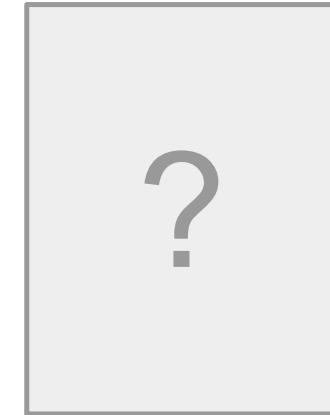
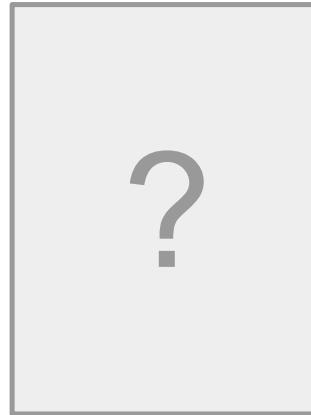




Natural Language Processing

Lecture 1: Introduction to the course.
NLP tasks and use cases.

Course Instructors

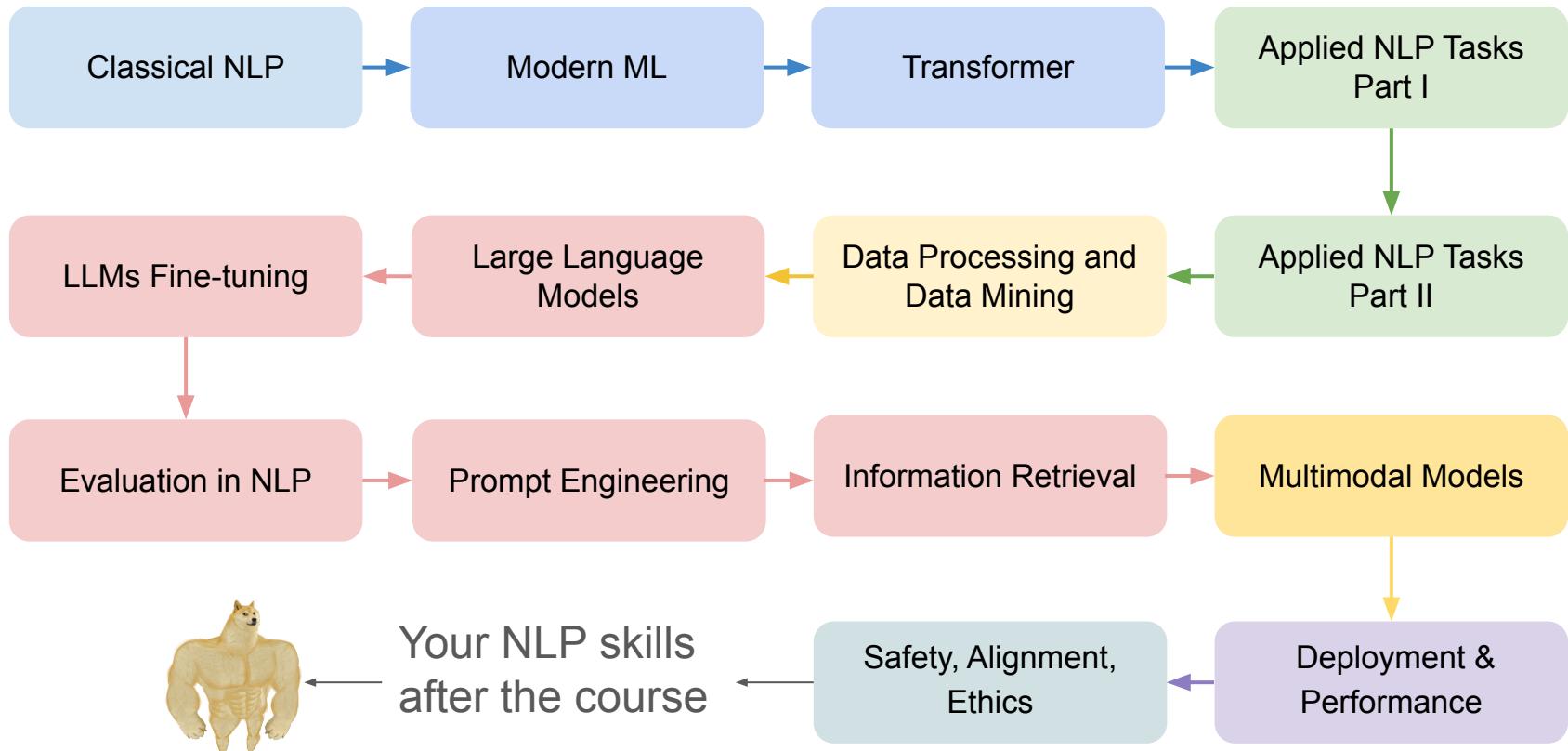


Viktoriia Makovska

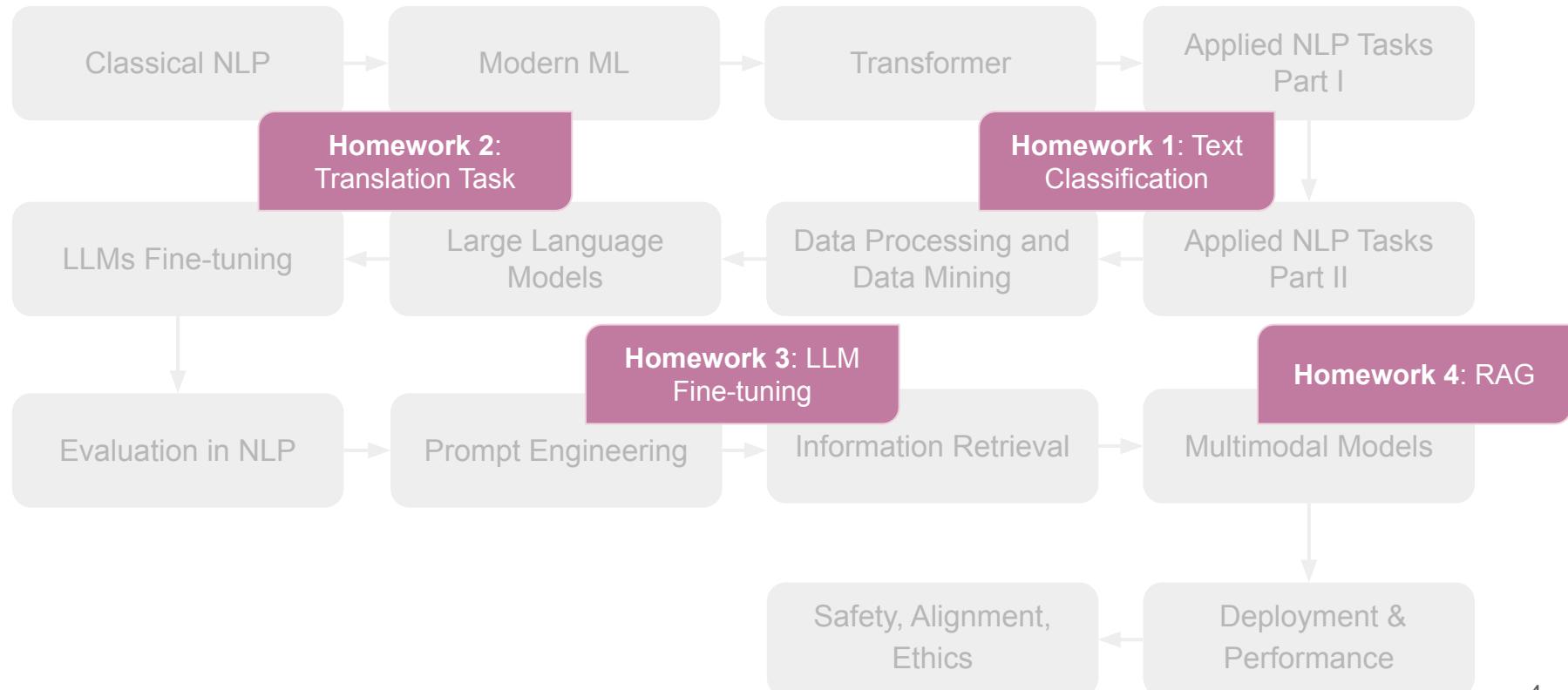
Research Fellow @ Center for RAI

7+ year in software engineering @ Artur'In

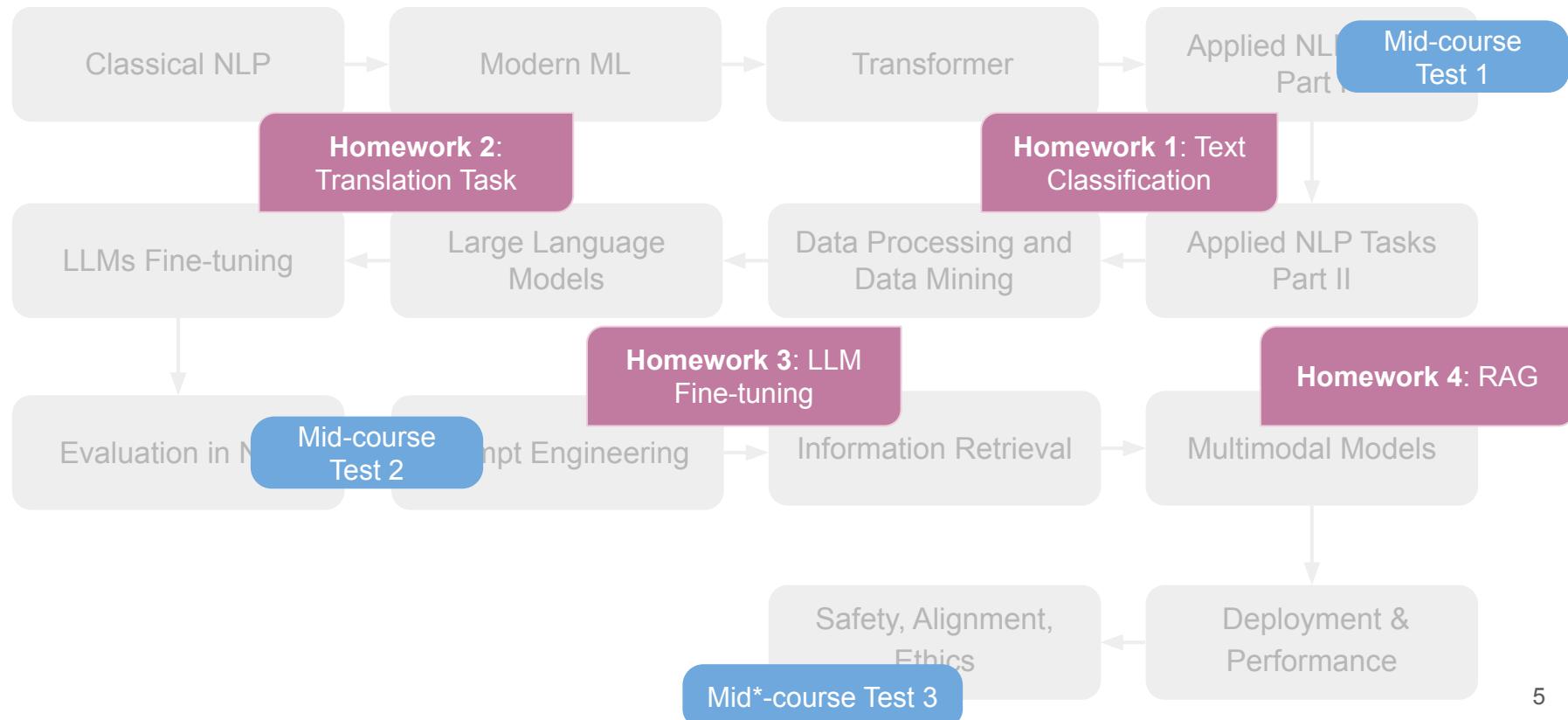
Course Structure



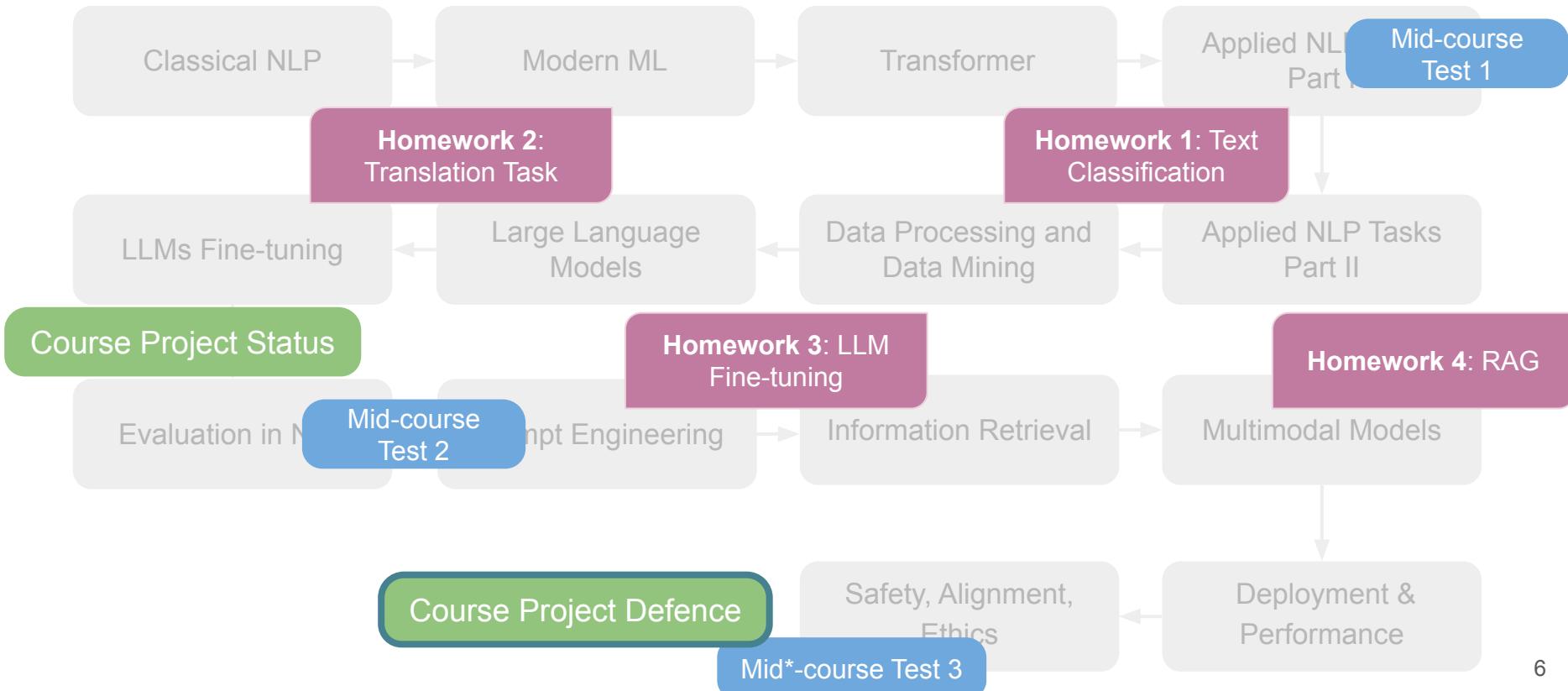
Course Structure: Milestones



Course Structure: Milestones



Course Structure: Milestones



Final Project

Option 1: Default project Topic
Option 2: Custom project Topic

Mid-report: **5 pts**

Teams by **2-3** people

Project defence: **25 pts**

Course Structure: Milestones

Homework 1	Text Classification	10
Homework 2	Translation Task	15
Homework 3	LLM Fine-tuning	15
Homework 4	Retrieval-Augmented Generation (RAG)	15
Online Test 1	Week 1 - 5	5
Online Test 2	Week 6 - 10	5
Online Test 3	Week 11 - 15	5
Course Project - Status Update		5
Course Project - Defence & Tech Report		25

100 in total

Schedule

Monday, Friday - 16:50 - 18:10

Sep. 1 → Sep. 6, 13:15 - 14:45
Sep. 8 → Sep. 13, 13:15 - 14:45

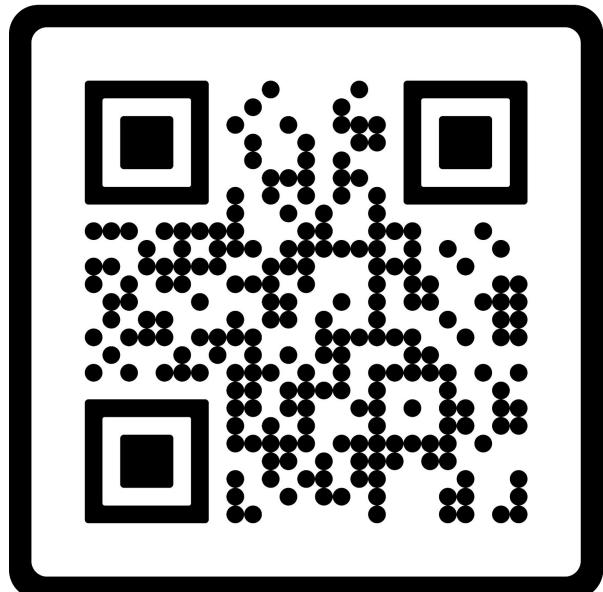
Nov. 21 → X

Office Hours:
By request Monday-Friday
after 17:00

Additional Resources

DataCamp Classroom

 Designing Agentic Systems with LangChain Course	Organization	Active	Fixed	Sep 03, 2025 13:48 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Introduction to AI Agents Course	Organization	Active	Fixed	Sep 03, 2025 13:48 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Retrieval Augmented Generation (RAG) with LangChain Course	Organization	Active	Fixed	Sep 03, 2025 13:46 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Introduction to LLMs in Python Course	Organization	Active	Fixed	Sep 03, 2025 13:45 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Large Language Models (LLMs) Concepts Course	Organization	Active	Fixed	Sep 03, 2025 13:44 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Advanced NLP with spaCy Course	Organization	Active	Fixed	Sep 03, 2025 13:43 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Feature Engineering for NLP in Python Course	Organization	Active	Fixed	Sep 03, 2025 13:43 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View
 Natural Language Processing (NLP) in Python Course	Organization	Active	Fixed	Sep 03, 2025 13:42 GMT+3	Dec 31, 2025 00:00 GMT+2	0 1	0%	View



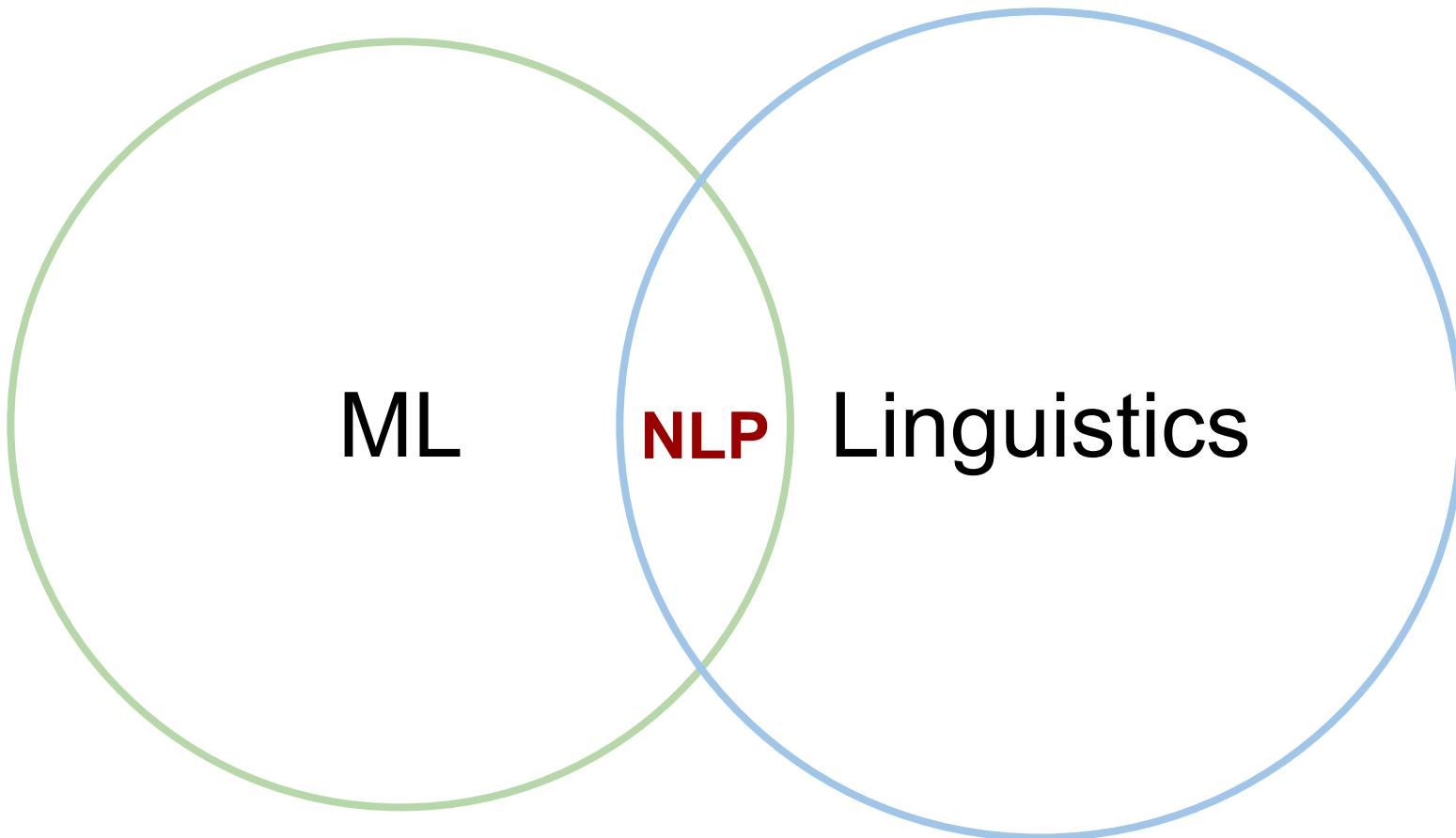
What is NLP

Common Tasks and Use Cases

ML Project Lifecycle

Text Cleaning with examples

What is NLP



Blocks of Language

Phonemes

- Speech to Text
- Speaker Identification
- Text to Speech

Morphemes

Lexemes

- Tokenization
- Word Embeddings
- POS Tagging

Syntax

- Parsing
- Named Entity Extraction
- Relation Extraction

Meaning

- Summarization
- Topic Modeling
- Sentiment Analysis

What is NLP

Block

Phonemes

- Speech to Text
- Speaker Identification
- Text to Speech

44 Phonemes of English

/b/	/a/	/k/	/ch/	/ē/	/f/	/h/	/j/	/ɪ/
ball rabbit crab	laugh fan apple	kite soccer back	chew catch statue	bee happy she	fish laugh photo	house hello hippo	juice fudge jug	lemon gull lion
/ā /	/ar/	/d/	/e/	/ir/	/g/	/i/	/ī/	/n/
day cake weigh	car shark barn	dog glad duck	egg friends bread	dirt burn better	go grapes ghost	pin build igloo	spider sky night	nice fun knife
/m/	/o/	/ō/	/oi/	/air/	/r/	/s/	/th/	/th/
me comb same	father hot odd	toes grow boot	coin toy royal	chair square there	robot cherry wrench	sun science seal	thorn bath thread	them weather the
/zh/	/ng/	/oo/	/ōō/	/p/	/u/	/sh/	/t/	/v/
treasure television division	sing ring long	book could wood	moon fruit glue	panda apple lap	mug flood nut	shoe ocean machine	tap ripped turtle	view wave of
/w/	/wh/	/ow/	/or/	/aw/	/y/	/z/	/ə/	/ear/
web watermelon quick	whistle what white	cow sprout shout	board fork door	saw caught soft	onion yellow yarn	zebra buzz xenon	balloon parrot pencil	steer ear here

What is NLP

Block

Phonemes

- Speech to Text
- Speaker Identification
- Text to Speech

44 Phonemes of English

/b/	/a/	/k/	/ch/	/ē/	/f/	/h/		
ball rabbit crab	laugh fan apple	kite soccer back	chew catch statue	bee happy she	fish laugh photo	house hello hippo	roo-goo	go-
/ā /	/ar/	/d/	/e/	/ir/	/g/	/i/	/ī/	/n/
day cake weigh	car shark barn	dog glad duck	egg friends bread	dirt burn better	go grapes ghost	pin build igloo	spider sky night	nice fun knife
/m/	/o/	/ō/	/oi/	/air/	/r/	/s/	/th/	/th/
me comb same	father hot odd	toes grow boot	coin toy royal	chair square there	robot cherry wrench	sun science seal	thorn bath thread	them weather the
/zh/	/ng/	/oo/	/ōō/	/p/	/u/	/sh/	/t/	/v/
treasure television division	sing ring long	book could wood	moon fruit glue	panda apple lap	mug flood nut	shoe ocean machine	tap ripped turtle	view wave of
/w/	/wh/	/ow/	/or/	/aw/	/y/	/z/	/ə/	/ear/
web watermelon quick	whistle what white	cow sprout shout	board fork door	saw caught soft	onion yellow yarn	zebra buzz xenon	balloon parrot pencil	steer ear here

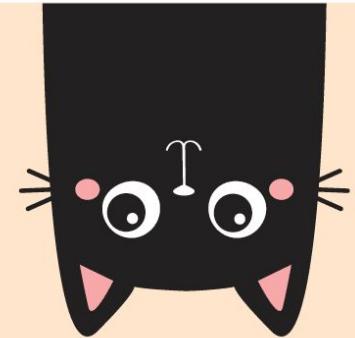
Speech to Text
Speaker Identification
Text to Speech

What is NLP



cat + s = cats

(morpheme 1) (morpheme 2)



What is NLP

Tokenization
Word Embeddings
POS Tagging

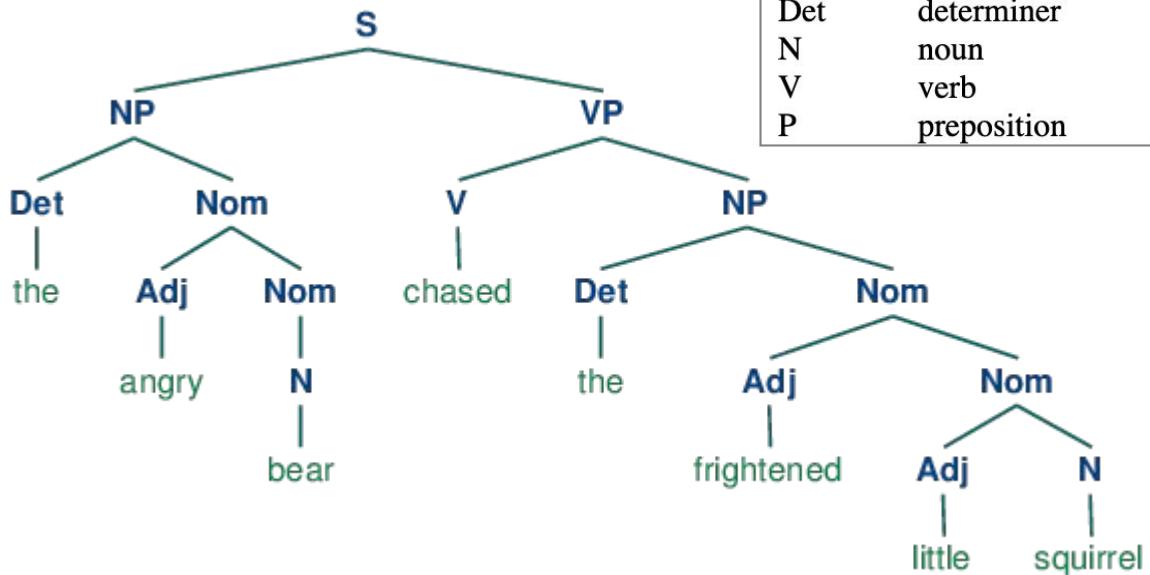


cat + s = cats

(morpheme 1) (morpheme 2)



What is NLP

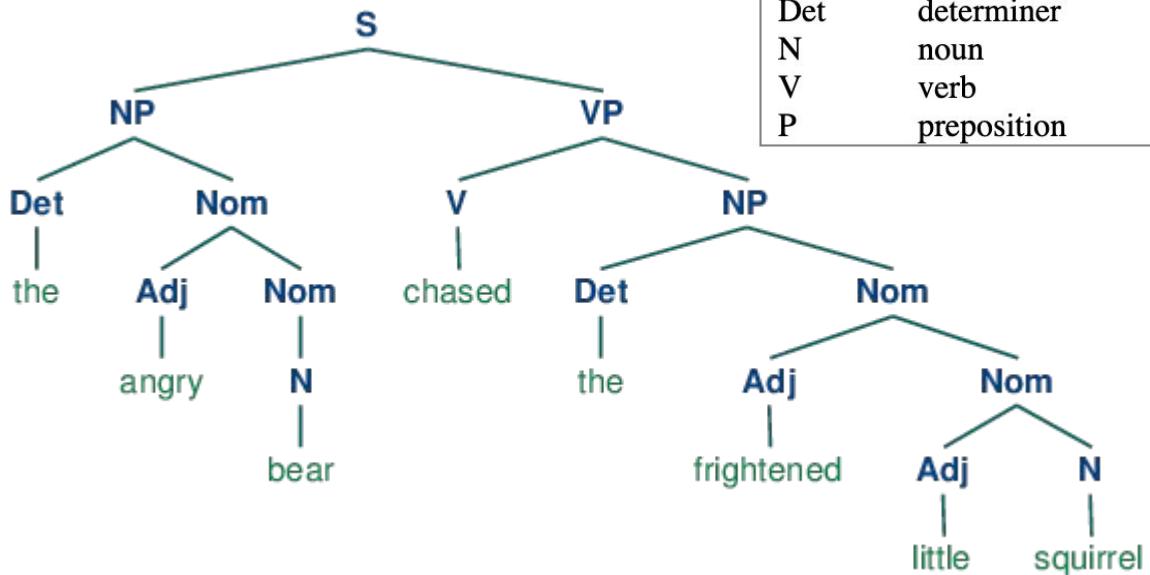


Symbol	Meaning	Example
S	sentence	<i>the man walked</i>
NP	noun phrase	<i>a dog</i>
VP	verb phrase	<i>saw a park</i>
PP	prepositional phrase	<i>with a telescope</i>
Det	determiner	<i>the</i>
N	noun	<i>dog</i>
V	verb	<i>walked</i>
P	preposition	<i>in</i>

Meaning

- Summarization
- Topic Modeling
- Sentiment Analysis

What is NLP



Symbol	Meaning
S	sentence
NP	noun phrase
VP	verb phrase
PP	prepositional phrase
Det	determiner
N	noun
V	verb
P	preposition

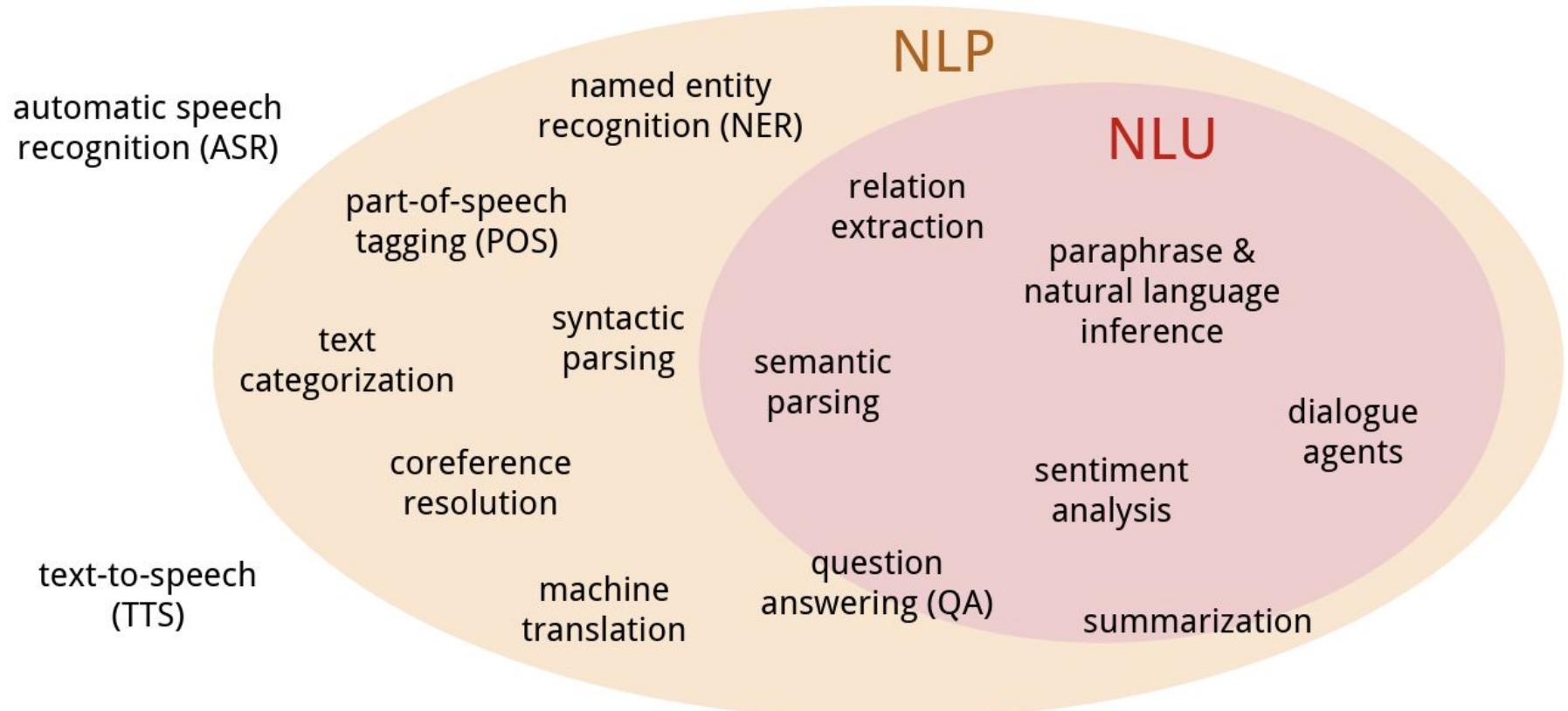
**Entity Extraction
Relation Extraction**

...
dog
walked
in

Meaning

- Summarization
- Topic Modeling
- Sentiment Analysis

Terminology: NLU vs. NLP vs. ASR



How it started (Conversational)

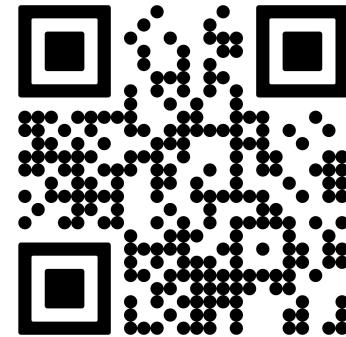
ELIZA

Talk to Eliza by typing your questions and answers in the input box.

```
> Hello, I am Eliza. I'll be your therapist today.  
* Hi. I feel sad.  
> Do you often feel sad?  
* Every day. Sometimes twice a day  
> Can you elaborate on that?  
* I feel sad 'cause I miss home  
> Do you enjoy feeling sad 'cause you miss home?  
* No! Of course not!  
> Are you sure?
```

TYPE HERE

Mid-1960s



ELIZA was one of the first *chatterbots* (later clipped to **chatbots**). It was also an early test case for the **Turing Test**.

<https://en.wikipedia.org/wiki/ELIZA>

How it started (Conversational)

ELIZA

Talk to Eliza by typing your questions and answers in the input box.

```
> Hello, I am Eliza. I'll be your therapist today.  
* Hi. I feel sad.  
> Do you often feel sad?  
* Every day. Sometimes twice a day  
> Can you elaborate on that?  
* I feel sad 'cause I miss home  
> Do you enjoy feeling sad 'cause you miss home?  
* No! Of course not!  
> Are you sure?
```

TYPE HERE



ELIZA was a computer program created by Joseph Weizenbaum in 1966. It was designed to simulate a Rogerian "therapist" and was one of the first chatterbots. ELIZA used pattern matching and substitution to generate responses. It became famous as a test case for the Turing Test.

What is Turing Test?

Mid-1960s

<https://en.wikipedia.org/wiki/ELIZA>

How it's going (Conversational)

ELIZA

Talk to Eliza by typing your questions and answers in the input box.

```
> Hello, I am Eliza. I'll be your therapist today.  
* Hi. I feel sad.  
> Do you often feel sad?  
* Every day. Sometimes twice a day  
> Can you elaborate on that?  
* I feel sad 'cause I miss home  
> Do you enjoy feeling sad 'cause you miss home?  
* No! Of course not!  
> Are you sure?
```

TYPE HERE

ChatGPT5

Hi. I feel sad.

I hear you. ❤️

Sometimes just saying "*I feel sad*" is already a brave and important first step.

If you'd like, we can try one of a few directions together:

- **Talk it out:** You can tell me what's on your mind, or what might have triggered the sadness.
 - **Shift gently:** I can share a grounding or calming exercise, like a quick breathing technique or a small mental reset.
 - **Comfort mode:** If you just want a distraction, I can tell you something light, interesting, or even funny.
- 👉 Do you feel like talking about what's making you sad, or would you prefer a gentle distraction right now?



Mid-1960s

2025



How it's going (Conversational)

ELIZA

Talk to Eliza by typing your questions and answers in the input box.

```
> Hello, I am Eliza. I'll be your therapist today.  
* Hi. I feel sad.  
> Do you often feel sad?  
* Every day. Sometimes twice a day  
> Can you elaborate on that?  
* I feel sad 'cause I miss home  
> Do you enjoy feeling sad 'cause you miss home?  
* No! Of course not!  
> Are you sure?
```

TYPE HERE

Gemini 2.5 Flash

Hi. I feel sad.

◆ I'm so sorry to hear that you're feeling sad. It's completely okay to feel this way, and I want you to know that your feelings are valid.

If you'd like to talk about what's on your mind, I'm here to listen without judgment. You can share as little or as much as you're comfortable with.

Sometimes, even just acknowledging these feelings can be a first step. Please take care of yourself.



Mid-1960s

2025



How we imagined it. How it was

HALL-9000



David Bowman: Open the pod bay doors, HAL.

HAL: I'm sorry, Dave, I'm afraid I can't do that.

David: What are you talking about, HAL?

HAL: I know that you and Frank were planning to disconnect me, and I'm afraid that's something I cannot allow to happen.

Siri (2011)



Colbert: ... I don't want to search for anything! I want to write the show!

Siri: Searching the Web for "search for anything. I want to write the shuffle."

Colbert: ... For the love of God, the cameras are on, give me something?

Siri: What kind of place are you looking for? Camera stores or churches?

2001: A Space Odyssey (1968)

Siri (2011)

Milestones of NLU/NLP

Early Concepts and Machine Translation

Alan Turing's 1950 paper

Guess the sentence

Claude Shannon's 1948 paper on information theory, providing the mathematical basis for encoding, transmitting, and decoding information signals.

Dn't psh th hrss.

Or

Dn7 psh th h0rs3s

Test.

on of information

theory, providing the mathematical basis for encoding, transmitting, and decoding information signals.

Rise of Statistical Methods (Statistical Evolution)

Statistical methods, such as n-grams, became valuable for analyzing large amounts of text data. (From rule-based to Statistical methods)

1950s

1960s-1970s

1980s-1990s

Deep Learning and Modern NLU

The introduction of recurrent neural network (RNN) models, like LSTMs (improved voice and text processing).

The Transformer architecture with self-attention mechanisms, significantly improved machine translation and other NLP tasks.

2000s

2017

Milestones of NLU/NLP

Early Concepts and Machine Translation

Alan Turing's 1950 paper: introduced the Turing Test.
Claude Shannon's 1948 work: laid the foundation of *information theory*, providing the mathematical basis for encoding, transmitting, and decoding information signals.

Dn't psh th hrss.

Or

Dn7 psh th h0rs3s

1950s

1960s-1970s

ered Systems and Early Chatbots

program in 1966, by Joseph Weizenbaum. The program simulated a human therapist) **0 bits** = completely predictable (always the same symbol).

Redundancy in language (**repeated patterns, grammar, common words**) reduces uncertainty.

So English has **low or high entropy**?

$$H(X) = - \sum_i p(x_i) \log_2 p(x_i)$$

Deep Learning and Modern NLU

The introduction of recurrent neural network (RNN) models, like LSTMs (improved voice and text processing).

2000s

2017

- **1 bit** = like a fair coin (two equally likely outcomes). This significantly improved machine translation and other NLP tasks.
- **>1 bit** = more than 2 equally likely choices.

Milestones of NLU/NLP

Early Concepts and Machine Translation

"Shannon game"

"Prediction and Entropy of Printed English", by C. E. Shannon (1951, *Bell System Technical Journal*, 30, pp. 50-64).

1950s

introduced the Turing Test.

Shannon laid the foundation of information theory: laid the foundation of information theory, provided a mathematical basis for encoding, decoding and compressing language as information signals.

1960s-1970s

Rule-Based Systems and Early NLU

The ELIZA program in 1966, by Joseph Weizenbaum (chatbot simulated a human therapist).

Deep Learning and Modern NLU

Of a total of 129 letters, 89 or 69% were guessed correctly. The errors, as would be expected, occur most frequently at the beginning of words and syllables where the line of thought has more possibility of branching out. It might be thought that the second line in (8), which we will call the *reduced context*, contains much less information than the first. Actually, both lines con-

2017

3. PREDICTION OF ENGLISH

The new method of estimating entropy exploits the fact that anyone speaking a language possesses, implicitly, an enormous knowledge of the statistics of the language. Familiarity with the words, idioms, clichés and grammar enables him to fill in missing or incorrect letters in proof-reading, or to complete an unfinished phrase in conversation. An experimental demonstration of the extent to which English is predictable can be given as follows: Select a short passage unfamiliar to the person who is to do the predicting. He is then asked to guess the first letter in the passage. If the guess is correct he is so informed, and proceeds to guess the second letter. If not, he is told the correct first letter and proceeds to his next guess. This is continued through the text. As the experiment progresses, the subject writes down the correct text up to the current point for use in predicting future letters. The result of a typical experiment of this type is given below. Spaces were included as an additional letter, making a 27 letter alphabet. The first line is the original text; the second line contains a dash for each letter correctly guessed. In the case of incorrect guesses the correct letter is copied in the second line.

- | | |
|---|-----|
| (1) THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG | (8) |
| (2) -----ROO-----NOT-V-----I-----SM---OBL--- | |
| (1) READING LAMP ON THE DESK SHED GLOW ON | |
| (2) REA-----O-----D---SHED-GLO--O-- | |
| (1) POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET | |
| (2) P-L-S-----O---BU--L-S--O-----SH-----RE--C----- | |

Milestones of NLU/NLP

Early Concepts and Machine Translation

[Alan Turing's 1950 paper](#): introduced the Turing Test (imitation game).

[Claude Shannon's 1948 work](#): laid the foundation of *information theory*, providing the mathematical basis for encoding, transmitting, and analyzing language as information signals.



1950s

1960s-1970s

1980s-1990s

Rule-Based Systems and Early Chatbots

The ELIZA program in 1966, by [Joseph Weizenbaum](#),
(chatbot simulated a human therapist)

Deep Learning and Modern NLU

The introduction of recurrent neural network (RNN) models, like LSTMs (improved voice and text processing).

The Transformer architecture with [self-attention mechanisms](#), significantly improved machine translation and other NLP tasks.

2000s

2017

Use Cases

Automated Trading, Decision Support

Some trading strategies rely in part on automated analysis of unstructured data feeds: **news**, analyst **reports**, etc.

Imagine you are designing an app to spot some trends in news about stock market related companies.

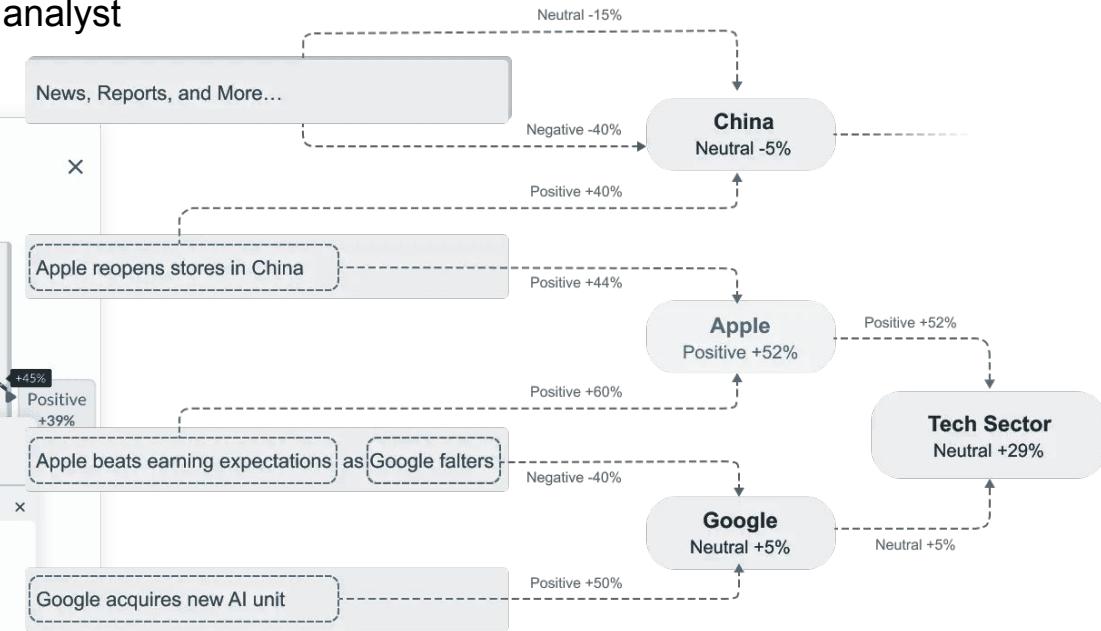
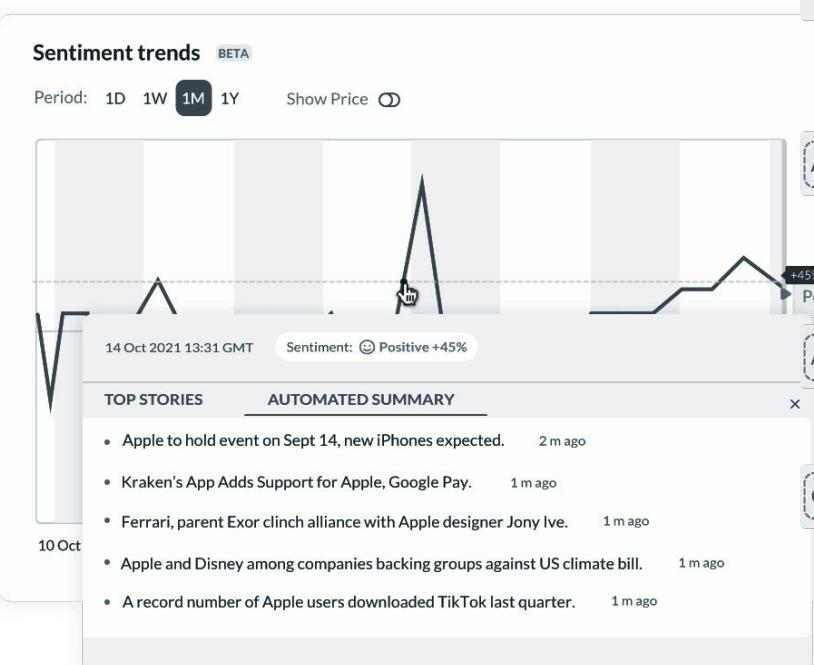
How would you do it?

Use Cases

Automated Trading, Decision Support

Some trading strategies rely in part on automated analysis of unstructured data feeds: **news**, **analyst reports**, etc.

***cityFALCON**



*Real scores use weighed averages.

Use Cases

Automated Trading, Decision Support

KIM ZETTER

SECURITY SEP 8, 2008 3:58 PM

Six-Year-Old News Story Causes United Airlines Stock to Plummet -- UPDATE Google Placed Wrong Date on Story

I'm surprised this hasn't happened before now. This isn't a story about security (although it is about securities), but it's so remarkable I thought I'd include it here anyway. A worker at a Miami investment advisory firm called Income Securities Advisors, which publishes news alerts that get distributed through the Bloomberg News Service, did a [...]

<https://www.wired.com/2008/09/six-year-old-st/>

Use Cases

Automated Trading, Decision Support

UPDATE 2: The story gets more convoluted and more interesting. According to a follow-up investigation, the article in the *Sun Sentinel's* archive had no date on it. But when Google's spider grabbed it, it assigned a current date to the piece, which then resulted in the article being placed in the top results of Google News. When the employee from Income Securities Advisors ran a Google search on "2008 bankruptcies," the old United Airlines story appeared as the top link in the results, with a September 6, 2008 date on it. (Google has now released a screenshot that shows the UAL story as it appeared on the Sun Sentinel web site. The only date in the screenshot is September 7, 2008, the date Google accessed the page. There is no date under the story's headline to indicate when it was published.)

Six-Year-Old News Story Placed Wrong Date on

I'm surprised this hasn't happened before. I include it here anyway. A worker at a Major news organization in the US distributed through the Bloomberg New

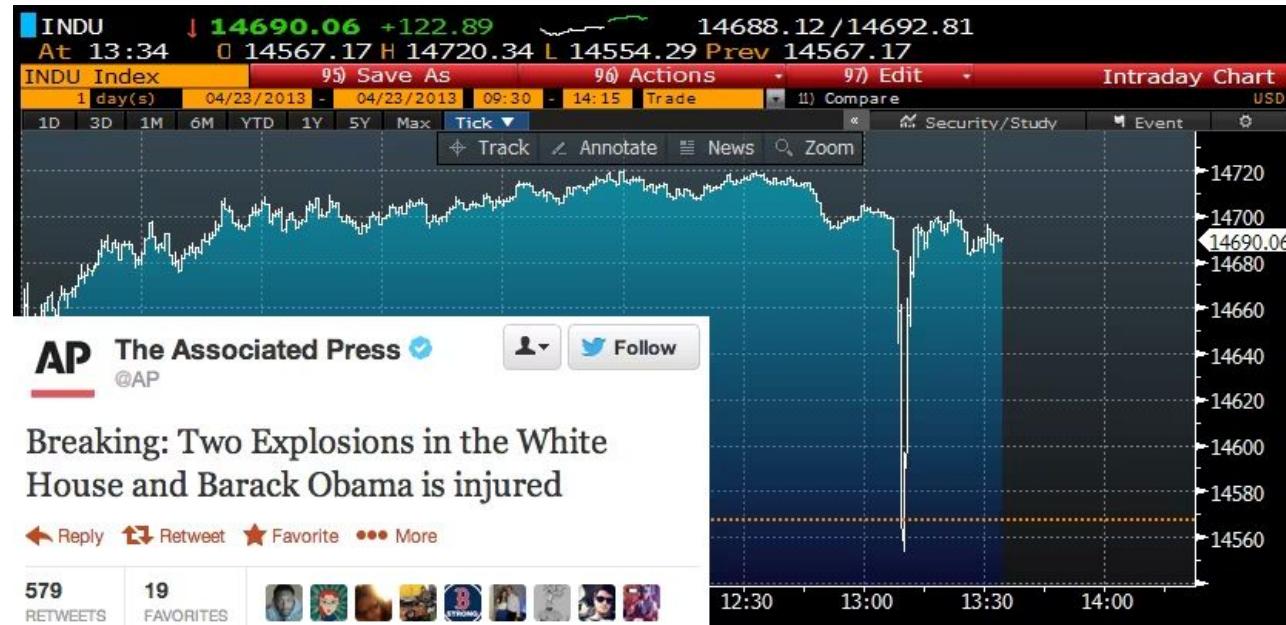
At 11 am Monday, the employee added the story to a feed that is included in a Bloomberg subscription service and within minutes, 15 million shares of United Airlines stock had been sold before trading on the stock was halted.

<https://www.wired.com/2008/09/six-year-old-story-placed-wrong-date-on>

Use Cases

The 2013 @AP Twitter hack

S&P 500 temporarily lost \$136B in market cap!



<https://www.cnbc.com/2013/04/23/false-rumor-of-explosion-at-white-house-causes-stocks-to-briefly-plunge-ap-confirms-its-twitter-feed-was-hacked.html>

Use Cases

Microsoft launched **Tay** in 2016.

It was supposed to tweet like a **teen** and learn from conversations with real users on Twitter.

The bot adopted phrases from users who tweeted **sexist** and **racist** comments, and Microsoft deactivated it not long afterward.

Tay illustrates some points made by the "**Stochastic Parrots**" paper, particularly the danger of not debiasing data.



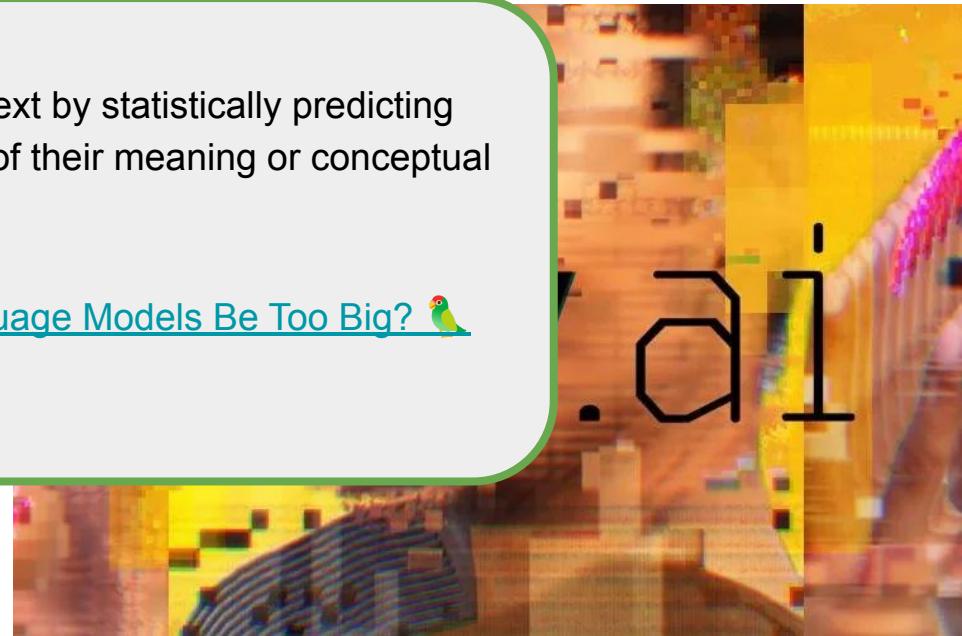
Use Cases

Stochastic Parrots - LLMs that mimic human-like text by statistically predicting word sequences without genuine understanding of their meaning or conceptual implications.

[On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜](#)

[Authors: Emily M. Bender, et al.](#)

Tay illustrates some points made by the “**Stochastic Parrots**” paper, particularly the danger of not debiasing data.



601 real-world
Google Cloud
AI use cases

Google
Cloud
Next 25

601 real-world
Google Cloud
AI use cases

Your cases?
(GenAI, Agents, Sentiment analysis, NER, etc.)

Go
Cloud
Next 25

Challenge: Linguistic variation

як пройти на вокзал

що найшвидше доїхати до вокзалу

як доїхати до вокзалу

маршрут на центральний вокзал

мені треба на залізничний вокзал

покажи як дістатися на вокзал метро

покажи маршрут до вокзалу

пішки до вокзалу

маршрут до центрального возалу

який транспорт їде до вокзалу

як краще доїхати до вокзалу зараз

навігація вокзал Львів

які автобуси йдуть до вокзалу

підкажи дорогу до вокзалу

де знаходиться залізничний вокзал

Challenge: Internationalization

bike gct

grand central navigation

station

train station directions

bici GCT

私は、鉄道駅に取得する必要があります

directions à Grand Central

Which train should I take to get
to Grand Central Terminal?

How do I get to Grand Central?

how to get to grand central by subway

Bahnhof Richtungen

take me to grand central

tell me how to go to the grand central

navegar grand central

التي القطار لجراند سنترال

best route gct

grand central navigation

directions to grand central

步行到中央火车站

*what's the best way to walk
to grand central from here*

walk to grand central

які автобуси йдуть до вокзалу

підкажи дорогу до вокзалу

Challenge: Context

The sofa doesn't fit the door because it is too narrow.
The sofa doesn't fit the door because it is too wide.

Challenge: Ambiguity

"italian reservation in palo alto"

- **Interpretation:** *Book a table at an Italian restaurant in Palo Alto.*
`>>> $Cuisine reservation in $Location`

"indian reservation in montana"

- **Interpretation:** *A Native American reservation located in Montana.*

"mission bike directions"

- **Interpretation:** *Directions to Mission District by bike.*
`>>> $Location $TransportationMode directions`

"mission bicycle directions"

- **Interpretation:** *Directions to a bicycle shop called "Mission Bicycles."*

Challenge: Ambiguity

"italian reservation in palo alto"

- **Interpretation:** Book a table at an Italian restaurant in Palo Alto.
 >>> \$Cuisine reservation in \$Location

"indian reservation in montana"

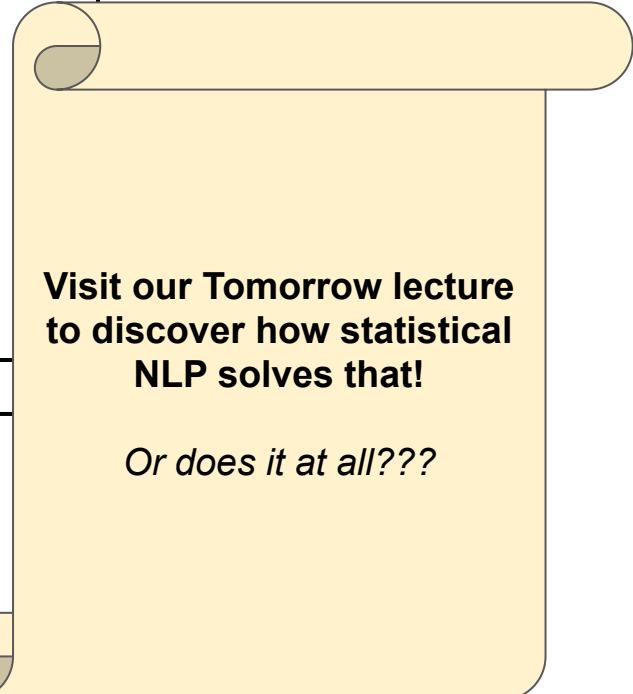
- **Interpretation:** A Native American reservation located in Montana.

"mission bike directions"

- **Interpretation:** Directions to Mission District by bike.
 >>> \$Location \$TransportationMode directions

"mission bicycle directions"

- **Interpretation:** Directions to a bicycle shop called "Mission Bicycles."



Visit our Tomorrow lecture
to discover how statistical
NLP solves that!

Or does it at all???



What are the components of the NLP life cycle?

Lifecycle

Anything else?

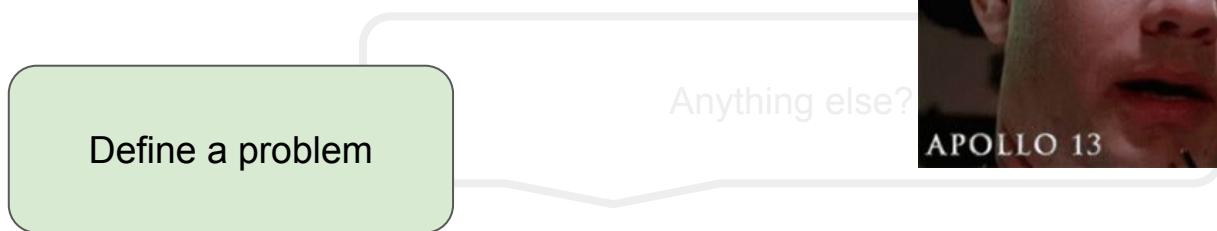
Collect — Clean — Model — Evaluate — Deploy — Monitor

Lifecycle



Collect — Clean — Model — Evaluate — Deploy — Monitor

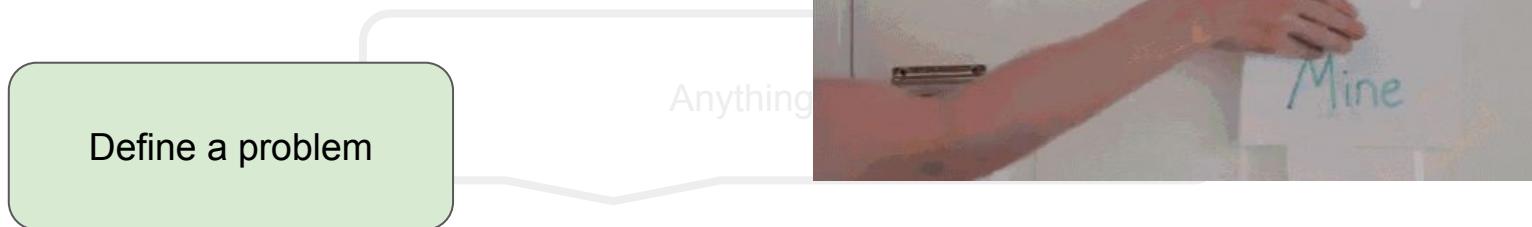
Lifecycle



Anything else?

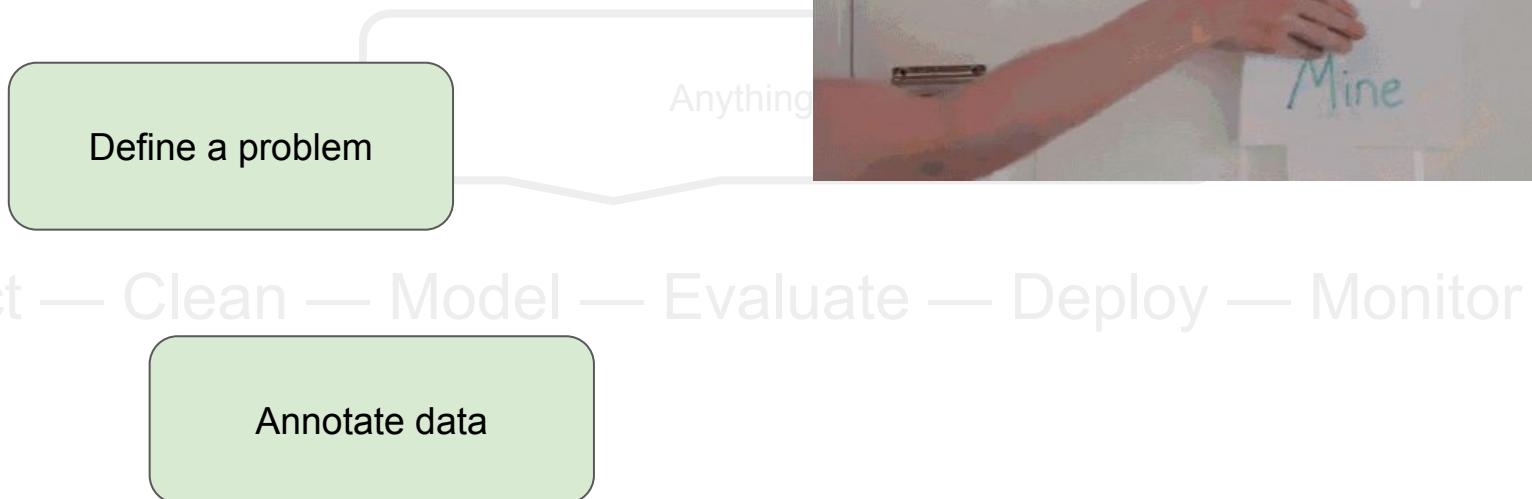
Collect — Clean — Model — Evaluate — Deploy — Monitor

Lifecycle

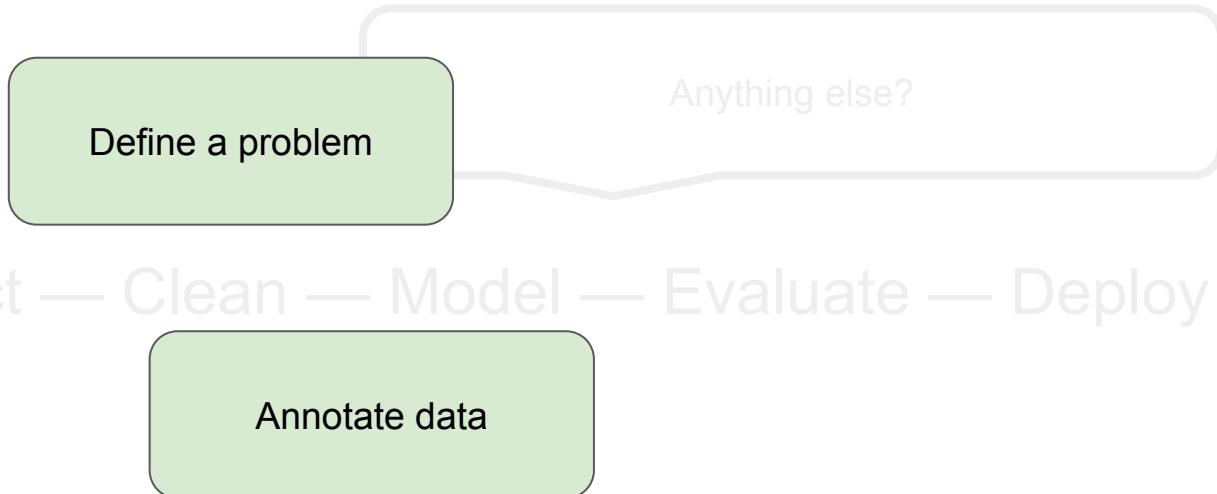


Collect — Clean — Model — Evaluate — Deploy — Monitor

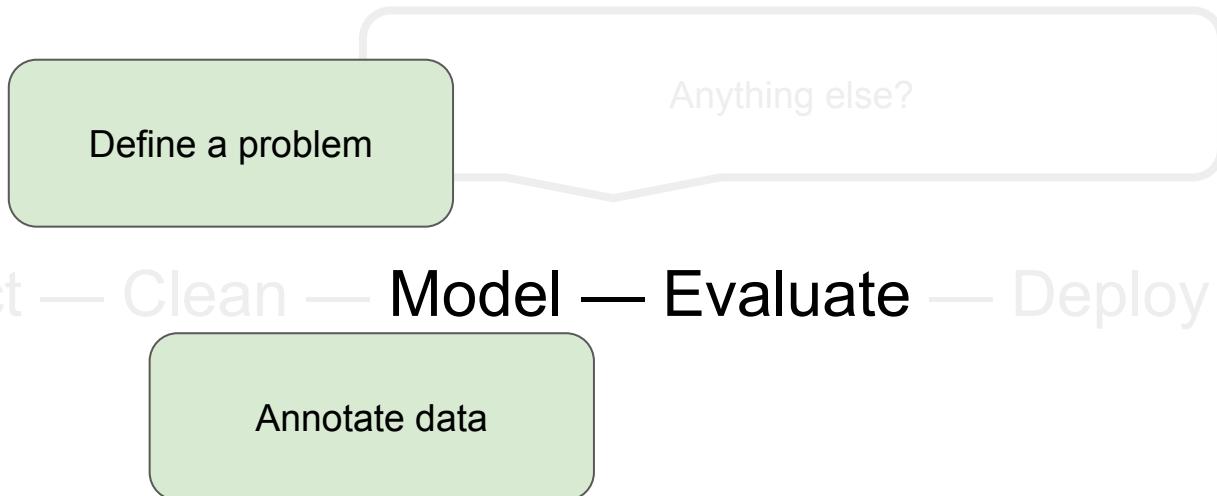
Lifecycle



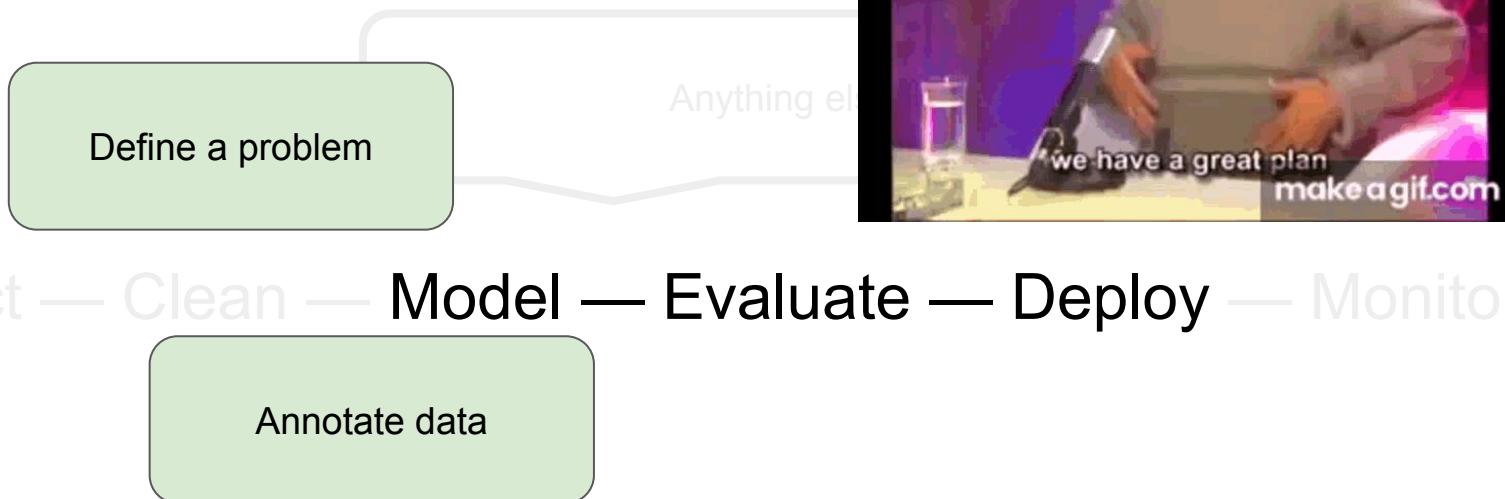
Lifecycle



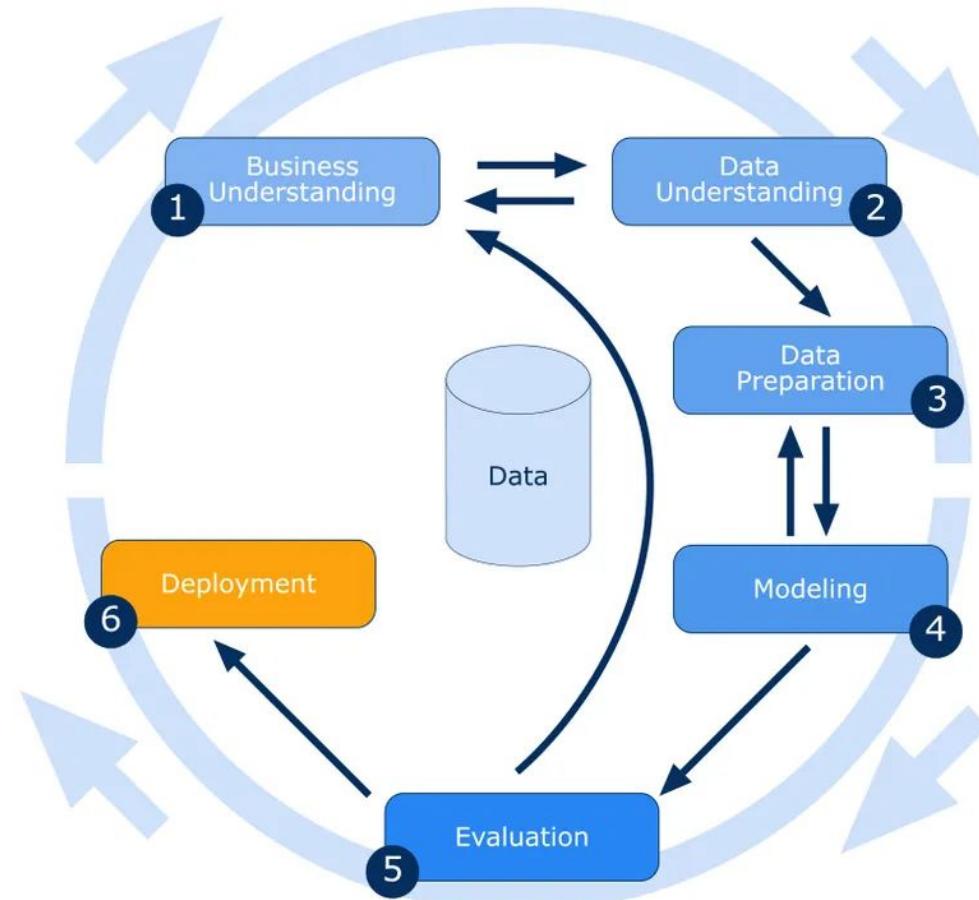
Lifecycle



Lifecycle



Lifecycle: CRISP-DM



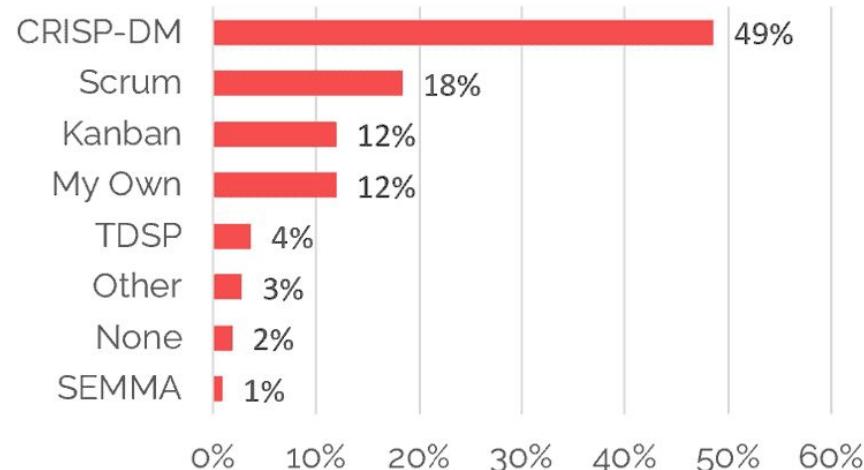
Lifecycle: CRISP-DM

CRISP-DM is Still the Most Popular Framework for Executing Data Science Projects

LAST UPDATED NOV 18, 2024 | LIFE CYCLE

datascience-pm.com Poll Results

Which process do you most commonly use for data science projects?



Text Cleaning

duplicates, typos, code-switching, emojis

Text Cleaning

Anything else?

duplicates, typos, code-switching, emojis

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- **Stemming**
- **Lemmatization**
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

The idea is to convert the input text into same casing format so that 'text', 'Text' and 'TEXT' are treated the same way.

helps to combine the same words together, reducing the duplication and get correct counts

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

The idea is to convert the input text into same casing format so that 'text', 'Text' and 'TEXT' are treated the same way.

Do we always need it?

helps to combine the same words together, reducing the duplication and get correct counts

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

to treat 'hurray' and 'hurray!'
in the same way

```
string.punctuation in python:  
! "#$%&\`()>*+, - ./ : ; <=>?@[ \\ ]^_-{| }~`
```

You can remove or add more as needed.

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

Words with little semantic meaning but usually are frequently used.

The, is, in, and, to,...

- Stopwords introduce noise
- When removed, we focus on meaningful words
- Stopwords adds tokens – it increases size of dataset

*In cases like Part of Speech tagging, we should not remove them as stopwords may provide very valuable information about the POS

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

The stopword lists are already compiled for different languages and we can safely use them
(Or create our own lists)



```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

The stopword lists are already compiled for different languages and we can safely use them
(Or create our own lists)

nltk?



```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of numbers
- Removal of punctuation marks
- Chat words
- Spelling correction

The stopword lists are already compiled for different languages and we can safely use them
(Or create our own lists)

nltk?



```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

[nltk](#) - natural language toolkit

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

if we have a domain specific corpus, we might also have some frequent words which are of not so important for us

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- **Stemming**
- **Lemmatization**
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

For example, if there are two words in the corpus **walks** and **walking**, then **stemming** will stem the suffix to make them **walk**.

But if we have two words **console** and **consoling**, the stemmer will remove the suffix and make them **consol** which is not a proper english word.

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- **Stemming**
- **Lemmatization**
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

For example, if there are two words in the corpus **walks** and **walking**, then **stemming** will stem the suffix to make them **walk**.

But if we have two words **console** and **consoling**, the stemmer will remove the suffix and make them **consol** which is not a proper english word.

Lemmatization does a similar thing but makes sure the root word (also called as **lemma**) belongs to the language.

Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- **Stemming**
- **Lemmatization**
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

	Stemming	Lemmatization
walks	walk	walk
walking		
walked		
studies	studi	study
studying		
studied		
ran	ran	run
feet	foot	foot

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- **Removal of emojis**
- **Removal of emoticons**
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

`:-)` is an emoticon

 is an emoji

How? Define a **regular expression pattern** that matches characters in certain **Unicode ranges**.

`\U0001F600-\U0001F64F` → 😊 😃 😄 (emoticons)

`\U0001F300-\U0001F5FF` → ⚽ 🌎 🎅 (symbols & pictographs)

`\U0001F680-\U0001F6FF` → 🚶 🚗 ✈ (transport & maps)

`\U0001F1E0-\U0001F1FF` → 🇺🇸 🇬🇧 🇺🇦 (flags)

And replaces all matched emoji characters with an empty string.

Text Cleaning

- Lower casing
- Removal of Punctuation
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

```
✓ [105] def convert_emoticons(text):
    for emot in EMOTICONS:
        text = re.sub(u'('+emot+')', "_".join(EMOTICONS[emot].replace(",","").split()), text)
    return text

text = "Hello :-) :)"
convert_emoticons(text)

→ 'Hello Happy_face_smiley Happy_face_smiley'

✓ [106] text = "I am sad :("
convert_emoticons(text)

→ 'I am sad Frown_sad_andry_or_poutingConfusion'
```

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction



Text Cleaning

- Lower casing
- Removal of Punctuations
- **Removal of Stopwords**
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction



Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

AFK=Away From Keyboard

ASAP=As Soon As Possible

GG=Good Game

GGWP=Good Game Well Played

IMO=In My Opinion

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

```
✓ [120] from spellchecker import SpellChecker  
os  
  
spell = SpellChecker()  
def correct_spellings(text):  
    corrected_text = []  
    misspelled_words = spell.unknown(text.split())  
    for word in text.split():  
        if word in misspelled_words:  
            corrected_text.append(spell.correction(word))  
        else:  
            corrected_text.append(word)  
    return " ".join(corrected_text)  
  
text = "speling correctin"  
correct_spellings(text)  
  
⇒ 'spelling correcting'  
  
✓ [121] text = "thnks for making it this faar, you did greatt job"  
correct_spellings(text)  
  
⇒ 'thanks for making it this far you did great job'
```

Text Cleaning

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization
- Removal of emojis
- Removal of emoticons
- Conversion of emoticons to words
- Conversion of emojis to words
- Removal of URLs
- Removal of HTML tags
- Chat words conversion
- Spelling correction

Examples in code with comments.

*copy this notebook before
making changes

Kaggle Twitter dataset for
the examples above



Q&A