The "Deep Learning for NLP" Lecture Roadmap

Lecture 5: Text Vectorization and the Bag-of-Words Model

Lecture 6: Embeddings
Lecture 7: Transformers – Theory
Lecture 8: Transformers – Applications, Self-Supervised Learning
Lectures 9-10: LLMs

MIT MANAGEMENT SLOAN SCHOOL

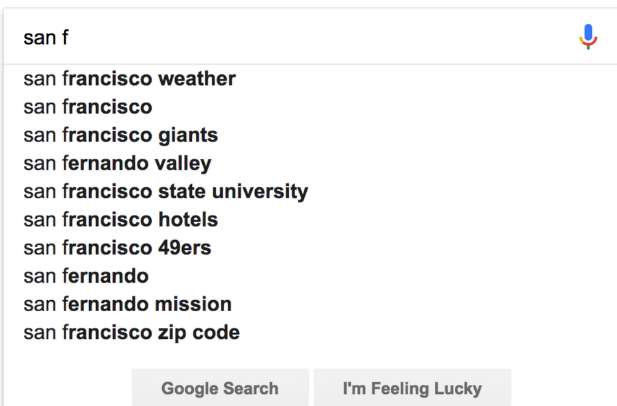
15.S04: Hands-on Deep Learning
Spring 2024
Farias, Ramakrishnan

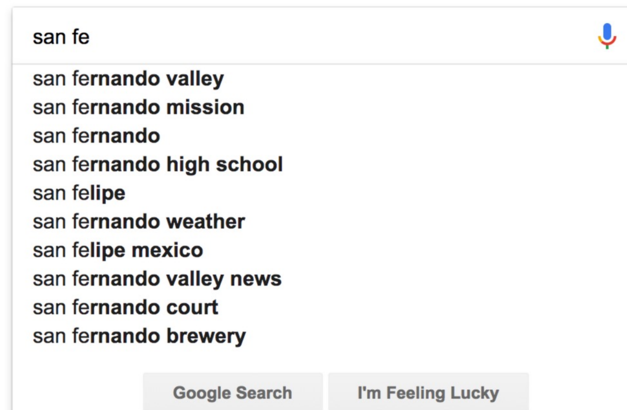
Why Natural Language Processing (NLP)?

- 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

NLP is in action all around us





According to Google, *Autocomplete*

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

NLP is in action all around us 🙂

RA **You**

Write a limerick about the beauty and power of deep learning

ChatGPT

In a world where data flows like a stream,

Deep learning's more than a dream.

It sifts through the noise,

With an elegant poise,

Unveiling insights that gleam!

NLP has extraordinary potential for making products and services smarter

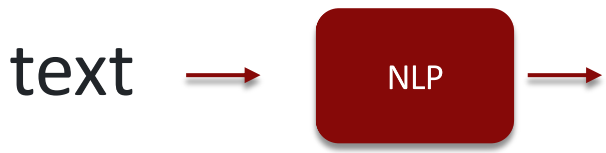
This seemingly simple capability covers a vast range of applications

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

Summarize long-form text into

- Bullet points
- Abstracts
- Titles
- …

Example applications: Text *Generation*

text → NLP →

- 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
- …

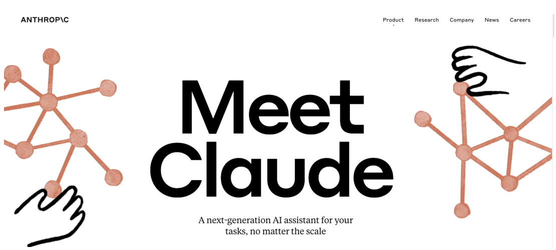
Example domain: *Call Center Optimization*

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?
- …

NLP's potential is now widely recognized in public discourse due to the meteoric rise of Large Language Models



https://www.anthropic.com/index/introducing-claude

🏆 LMSYS Chatbot Arena Leaderboard

Rank	Model	⭐ Arena Elo
1	GPT-4-1106-preview	1254
2	GPT-4-0125-preview	1253
3	Bard (Gemini Pro)	1218
4	GPT-4-0314	1191
5	GPT-4-0613	1164
6	Mistral Medium	1152
7	Claude-1	1150
8	Qwen1.5-72B-Chat	1147
9	Claude-2.0	1132
10	Gemini Pro (Dev API)	1122
11	Claude-2.1	1120
12	Mixtral-8x7b-Instruct-v0.1	1120
13	GPT-3.5-Turbo-0613	1118
14	Gemini Pro	1115
15	Yi-34B-Chat	1111

https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

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

Y Combinator W23 Generative AI Landscape

		Example Use Case	YC W23 Startups
Business Function	Accounting Finance	Automate bookkeeping, data categorization	truewind, ALPHAWATCH
	Marketing	Image, video, and content creation	Booth.ai, Speedy
	Sales	Summarize transcript, automate outbound	Perspectiva, Tennr, FABIUS
	Customer Success	Support agents, automated responses	Hazel, OpenSight, Buff, OfOne, Parabolic, Yuma.ai
	Knowledge Management	Collaboration, summarize meeting notes, project management	Credal.ai, type
	Media	Generate game assets; real-time voice change	Iliad, decoherence, Texel
Engineering Function	Data Analytics	Text to SQL, data transforms	lightski, turntable, Lume, Defog.ai, Merse
	ML Ops Platform	Customize and optimize LLMs	vellum, GRADIENTJ, Baseplate
	Infrastructure	Data platforms, integrations, LLM infrastructure	pyq, stack, Helicone, BerriAI, ANARCHY
	Developer Tools	Observability, manage production, low code	CodeComplete, Meru, Lasso, FOUNDATION, Second

Created by your friends at Truewind (YC W23)

This is a work in progress. Reach out to us if you want to be added to the next iteration

Enterprise vendors are rushing to add NLP features to their products

ARTIFICIAL INTELLIGENCE

Salesforce Announces Einstein GPT, the World's First Generative AI for CRM

https://www.salesforce.com/news/press-releases/2023/03/07/einstein-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

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

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!

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

$$x = \text{text}$$

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

$$w = \text{weights}$$

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

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?

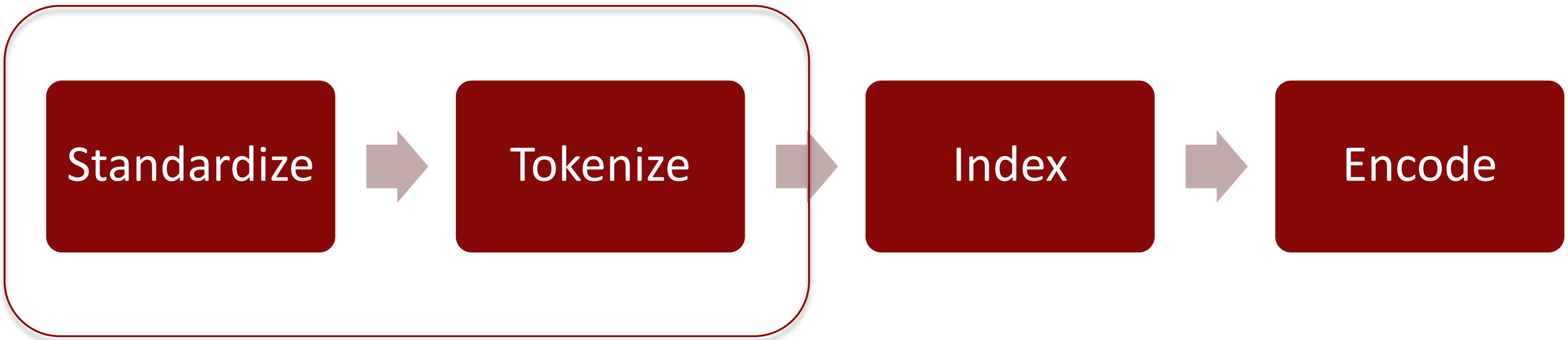
Processing Basics

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

This process is called *text vectorization*

Basic Pre-Processing



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

*aka "training corpus"

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Standardization

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

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

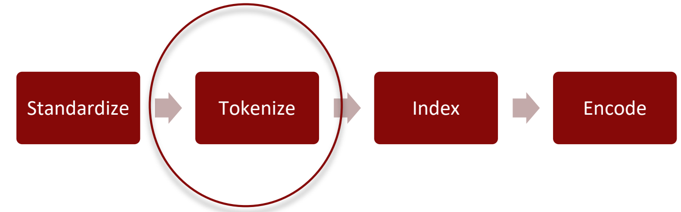
Standardization

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

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

*Modern LLMs use other tokenization schemes (more on this shortly)

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

Tokenization

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

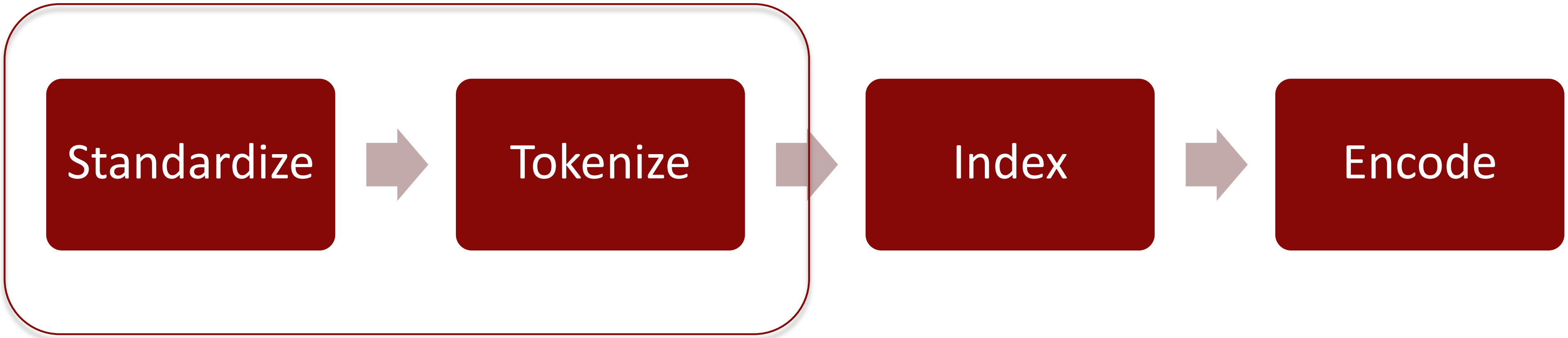
hola what do you picture when you [thinks] of [travels] to mexico [sips] real margarita while [soaks] up sun on laidback beach in puerto vallarta

⬇

“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”

The Standardization and Tokenization we have described is a good default for many NLP tasks but there are disadvantages, especially for text generation tasks. Modern LLMs use other schemes (e.g., Byte Pair Encoding) that we will describe later.

Basic Pre-Processing



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

Basic Pre-Processing

Standardize → Tokenize → Index → Encode

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

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

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

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

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

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

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 = # 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 = # 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

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

- Dimension of encoding vector = # 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

*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

*change to lowercase, strip punctuation, leave stop words as is, no stemming

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.

VOCABULARY

← 100 →

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

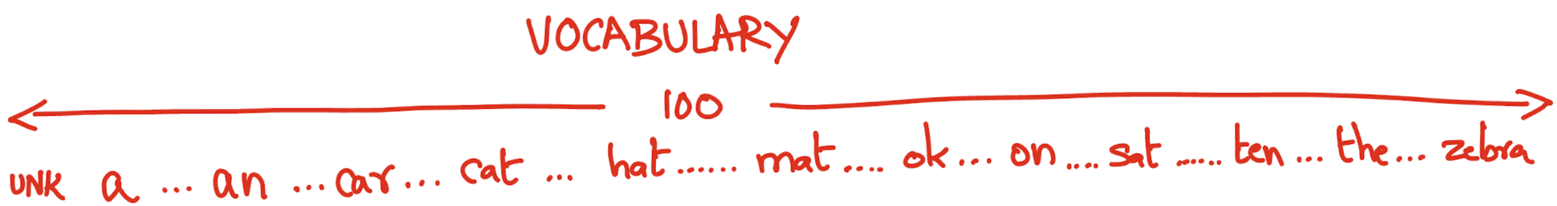
- This input text string arrives - "The cat sat on the mat" – and we run it through STIE

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

- 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.

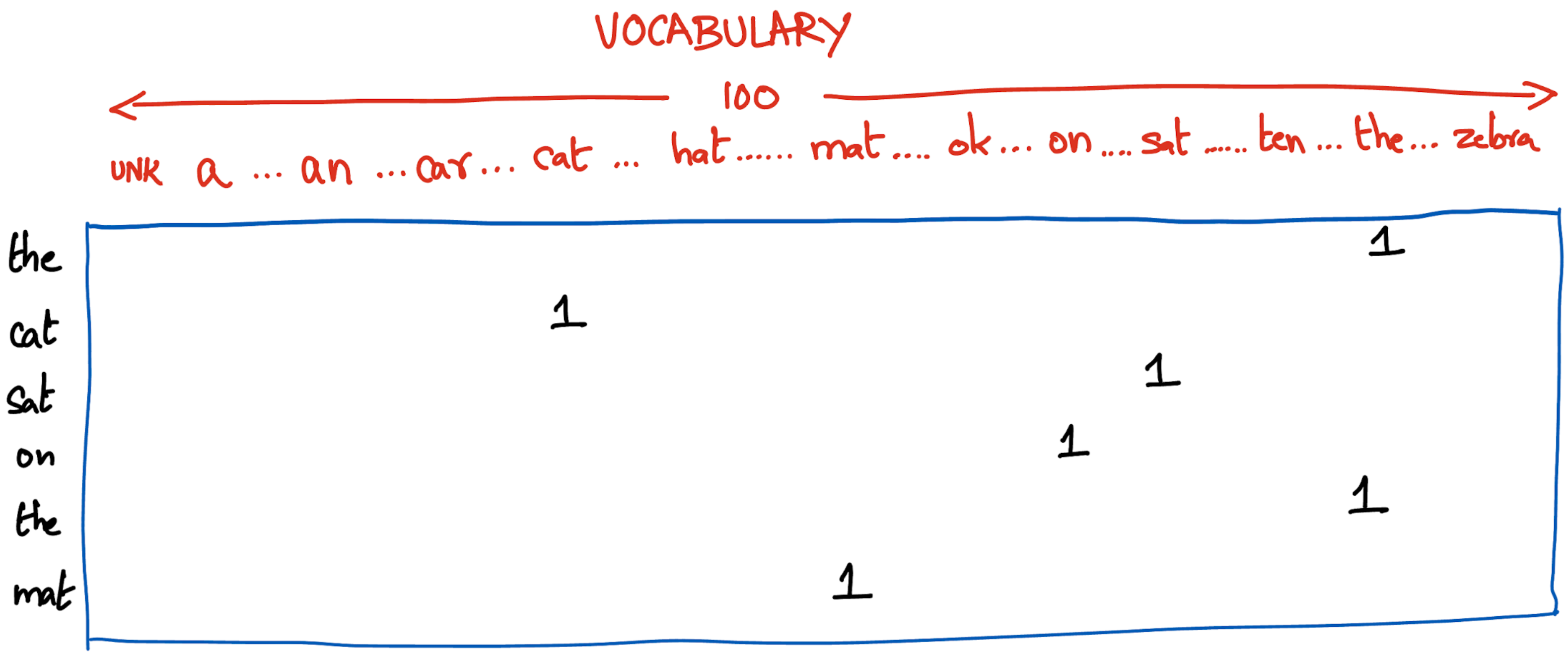


- This input text string arrives - "The cat sat on the mat" – and we run it through STIE

- The output is a 6 x 100 table.

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

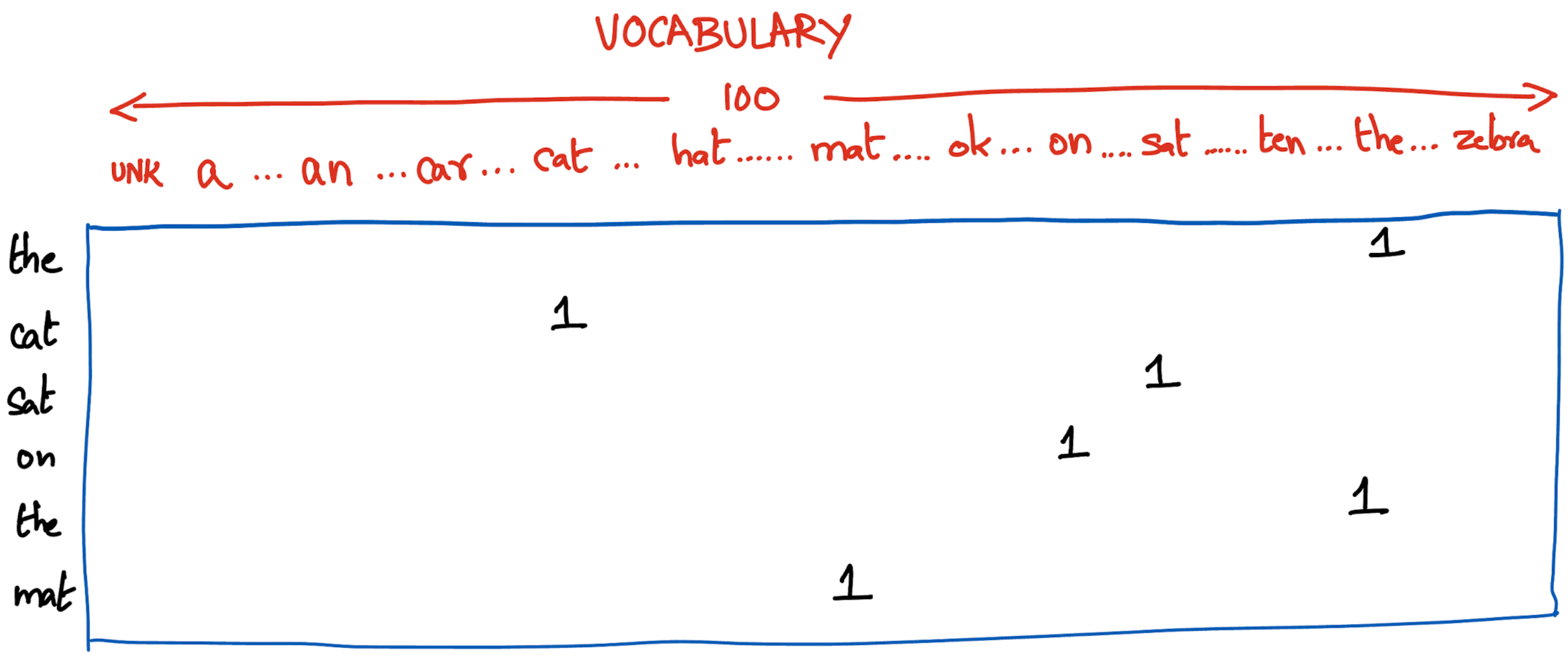
The output table*



*Not showing 0s to avoid clutter

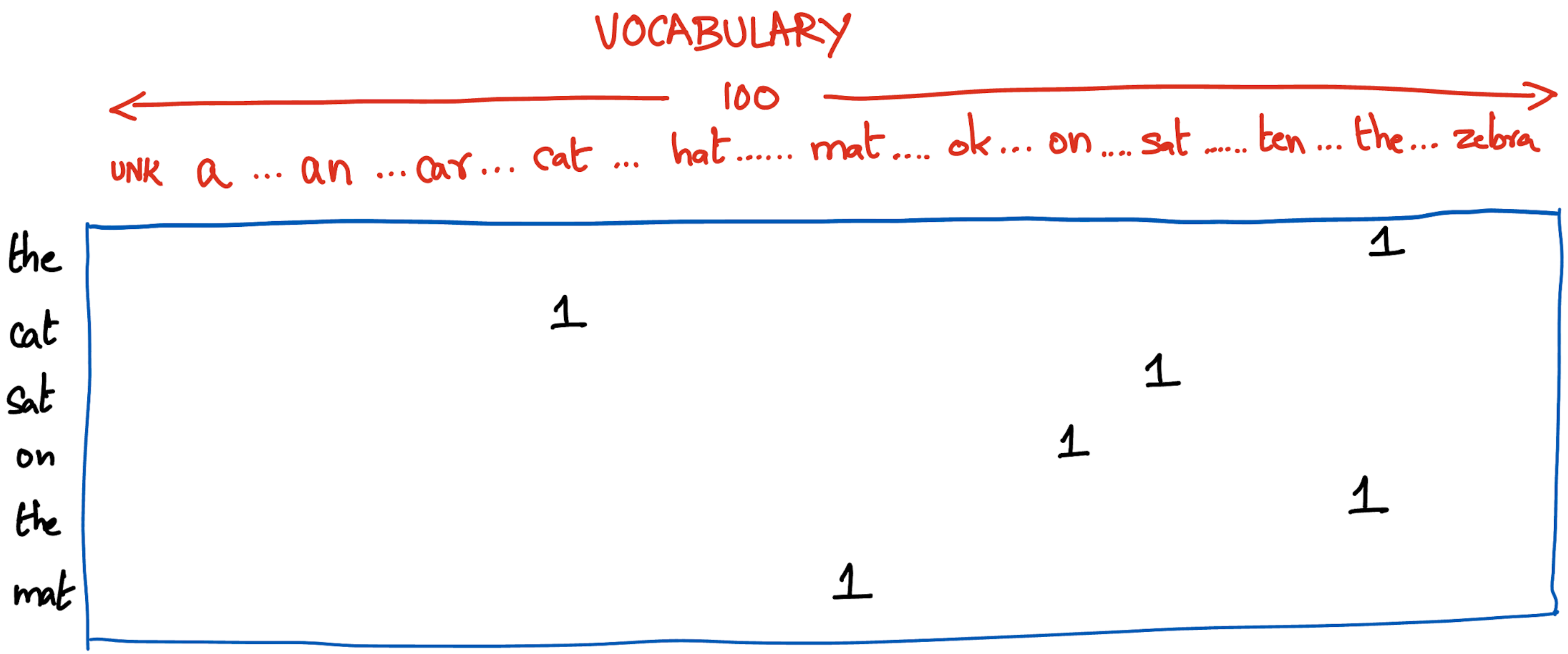
How to get a *new* input sentence ready to be “fed” into a DNN

- 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

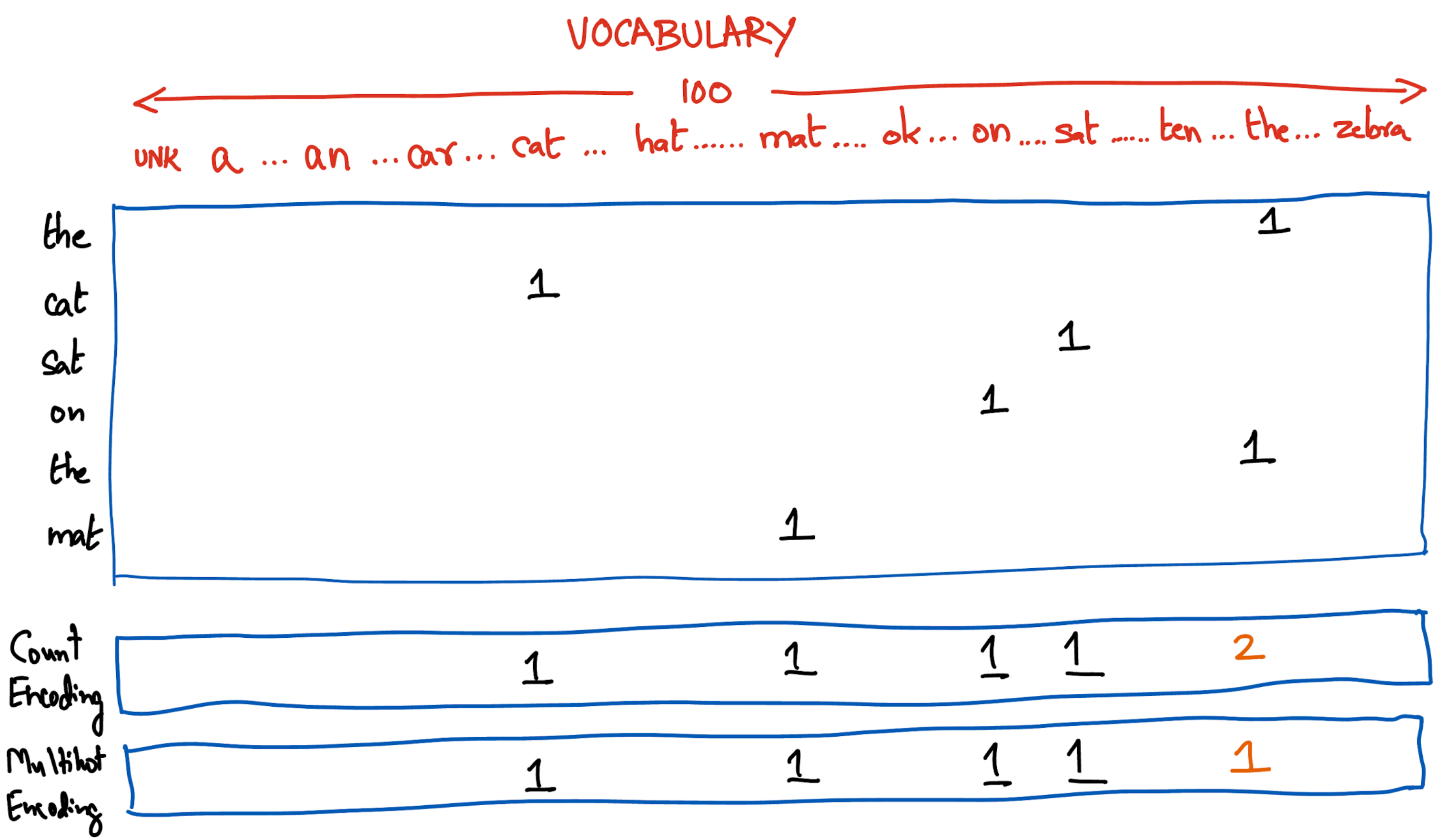
How to get a *new* input sentence ready to be “fed” into a DNN

- 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
- What if 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
- What if 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
- What if 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?

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”)

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.

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

Application: Genre Prediction

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

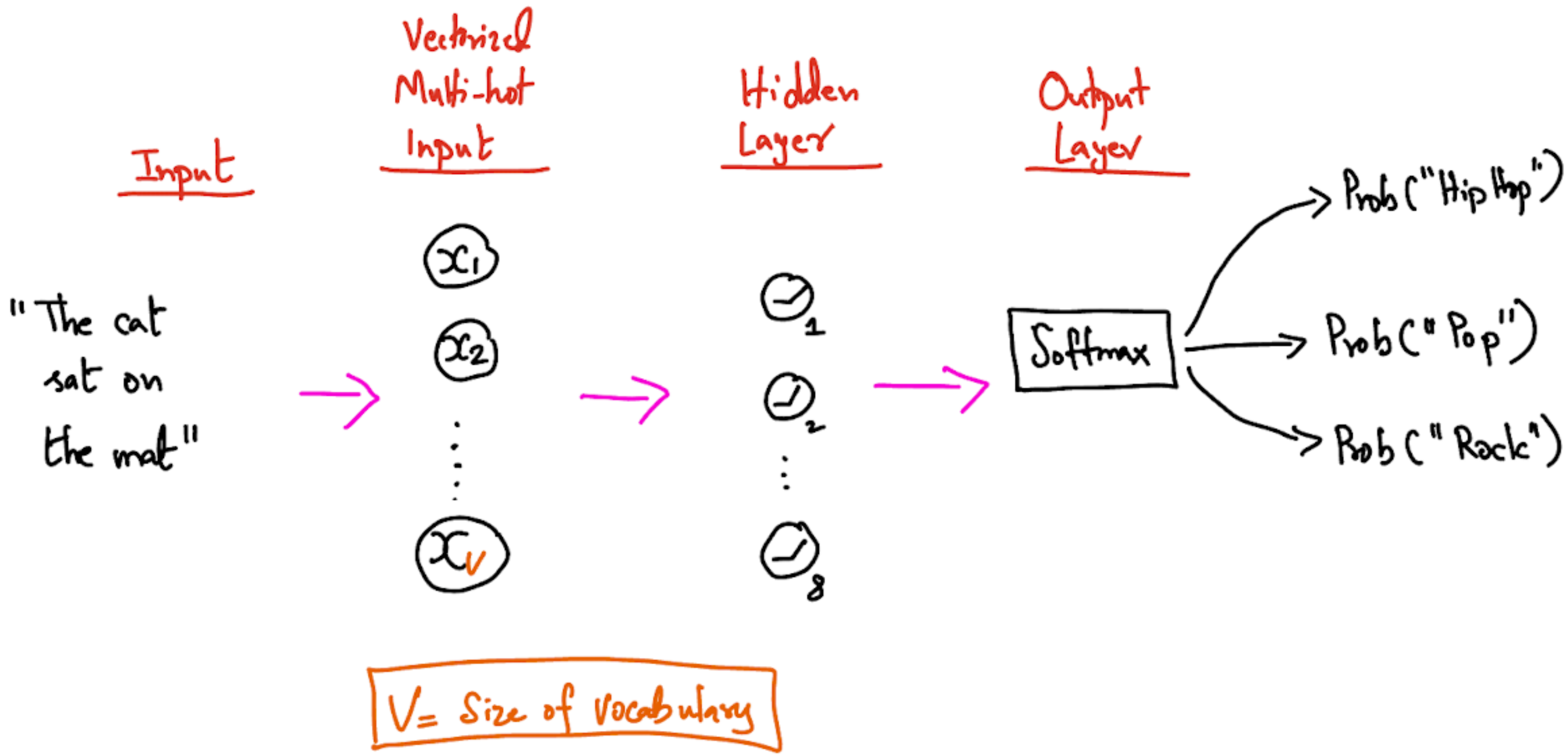
I walked through the door with you
The air was cold
But something about it felt like home somehow
And I, left my scarf there at your sisters house

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

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

Blackboard

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



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

[Link to Colab]