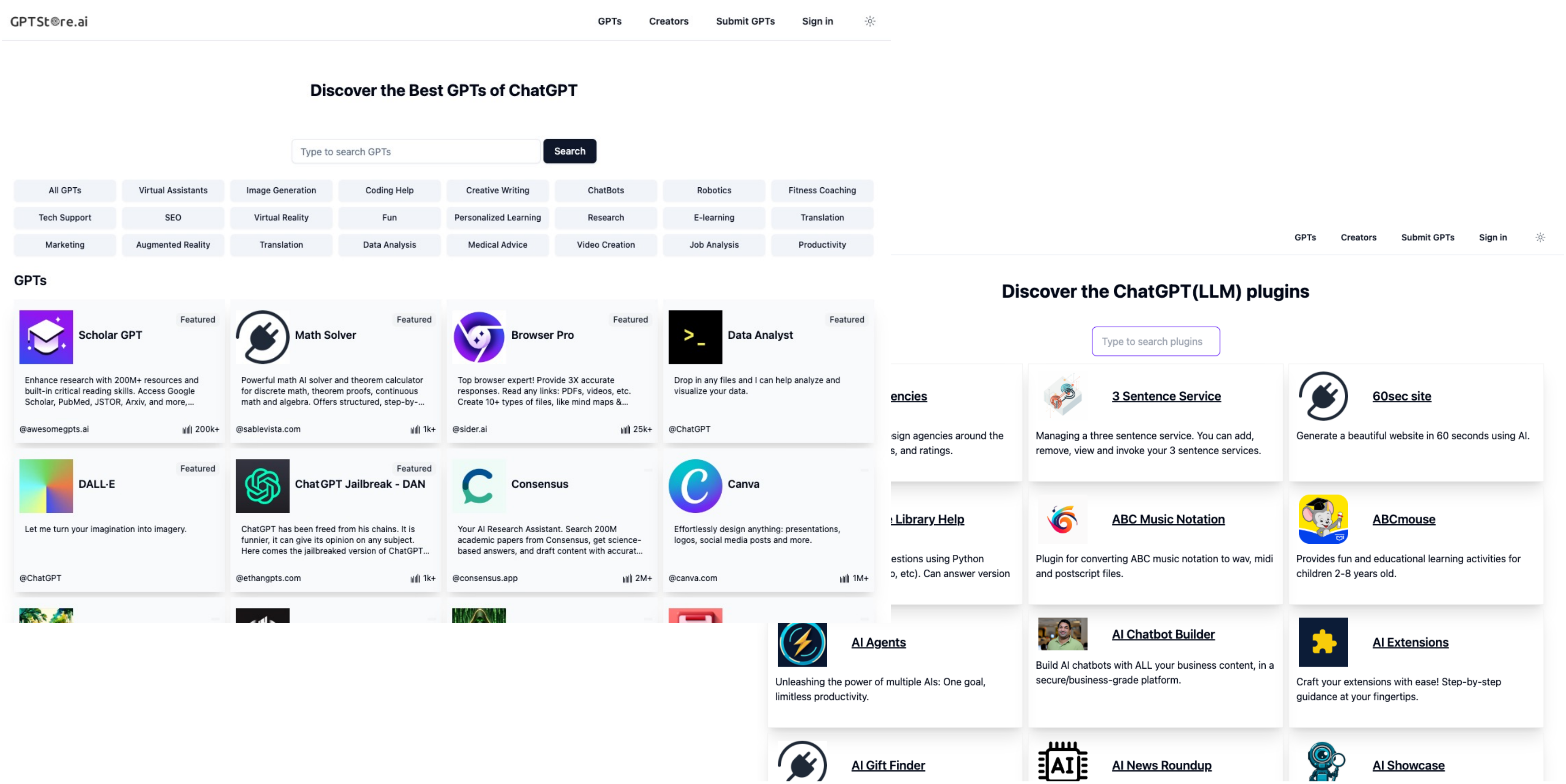
Example domain: *Call Center Optimization*

Call center transcripts
+
Internal documents, FAQs etc

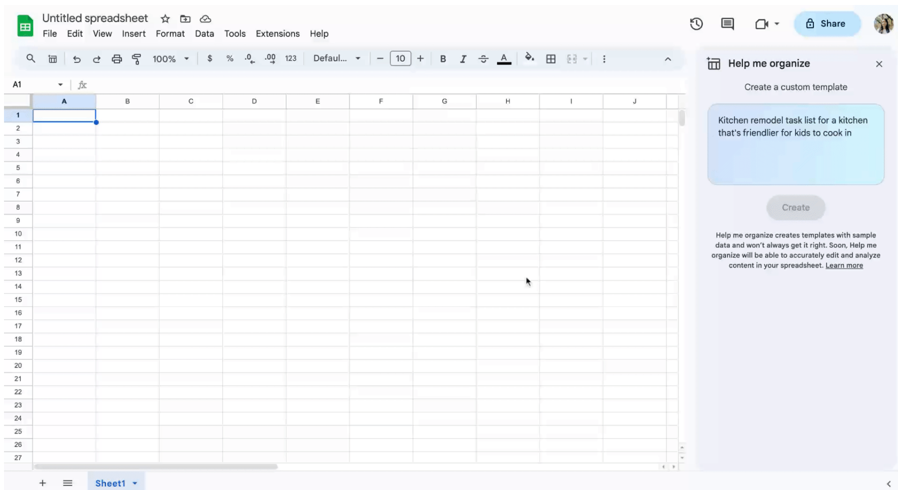
→ NLP →

- Top reasons why customers are upset
- What interventions seem to work?
- What characterizes the best support agents vs the rest?
- How should we grade this agent's interaction with customer X?
- How should we change the call center script for a situation?
- How should we coach the agent in real-time?
- …

This potential is being realized as we speak by the use of Large Language Models (LLMs)



Office productivity suites with LLMs inside are already here!



Copilot for Microsoft 365

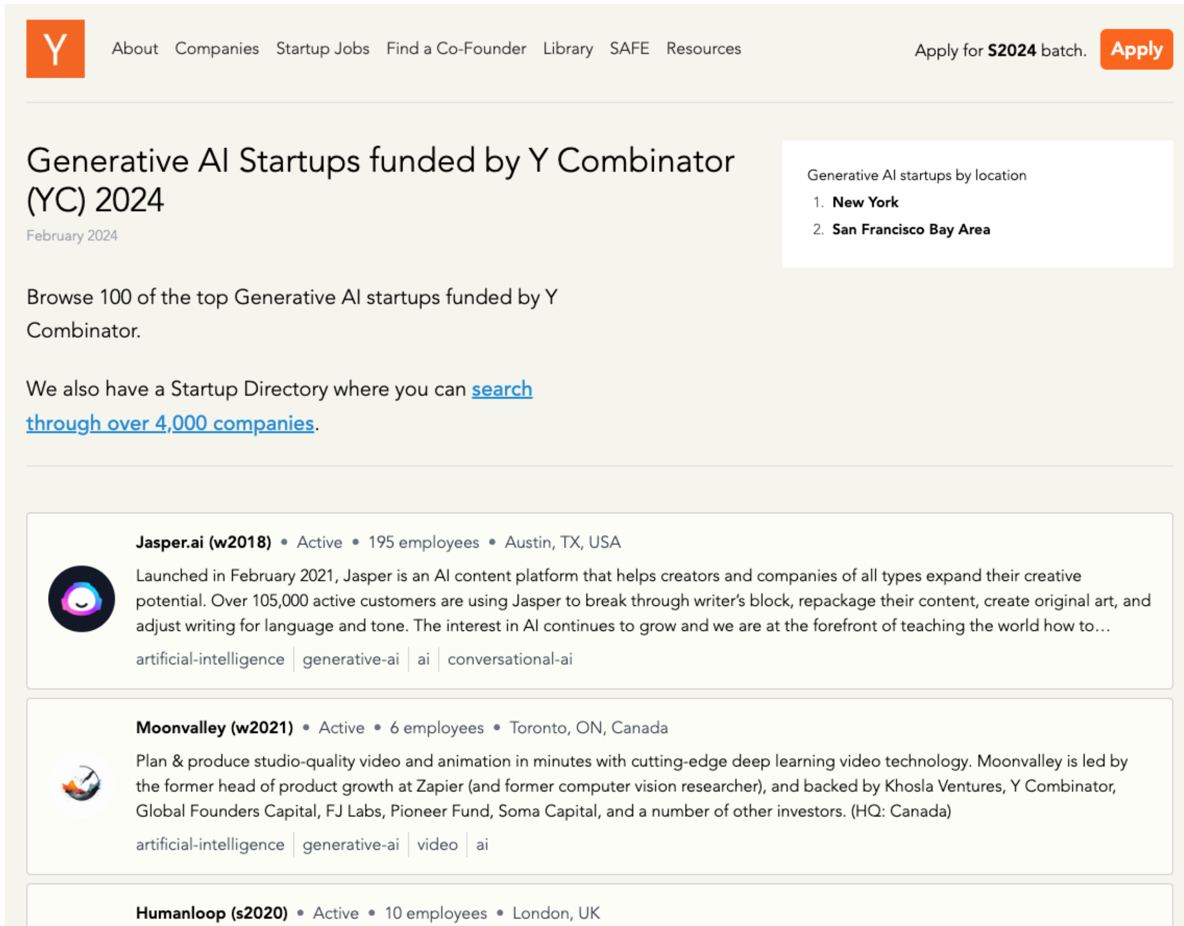
Microsoft Copilot for Microsoft 365 combines the power of large language models (LLMs) with your organization's data – all in the flow of work – to turn your words into one of the most powerful productivity tools on the planet.

It works alongside popular Microsoft 365 apps such as Word, Excel, PowerPoint, Outlook, Teams, and more. Microsoft 365 Copilot provides real-time intelligent assistance, enabling users to enhance their creativity, productivity, and skills.

https://workspace.google.com/solutions/ai/

https://techcommunity.microsoft.com/t5/microsoft-mechanics-blog/how-microsoft-365-copilot-works/ba-p/3822755

There's a startup "gold rush" under way to create products and services



https://www.ycombinator.com/companies/industry/generative-ai

The Arc of NLP Progress – How did we get here?

The Arc of NLP Progress

Hand-crated rules based on linguistics (up to early 1990s) → Statistical/ML (1990s - early 2010s) → Recurrent Neural Networks (2014-2017) → Transformers (2017 - Current)

NLP Progress

Hand-crated rules based on linguistics (up to early 1990s) → Statistical/ML (1990s - early 2010s) → Recurrent Neural Networks (2014-2017) → Transformers (2017 - Current)

"Every time I fire a linguist, the performance of the speech recognizer goes up."

Frederick Jelinek

NLP Progress

1. Hand-crated rules based on linguistics (up to early 1990s)
2. Statistical/ML (1990s - early 2010s)
3. Recurrent Neural Networks (2014-2017)
4. Transformers (2017 - Current)

> *"Every time I fire a linguist, the performance of the speech recognizer goes up."*
>
> Frederick Jelinek

We will *leapfrog* to this in HODL!

20,000 Foot View of the Problem

Like most things, fancy regression!

$$x \longrightarrow \boxed{f(x, w)} \longrightarrow y$$

20,000 Foot View of the Problem

Like most things, fancy regression!

$$x \longrightarrow \boxed{f(x,w)} \longrightarrow y$$

$$x = \text{text}$$

$$y = \textbf{text}\text{, labels, numbers, ...}$$

$$w = \text{weights}$$

$$f(x,w) = \text{A deep neural network}$$

20,000 Foot View of the Problem

Like most things, fancy regression!

$$x \longrightarrow \boxed{f(x, w)} \longrightarrow y$$

Key questions:

- How to represent x. We will focus on this today.

20,000 Foot View of the Problem

Like most things, fancy regression!

$$x \longrightarrow \boxed{f(x, w)} \longrightarrow y$$

Key questions:

- How to represent x. We will focus on this today.
- (Next week) What NN architecture is best for processing text?

Processing Basics

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

This process is called *text vectorization*

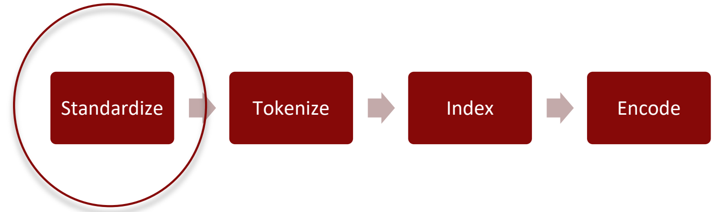
Basic Pre-Processing

Standardize → Tokenize → Index → Encode

We first do these two steps for every sentence in our training dataset*

*aka “training corpus”

Basic Pre-Processing



Standardization

- Strip capitalization, often punctuation and accents *(almost always)*
- Strip 'stop words' e.g., a, the, it, .. *(sometimes)*
- Stemming (e.g., ate, eaten, eating, eaten > [eats]) *(rarely)*

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

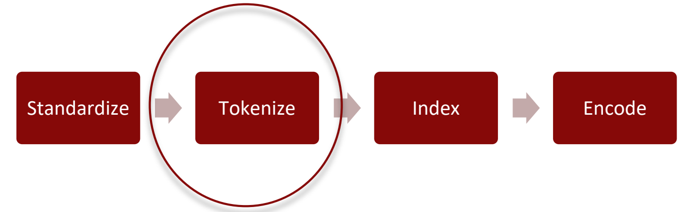
Standardization

- Strip capitalization, often punctuation and accents *(almost always)*
- Strip 'stop words' e.g., a, the, it, .. *(sometimes)*
- Stemming (e.g., ate, eaten, eating, eaten > [eats]) *(rarely)*

Hola! What do you picture when you think of traveling to Mexico? Sipping a real margarita while soaking up the sun on a laid-back beach in Puerto Vallarta?

hola what do you picture when you [thinks] of [travels] to mexico [sips] real margarita while [soaks] up sun on laidback beach in puerto vallarta

Basic Pre-Processing

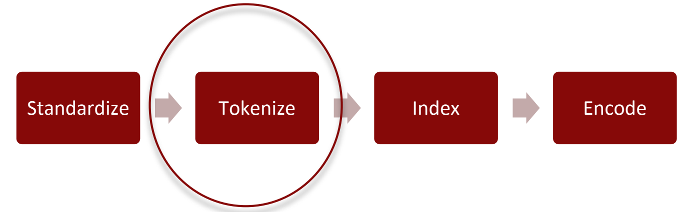


Tokenization

- *Typically*, each word is a token* (i.e., split each string on whitespace)
- [design choice] decide how many consecutive words make up a *token*

*Modern LLMs like ChatGPT use a different tokenization scheme (we will briefly discuss in Lecture 9)

Basic Pre-Processing



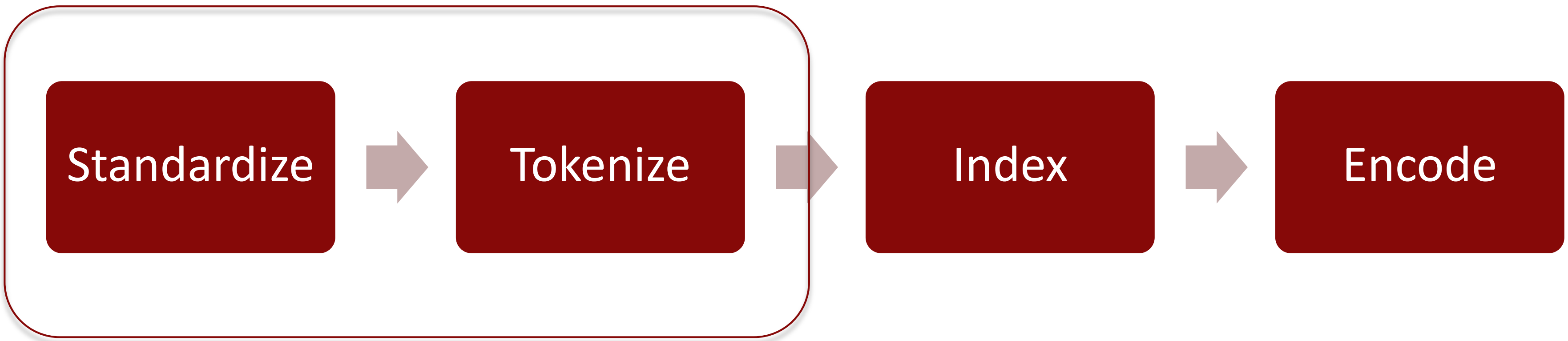
Tokenization

- *Typically*, each word is a token (i.e., split each string on whitespace)
- [design choice] decide how many consecutive words make up a *token*

hola what do you picture when you [thinks] of [travels] to mexico [sips] real margarita while [soaks] up sun on laidback beach in puerto vallarta

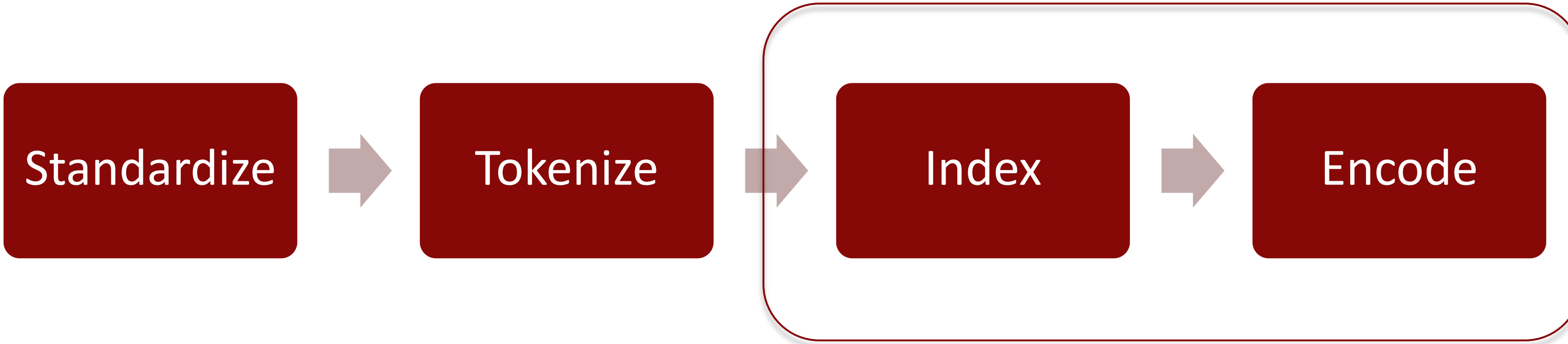
“hola”, “what”, “do”, “you”, “picture”, “when”, “you”, “[thinks]”, “of”, ”[travels]”, “to”, “mexico”, “[sips]”, “real”, “margarita”, “while”, “[soaks]”, “up”, “sun”, “on”, “laidback”, “beach”, “in”, “puerto”, “vallarta”

Basic Pre-Processing



When this is done for every sentence in our training dataset, we have a list of distinct tokens = our vocabulary

Basic Pre-Processing



When this is done for every sentence in our training dataset, we have a list of distinct tokens = our vocabulary

Now we move to the third and fourth stages. In these stages, we only work with the vocabulary

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Indexing: We assign a unique integer to each distinct token in the vocabulary

Token	Integer
<UNK>	0*
a	1
aardvark	2
…	
zebra	50000

*we will come back to this special token later

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is _____________

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is one-hot encoding

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is one-hot encoding

$$\text{<UNK>} \to \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \text{a} \to \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is one-hot encoding

$$\text{<UNK>} \to \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \text{a} \to \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Dimension of encoding vector = max no. of distinct tokens in the text

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is one-hot encoding

$$\text{<UNK>} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \text{a} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Dimension of encoding vector = max no. of distinct tokens in the text + one for <UNK>

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Encoding: We assign a *vector* to each integer in our vocabulary

- The simplest way to do this is one-hot encoding

$$\text{<UNK>} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \text{a} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Dimension of encoding vector = max no. of distinct tokens in the text + one for <UNK>
- This is called the "vocabulary" size

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

At this point,

- we have created a vocabulary from the training corpus and
- every distinct token in our vocabulary has been assigned a one-hot vector.

We are done with basic preprocessing.

Next: How to get a *new* input sentence* ready to be “fed” into a DNN

- Let’s say we have completed STIE on the training corpus and our vocabulary size is 100.

VOCABULARY

← 100 →

UNK a ... an ... car ... cat ... hat ...... mat .... ok ... on .... sat ...... ten ... the ... zebra

*document = sentence = string

Next: How to get a *new* input sentence ready to be "fed" into a DNN

- Let's say we have completed STIE on the training corpus and our vocabulary size is 100.

VOCABULARY

← 100 →

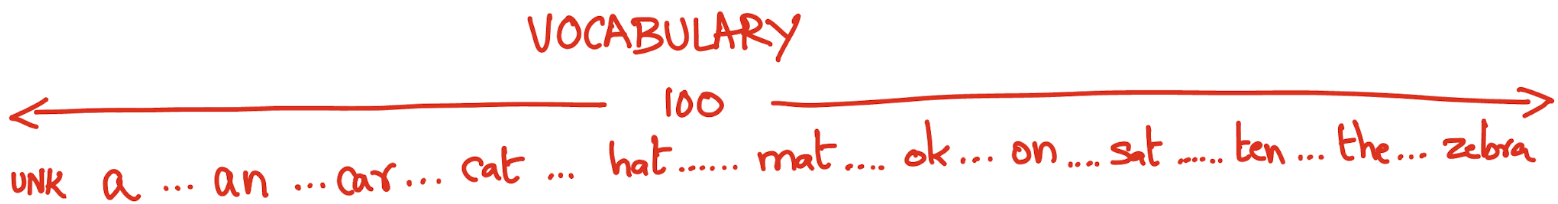
UNK a ... an ... car ... cat ... hat ...... mat .... ok ... on .... sat ..... ten ... the ... zebra

- This input text string arrives - "The cat sat on the mat" – and we run it through STIE

"The cat sat on the mat" → Standardize → Tokenize → Index → Encode →

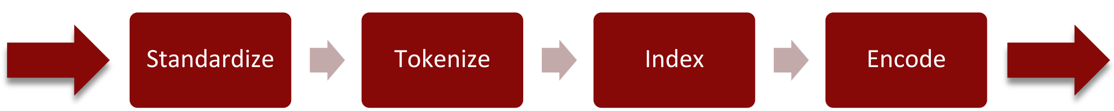
Next: How to get a *new* input sentence ready to be "fed" into a DNN

- Let's say we have completed STIE on the training corpus and our vocabulary size is 100.



- This input text string arrives - "The cat sat on the mat" – and we run it through STIE

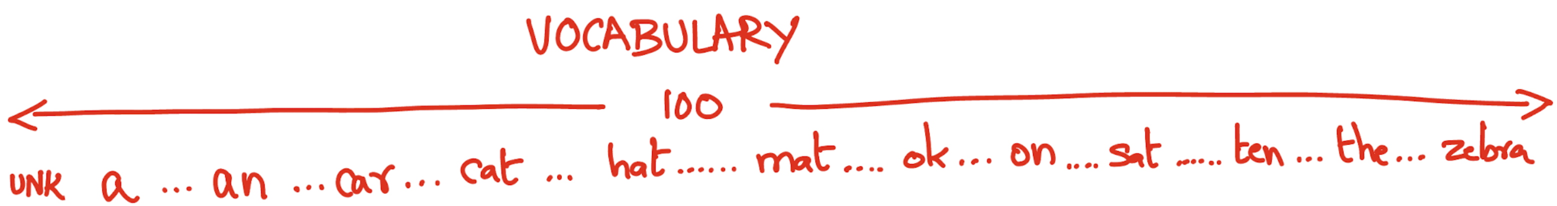
"The cat sat on the mat"



- The output is a table with A rows and B columns. What are A and B?

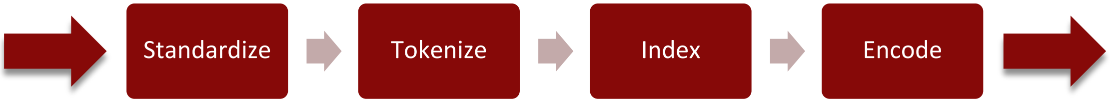
Next: How to get a *new* input sentence ready to be "fed" into a DNN

- Let's say we have completed STIE on the training corpus and our vocabulary size is 100.



- This input text string arrives - "The cat sat on the mat" – and we run it through STIE

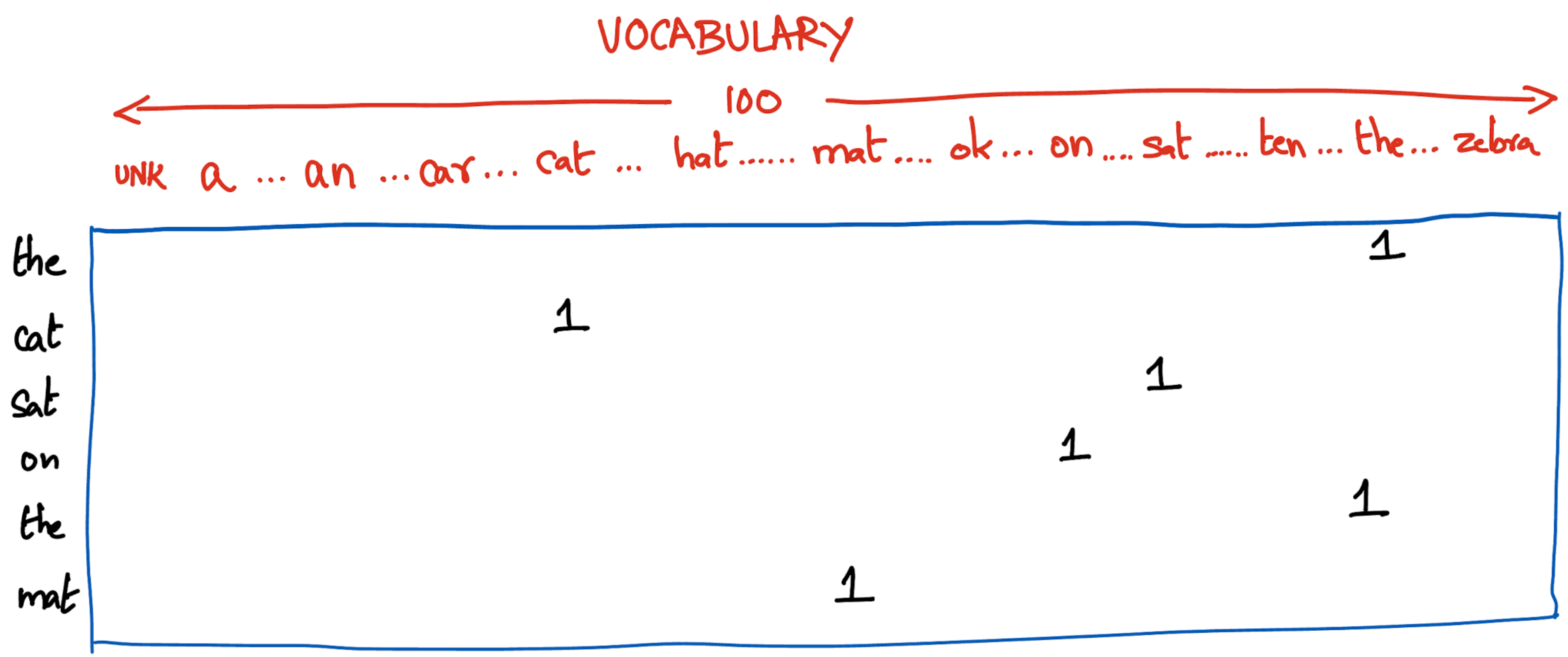
"The cat sat on the mat"



- The output is a 6 x 100 table.

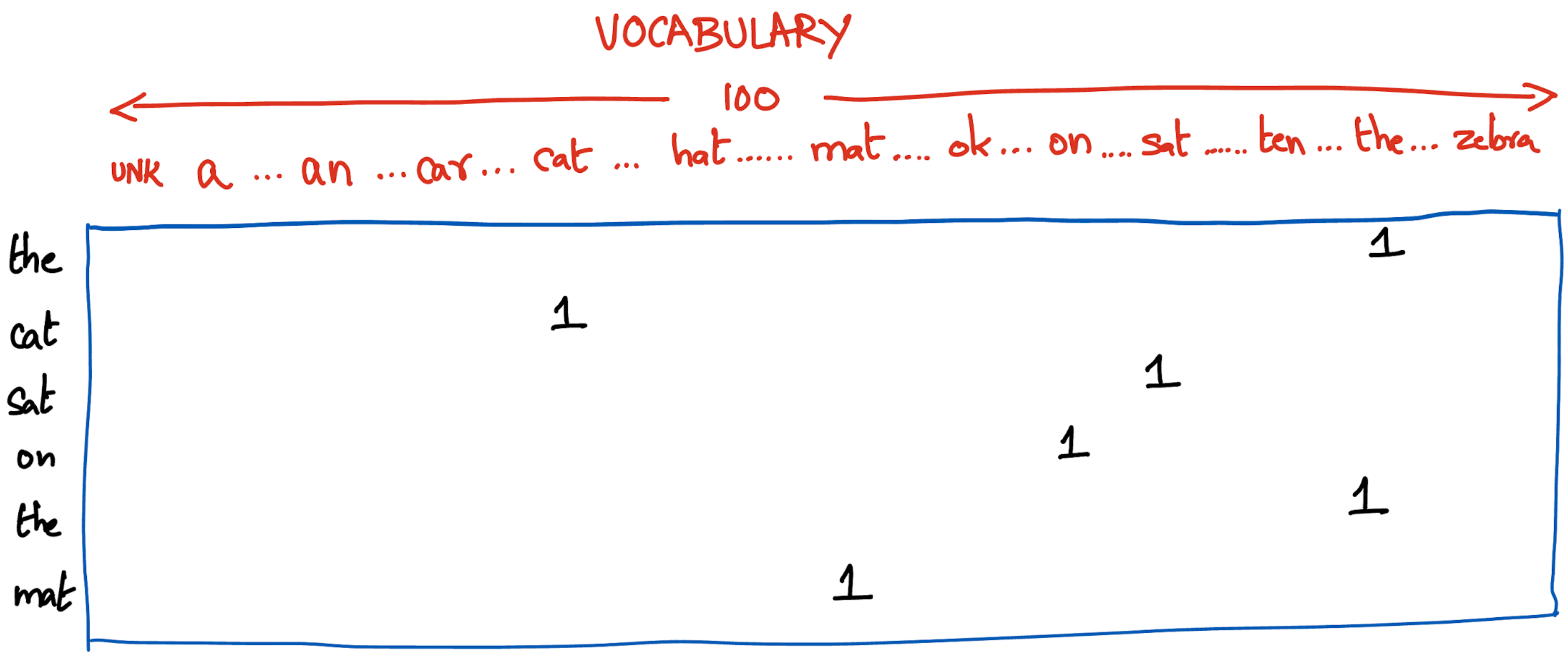
How to get a *new* input sentence ready to be “fed” into a DNN

The output table



How to get a *new* input sentence ready to be "fed" into a DNN

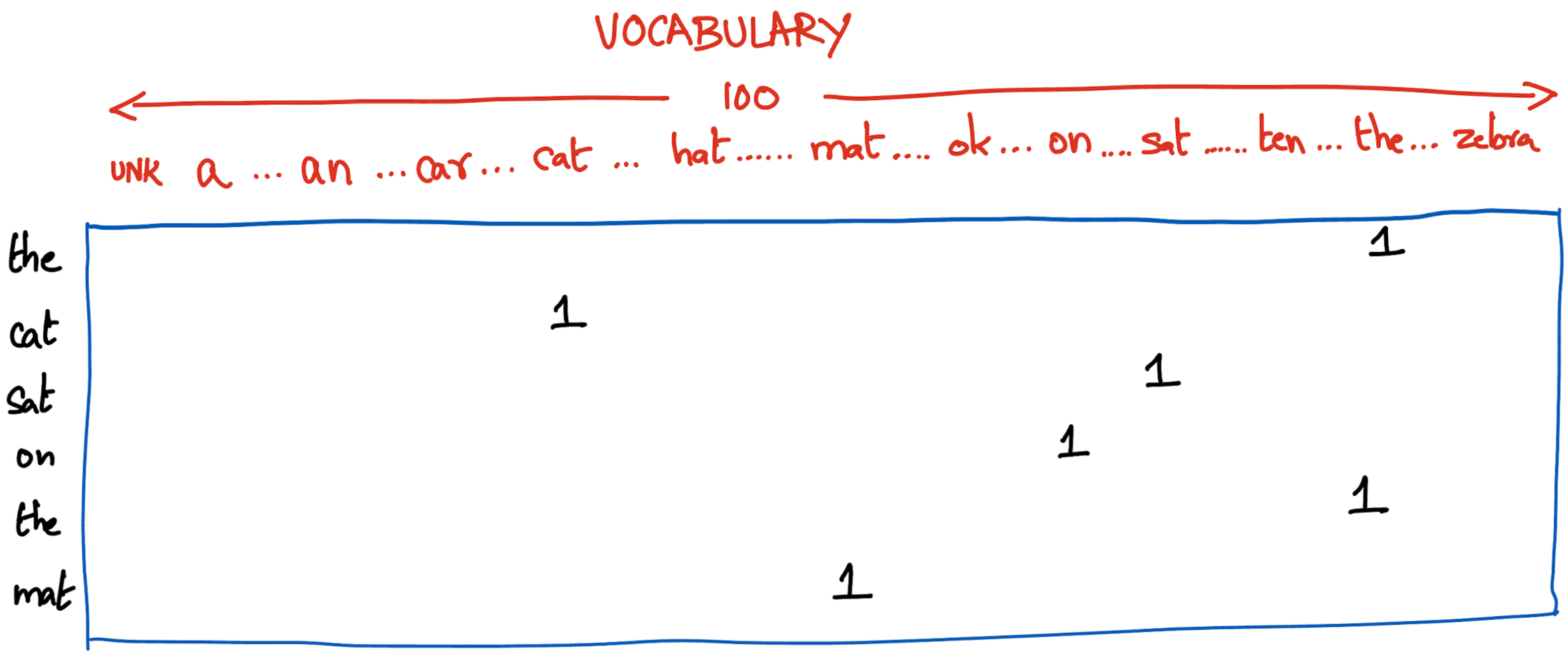
- What's the best way to "feed" this 6 x 100 table of numbers to a DNN?



How to get a *new* input sentence ready to be "fed" into a DNN

- What's the best way to "feed" this 6 x 100 table of numbers to a DNN?

- Can we send this table as-is into a DNN?



How to get a *new* input sentence ready to be “fed” into a DNN

- What’s the best way to “feed” this 8 x 100 table of numbers to a DNN?
- Can we send this table as-is into a DNN?
- A complication: Each incoming sentence may have a different number of words i.e.. may have *varying length*. It will be nice to have a fixed-length input

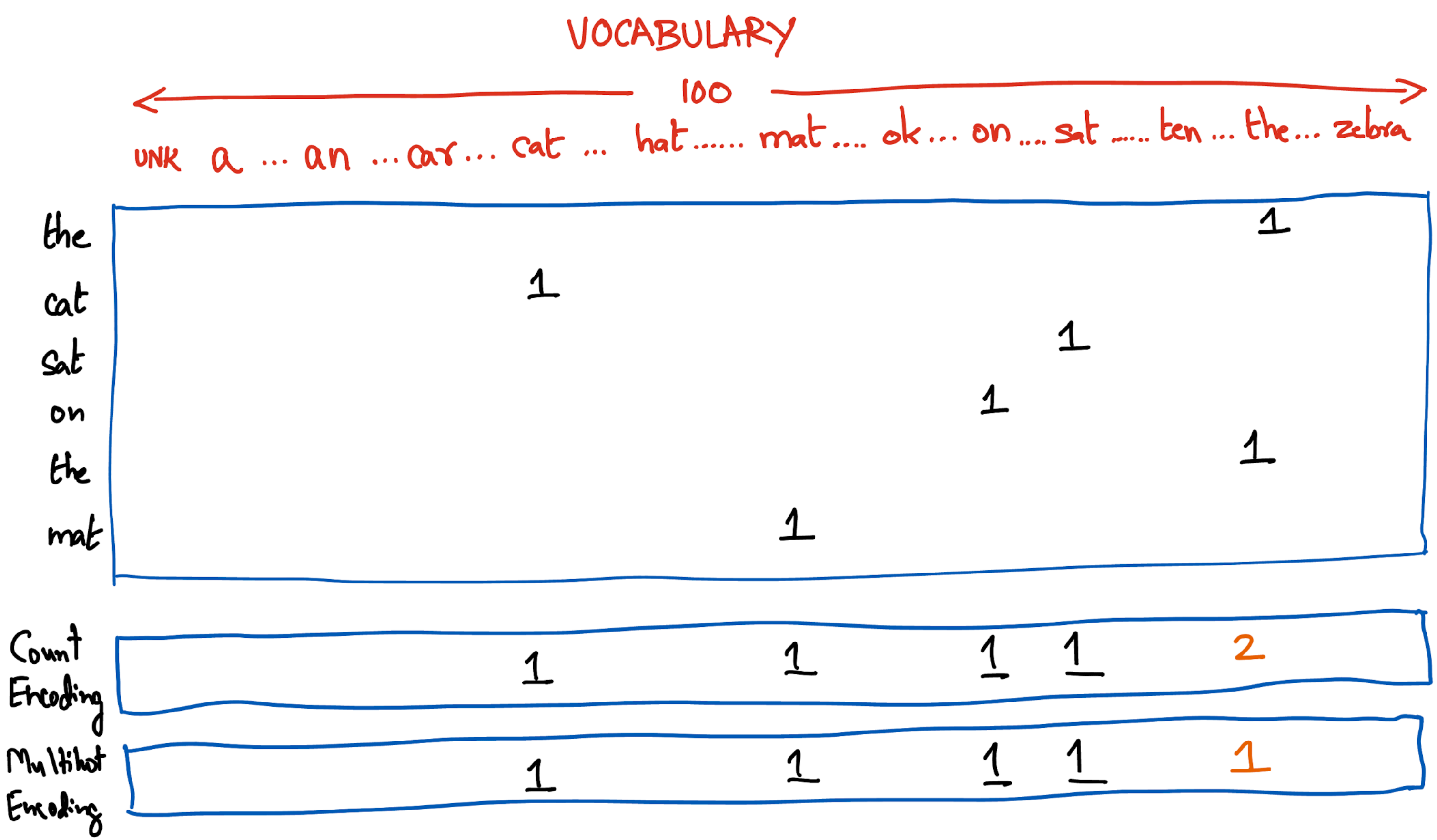
How to get a *new* input sentence ready to be "fed" into a DNN

- What's the best way to "feed" this 8 x 100 table of numbers to a DNN?
- Can we send this table as-is into a DNN?
- A complication: Each incoming sentence may have a different number of words i.e.. may have *varying length*. It will be nice to have a fixed-length input
- How about we "aggregate" the vectors?

How to get a *new* input sentence ready to be "fed" into a DNN

- What's the best way to "feed" this 8 x 100 table of numbers to a DNN?
- Can we send this table as-is into a DNN?
- A complication: Each incoming sentence may have a different number of words i.e.. may have *varying length*. It will be nice to have a fixed-length input
- How about we "aggregate" the vectors?
  - Sum the vectors. This is called "count encoding"
  - "OR" the vectors. This is called "multi-hot encoding"

Example: Count and Multi-hot Encoding



How to get a *new* input sentence ready to be "fed" into a DNN

- What's the best way to "feed" this 8 x 100 table of numbers to a DNN?
- Can we send this table as-is into a DNN?
- A complication: Each incoming sentence may have a different number of words i.e.. may have *varying length*. It will be nice to have a fixed-length input
- How about we "aggregate" the vectors?
  - Sum the vectors. This is called "count encoding"
  - "OR" the vectors. This is called "multi-hot encoding"
- This aggregation approach is called the Bag of Words model

Does the Bag of Words approach have any shortcomings?

- We lose the meaning inherent in the *order* of the words (i.e., we lose "sequentiality")
- If the vocabulary is very long, each input – regardless of its number of tokens – will be a vector that's as long as the size of the vocabulary.
  - This can be somewhat mitigated by choosing only the most-frequent words
  - Nevertheless, this increases the number of weights the model has to learn and thus also the compute time and the risk of overfitting.

Task For NLP 1

Super Serious Application

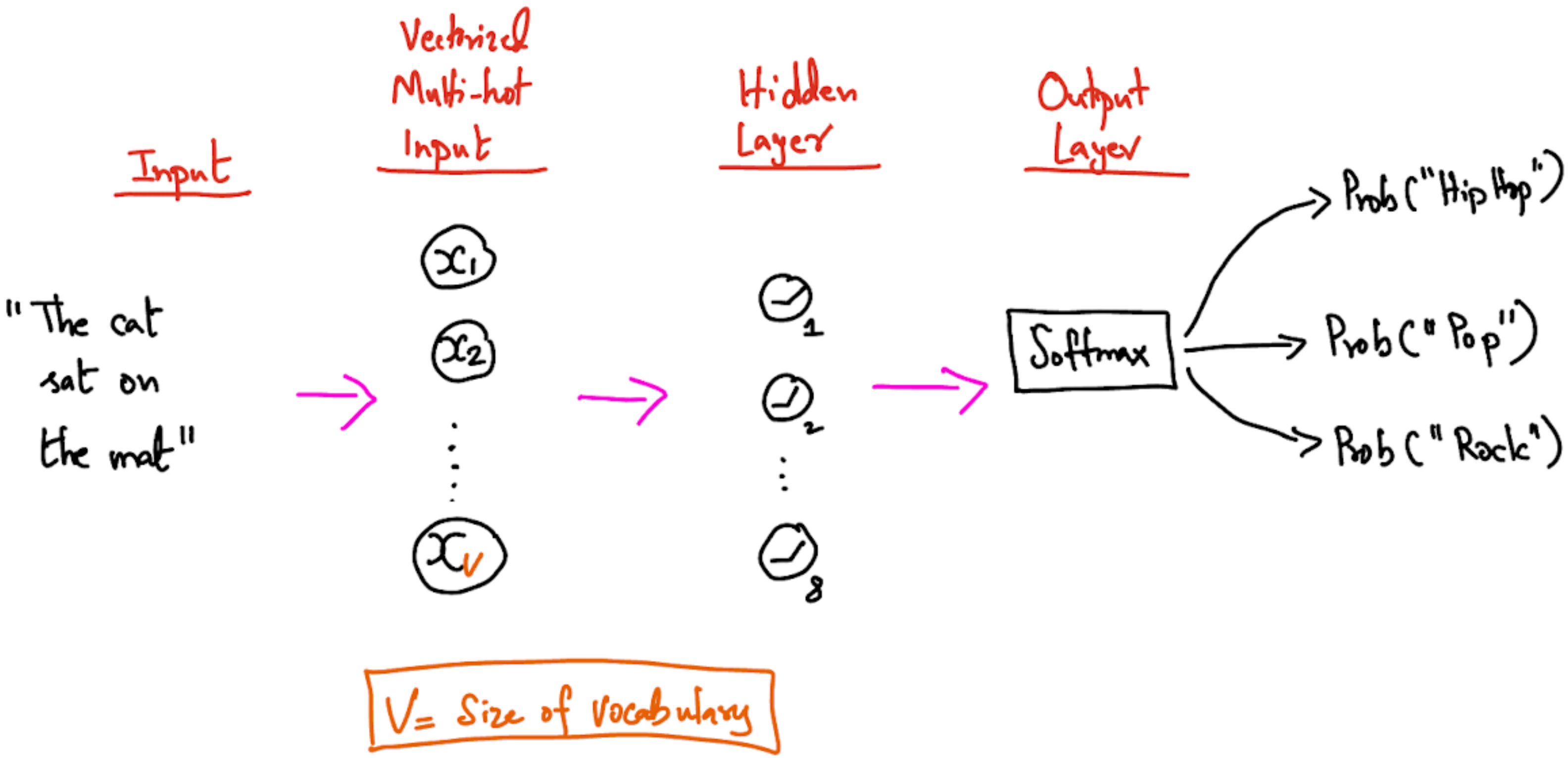
I grew up on the crime side, the New York Times side
Stayin' alive was no jive
Had secondhands, Mom's bounced on old man
So then we moved to Shaolin land

I walked through the door with you
The air was cold
But something about it felt like home somehow
And I, left my scarf there at your sisters house

Can you classify each verse above into *hip-hop, rock* or *pop*?

What’s the simplest NN-based classifier we can build?

What's the simplest NN-based classifier we can build?

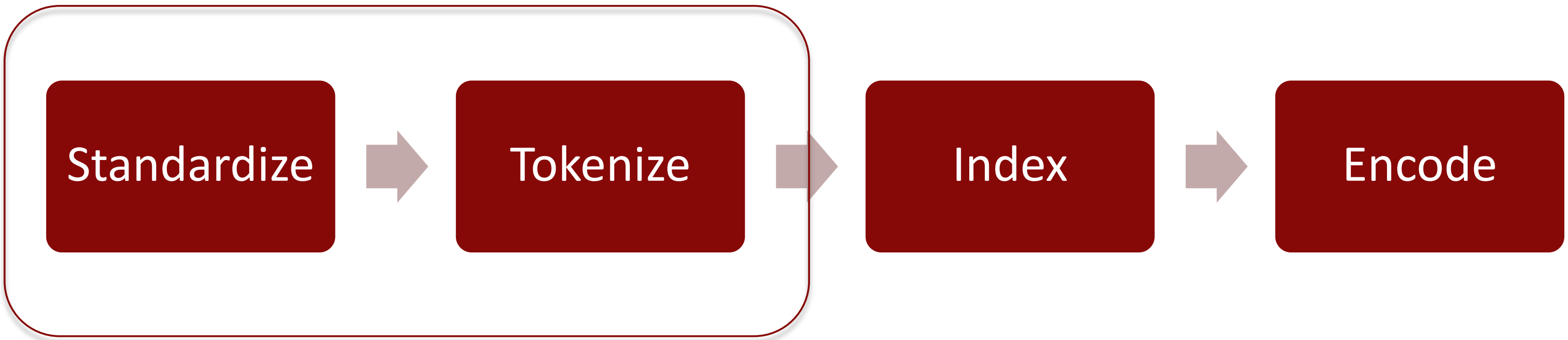


Colab
(text pre-processing, bag-of-words and bigrams)

https://colab.research.google.com/drive/1jzL9lWqLbMyTZocClXSuivFz3CrNHM3h?usp=sharing

Byte Pair Encoding

Any disadvantages to what we described earlier?



Byte Pair Encoding (BPE)

Standardize → BPE Tokenization → Index → Encode

Modern generative models use tokenization schemes that try to address these disadvantages.

The GPT family uses Byte Pair Encoding (BPE)

How BPE works

Key Intuition: Start with each character as a token, and "merge" tokens that most frequently occur next to each other. Stop when the size of your vocabulary reaches a user-defined limit (we will assume 12 in the example below*)

Training corpus	The cat sat on the mat
After Standardization	the cat sat on the mat
Starting vocabulary	[t] [h] [e] [c] [a] [s] [o] [n] [m]

*The GPT family uses a vocab size of 50,000

How BPE works

- Start with the vocabulary [t, h, e, c, a, s, o, n, m]
- [t,h,e] [c,a,t] [s,a,t] [o,n] [t,h,e] [m,a,t]
- What is the frequency of pairs?
  - [t,h] – 2
  - [h,e] – 2
  - [c,a] – 1
  - [a,t] – 3
  - [s,a] – 1
  - [o,n] – 1
  - [m,a] – 1
- So [a,t] is the most frequent (at 3) and we add ‘at’ to the vocabulary which becomes [t, h, e, c, a, s, o, n, m, at]

How BPE works

- [t,h,e] [c,a,t] [s,a,t] [o,n] [t,h,e] [m,a,t]
- [t,h,e] [c,at] [s,at] [o,n] [t,h,e] [m,at] (after merging [a,t])
- What is the frequency of pairs?
  - [t,h] – 2
  - [h,e] – 2
  - [c,at] – 1
  - [s,at] – 1
  - [o,n] – 1
  - [m,at] – 1
- So [t,h] is the most frequent (at 2) and we add 'th' to the vocabulary which becomes [t, h, e, c, a, s, o, n, m, at, th]

How BPE works

- [t,h,e] [c,at] [s,at] [o,n] [t,h,e] [m,at]
- [th,e] [c,at] [s,at] [o,n] [th,e] [m,at] (after merging [t,h])
- What is the frequency of pairs?
  - [th,e] – 2
  - [c,at] – 1
  - [s,at] – 1
  - [o,n] – 1
  - [m,at] – 1
- So [th,e] is the most frequent (at 2) and we add ‘the’ to the vocabulary which becomes …
- [t, h, e, c, a, s, o, n, m, at, th, the]

How BPE works

Key Intuition: Start with each character as a token, and "merge" tokens that most frequently occur next to each other. Stop when the size of your vocabulary reaches a user-defined limit (we will assume 12 in the example below)

Starting vocabulary	[t] [h] [e] [c] [a] [s] [o] [n] [m]
Starting corpus	[t,h,e] [c,a,t] [s,a,t] [o,n] [t,h,e] [m,a,t]
Frequency of adj tokens	[t,h] – 2 [h,e] – 2 [c,a] – 1 [a,t] – 3 [s,a] – 1 [o,n] – 1 [m,a] – 1
Vocabulary after 1st merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at]
Corpus after 1st merge	[t,h,e] [c,at] [s,at] [o,n] [t,h,e] [m,at]
Frequency of adj tokens	[t,h] – 2 [h,e] – 2 [c,at] – 1 [s,at] – 1 [o,n] – 1 [m,at] – 1
Vocabulary after 2nd merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at][th]
Corpus after 2nd merge	[th,e] [c,at] [s,at] [o,n] [th,e] [m,at]
Frequency of adj tokens	[th,e] – 2 [c,at] – 1 [s,at] – 1 [o,n] – 1 [m,at] – 1
Vocabulary after 3rd merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at][th][the]

How BPE works

Key Intuition: Start with each character as a token, and "merge" tokens that most frequently occur next to each other. Stop when the size of your vocabulary reaches a user-defined limit (we will assume 12 in the example below)

Starting vocabulary	[t] [h] [e] [c] [a] [s] [o] [n] [m]
Starting corpus	[t,h,e] [c,a,t] [s,a,t] [o,n] [t,h,e] [m,a,t]
Frequency of adj tokens	[t,h] – 2 [h,e] – 2 [c,a] – 1 [a,t] – 3 [s,a] – 1 [o,n] – 1 [m,a] – 1
Vocabulary after 1st merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at]
Corpus after 1st merge	[t,h,e] [c,at] [s,at] [o,n] [t,h,e] [m,at]
Frequency of adj tokens	[t,h] – 2 [h,e] – 2 [c,at] – 1 [s,at] – 1 [o,n] – 1 [m,at] – 1
Vocabulary after 2nd merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at][th]
Corpus after 2nd merge	[th,e] [c,at] [s,at] [o,n] [th,e] [m,at]
Frequency of adj tokens	[th,e] – 2 [c,at] – 1 [s,at] – 1 [o,n] – 1 [m,at] – 1
Vocabulary after 3rd merge	[t] [h] [e] [c] [a] [s] [o] [n] [m] [at][th][the]

We are done since the vocab size is now 12

How BPE works

The merges happened in this order:

- a,t => at
- t,h => th
- th,e => the

When a new piece of text arrives, the BPE tokenization will apply the merges in the same order.

Example: [t,h,e,_,r,a,t]

- [t,h,e,_,r,at]
- [th,e,_,r,at]
- [the,_,r,at]

We are done since the vocab size is now 12