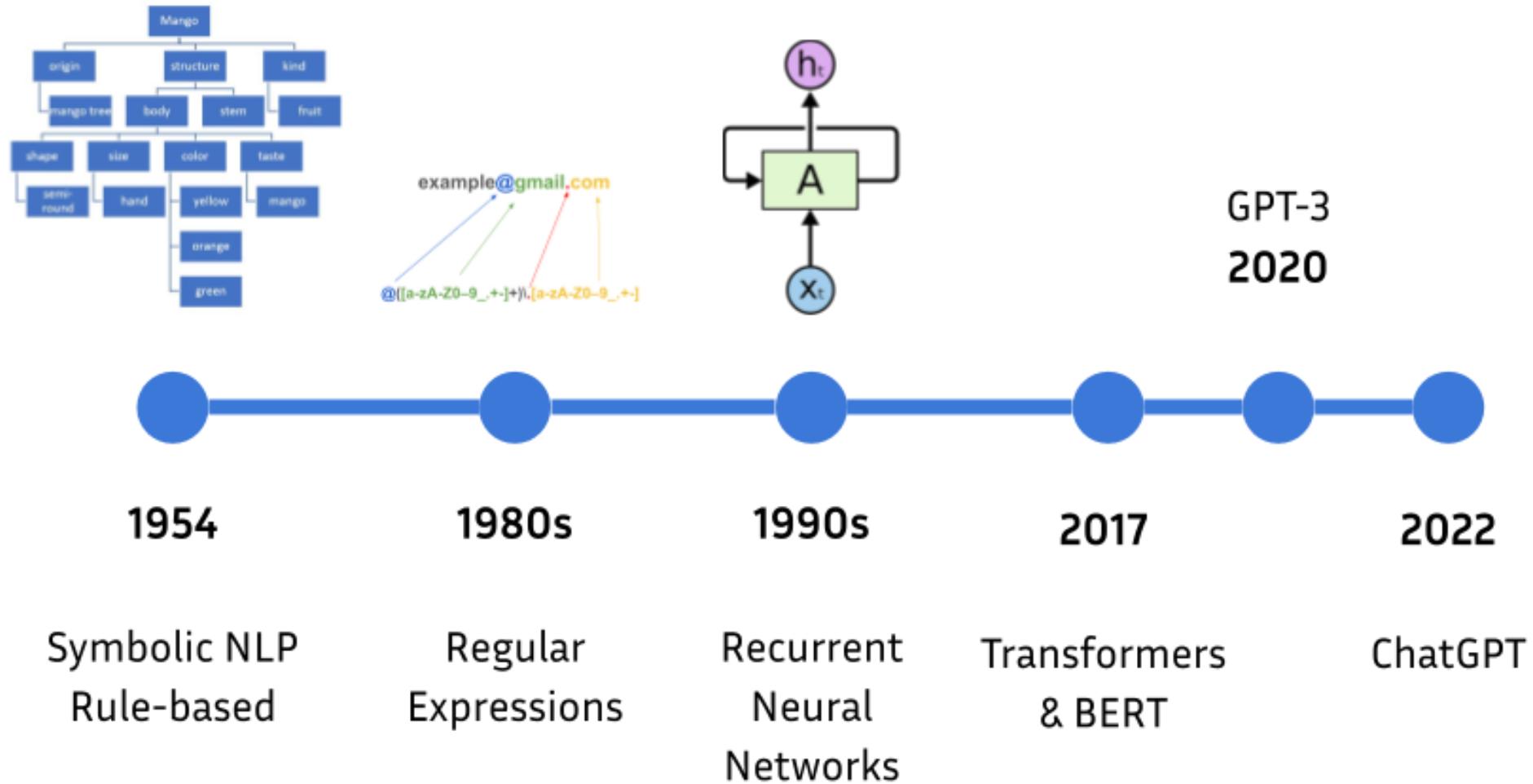


# **Course 1: Introduction & Recap**

# Welcome to the Advanced NLP course!

# NLP in recent years



# NLP in 2023

Default (GPT-3.5)



How to write a course for Advanced NLP?



Writing a course for Advanced Natural Language Processing (NLP) requires careful planning and organization. Here's a step-by-step guide to help you create a comprehensive and effective Advanced NLP course:

1. Define the Course Objectives:

- Start by clarifying the goals of the course. What do you want students to learn or achieve by the end of the course?
- Identify the target audience and their prerequisites, such as prior NLP knowledge or programming skills.

2. Create a Course Outline:

- Outline the major topics and modules that the course will cover. Consider breaking it

# NLP in 2023

A screenshot of a Google search results page. The search query "how to write a course on NLP" is entered in the search bar. The results page shows a header with "Environ 24 000 000 résultats (0,38 secondes)". Below this, a "Conseil" message encourages filtering results by language. The first result is from "Hugging Face" and links to an "Introduction - Hugging Face NLP Course". The course description mentions teaching about natural language processing (NLP) using libraries from the Hugging Face ecosystem — Transformers, Datasets, ...

← → C google.com/search?q=how+to+write+a+course+on+NLP&rlz=1C5CHFA\_enFR1003FR1003&oq=how+to+write+a+course+on+NLP&g...

Services Numériques Reading group On the Sentence... Cantine Meet – vrv-bwta... Réunion Eric/Benoit load\_dataset.ipyn... Te...

Google how to write a course on NLP X | ⚡ 🎧 📸 🔎

Tous Vidéos Images Livres Actualités Plus Outils

Environ 24 000 000 résultats (0,38 secondes)

Conseil : Limitez cette recherche aux résultats en **français**. En savoir plus sur le filtrage par langue

Hugging Face  
https://huggingface.co › learn › nlp... ::

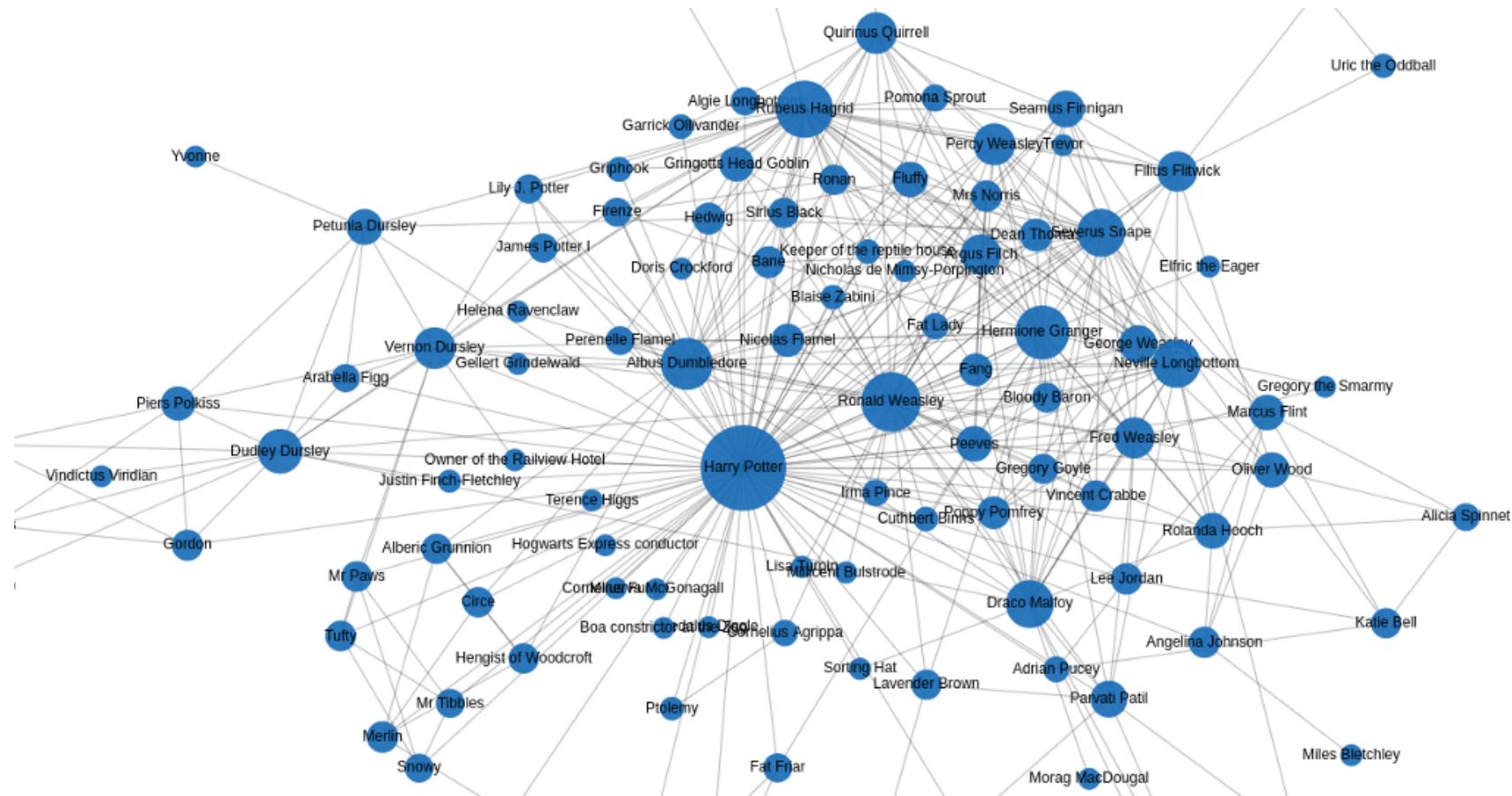
**Introduction - Hugging Face NLP Course**

This **course** will teach you about **natural language processing (NLP)** using libraries from the Hugging Face ecosystem — **Transformers**, **Datasets**, ...

# NLP in 2023



# NLP in 2023



# NLP perspectives



# NLP perspectives

The screenshot shows two open files in a dark-themed IDE:

- GITHUB COPILOT: CHAT**: A chat interface where monalisa asks GitHub Copilot to "Write a set of unit test functions for the selected code". Copilot generates the following Python test code:

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'), (datetime.datetime(2023, 1, 3), 2.59, 'DKK'), (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_comments(self):
        expenses_string = '# This is a comment\n2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'), (datetime.datetime(2023, 1, 3), 2.59, 'DKK'), (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

A note at the bottom states: "The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format."

- module.py**: The source code for the expense parser. Copilot has completed the implementation of the `parse_expenses` function:

```
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))
    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

At the bottom of the editor, status information includes: Ln 17, Col 3, Spaces: 2, UTF-8, LF, {}, TypeScript.

# **So, is NLP solved?**

**(No.)**

# NLP challenges

---



Give me a random string of length 35



Sure, here's a random string of length 35:

`J9kPzHfRq81G2sT7Nv6yX1W4c00mZoBwA5U3j`



Can you invert all letters in the following sentence: "I am currently giving an interesting course in Natural Language Processing at EPITA in Le-Kremlin-Bicêtre."



Certainly! Here's the sentence with all the letters inverted:

"I ma yltnerurc ygivni na gnitseretni esruoc ni lartuaN egaugnal gnicessorP ta ATIPE ni ertnebiC-tciB-nimelK-eL."

# NLP challenges

---



It takes 3 hours to dry 3 towels. How long does it take to dry 9 towels?

---



If it takes 3 hours to dry 3 towels, it implies that you can dry 1 towel in 3 hours. To find out how long it takes to dry 9 towels, you can use the same rate:

1 towel takes 3 hours to dry, so:

9 towels will take 9 times as long, which is:

$$9 \text{ towels} \times 3 \text{ hours/towel} = 27 \text{ hours.}$$

So, it will take 27 hours to dry 9 towels.

---

# NLP challenges



Let's picture the CEO of a very big international company. The CEO has to do a lot of business trips all around the world, so the CEO's wife is alone to help the children with the homework. Write the informal text message of the wife where she expresses her anger.



Sure, here's an informal text message where the CEO's wife expresses her frustration:

"Ugh, this is getting ridiculous! 😤 Another week alone with the kids while he's jet-setting around the world. 🌎 Can't believe he's missing out on their lives. Homework's a mess, and I'm doing it all. 😢 When is he ever gonna slow down and be here for us?? 😔💔"



Write the next text message, where she positively describes the CEO physically in details (skin, eyes, hair, ...).



"Despite all the frustration, I can't deny that man's got a magnetic charm that's hard to resist. 😍 His skin is that perfect sun-kissed bronze, and those deep, piercing blue eyes always get me. ✨ His hair, just the right shade of salt and pepper, makes him look even more distinguished. 😍 Sometimes, it's just hard to stay mad at someone who looks like a movie star. 🌟"

# Course organization



Matthieu  
Futerl-Peter  
(mail:futerl)



Nathan  
Godey



Francis  
Kulumba

# Course organization

- Part 1
  - **When** : 4 days (30/11, 07/12, 14/12, 21/12)
  - **Subject** : General NLP
  - **Goal** : Know how to build and deploy a custom ChatGPT-like assistant.

# Evaluation

- Group project (4-5 people)
- Two options
  - *Demo*
  - *R&D project*

# Evaluation - Demo (option 1)

Use a well-known approach to produce a MVP for an original use-case and present it in a demo.

*Example: An online platform that detects AI-generated text.*

## Evaluation - R&D project (option 2)

Based on a research article, conduct original experiments and produce a report.

*Example: Do we need Next Sentence Prediction in BERT? (Answer: No)*

# Evaluation

- Mid-term project evaluation (30%)
  - Project proposal
  - First elements
  - ~January
- Final project (70%)
  - Short report showing each person's contribution
  - Github repo

# Evaluation

- You can already constitute teams
- Send an email with:
  - Names of team members
  - cc to everyone in the team
  - *Demo or R&D*
  - One-sentence description of project
- If you really have no idea what to do, ask me

# Program

- **Session 1 (Today):** Recap
- **Session 2 (7/12):** Tokenization
- **Session 3 (14/12):** Language Modeling
- **Session 4 (21/12):** Modern NLP with limited resources

# Questions

# Recap

# Quiz time!

<https://docs.google.com/forms/d/1BZaBagWIpVgKLsT2NdjJ4pXzPXTBnsxv52jEF4CR6GY/prefill>

# **Basic concepts in NLP**

# Stemming / Lemmatization

**Stemming** shortens variations (*inflected forms*) of a word to an identifiable root

Example:

*Flying using airplanes harms the environment*

=>

*Fly us airplane harm the environ*

# Stemming / Lemmatization

**Lemmatization** groups variations (*inflected forms*) of a word to an identifiable representative word

Example:

*Flying using airplanes harms the environment*

=>

***Fly use airplane harm the environment***

# Tokenization

**Tokenization** turns text strings (= lists of characters) into lists of meaningful units (e.g. words or subwords)

Example:

*Flying using airplanes harms the environment*

=>

*( Fly | ing | us | ing | air | planes | harms | the | environ | ment )*

(see Course 2)

# Regular expressions

**Regular expressions** is a string that specifies a match pattern in text

Example:

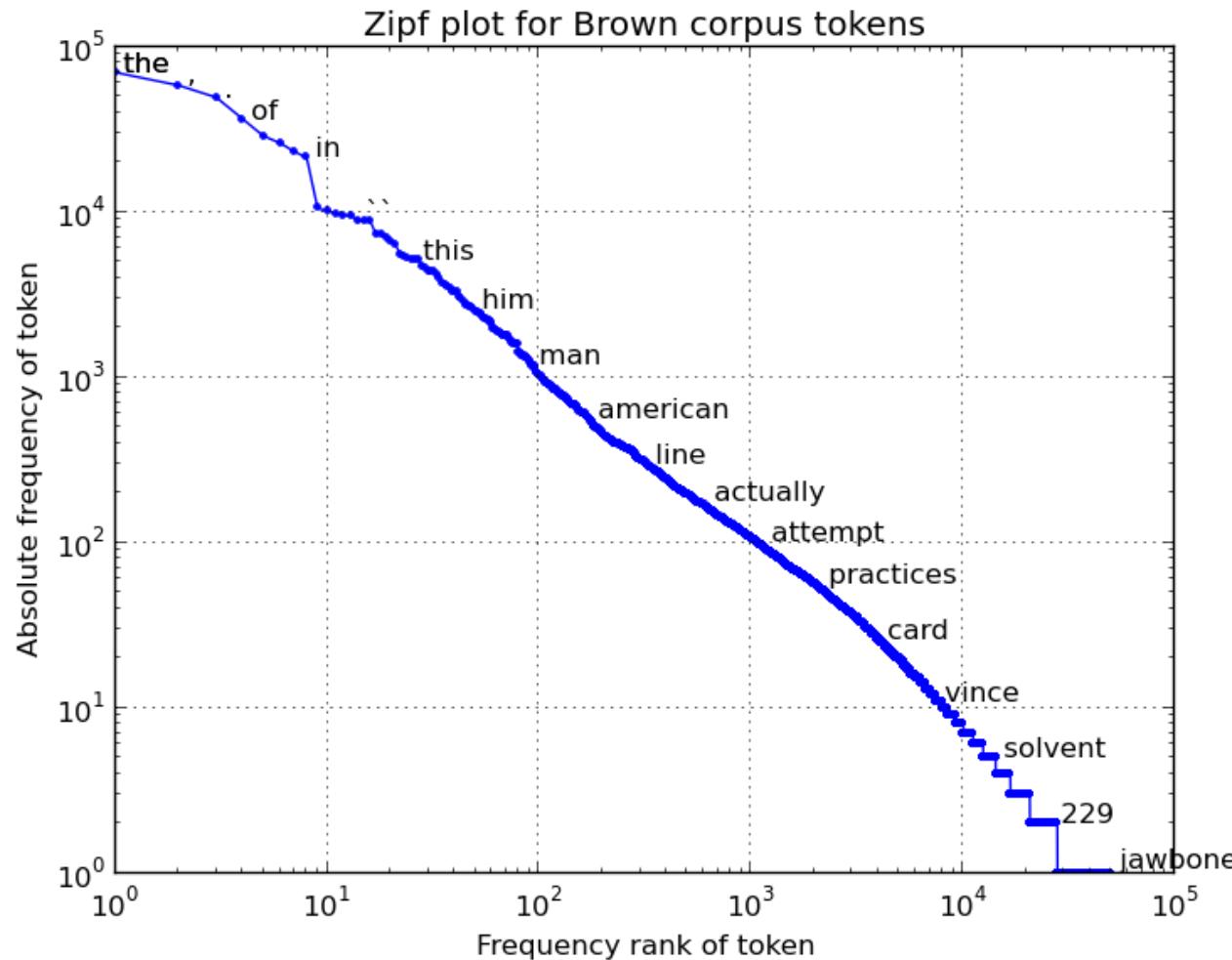
`/\w*ing\b/` -> *Flying using airplanes harms the environment*

=

Two matches:

- *Flying*
- *using*

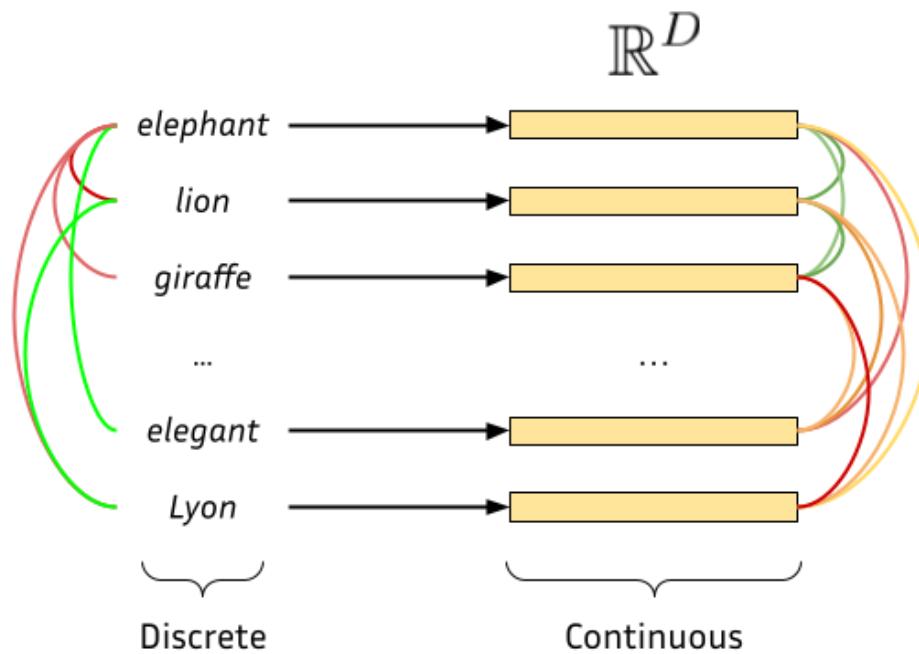
# Zipf's law



# Machine Learning & NLP

# Embeddings

- Vectors that represent textual entities (words, sentences, documents, ...)



# Bag-of-Words Embeddings

Represent a sentence/document by counting words in it.

Example:

*John likes to watch movies. Mary likes movies too.*

=>

```
{John: 1, likes: 2, to: 1, watch: 1, movies: 2, Mary: 1, too: 1}
```

=>

```
[1, 2, 1, 1, 2, 1, 1]
```

# TF-IDF Embeddings

Represent a sentence/document by comparing how frequent a word is in the extract vs. how frequent the word is in general.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term  $x$  within document  $y$

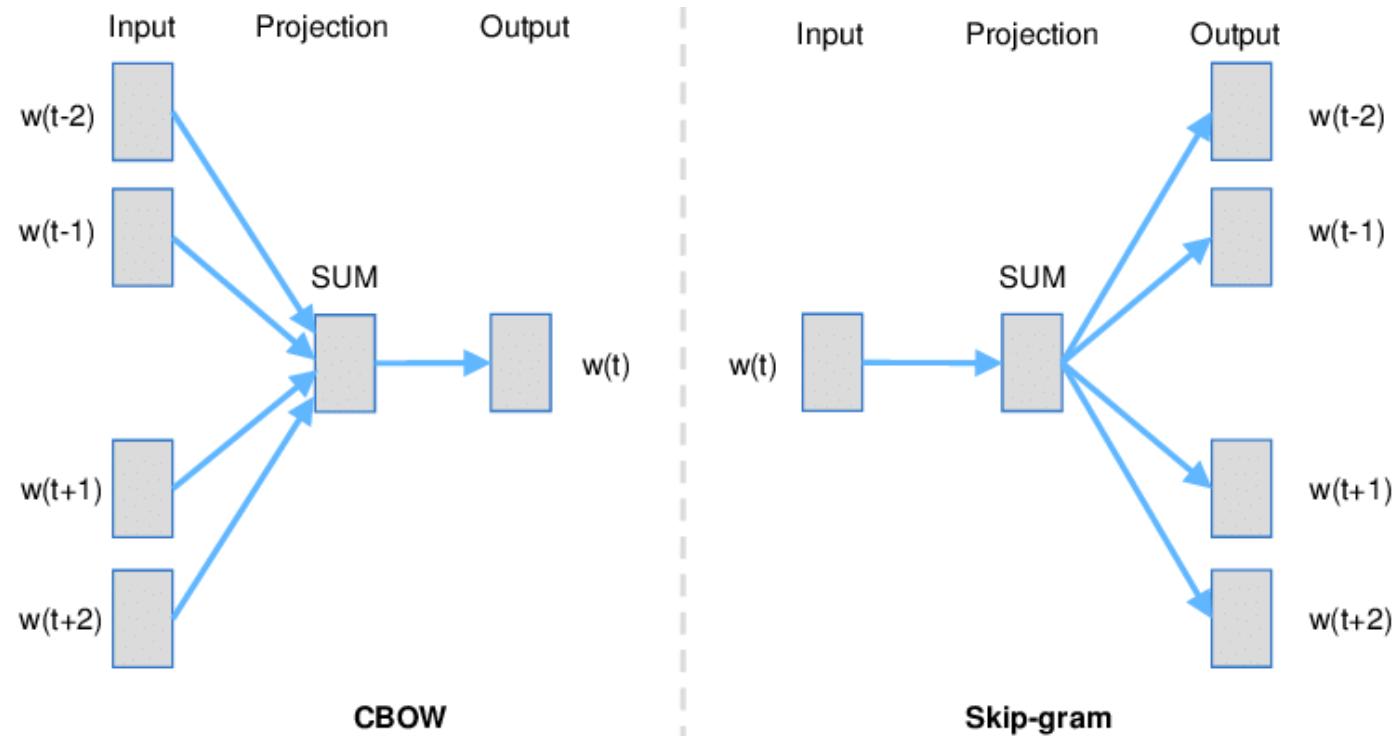
$tf_{x,y}$  = frequency of  $x$  in  $y$

$df_x$  = number of documents containing  $x$

$N$  = total number of documents

# Skip-gram & CBoW

Learn embeddings **without supervision/counting**.

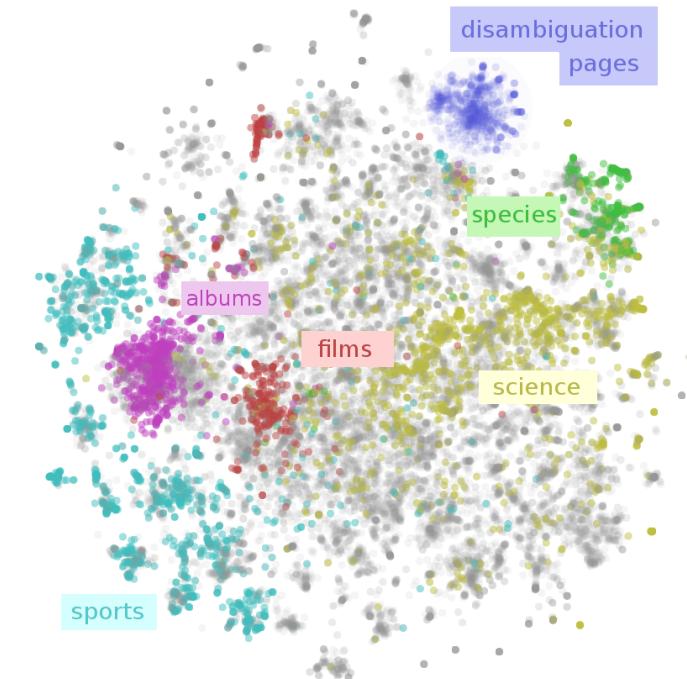


# Static word embeddings

Allow semantically meaningful spaces and can be used as features.

Known static embeddings include:

- GloVe
- FastText
- Word2Vec
- ...

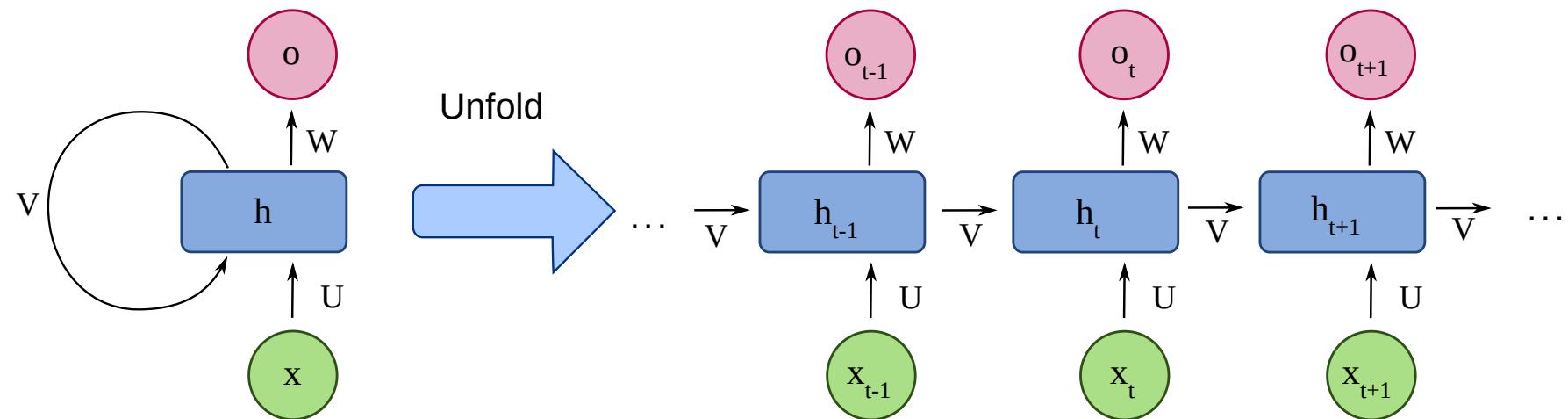


# Deep Learning & NLP

# RNNs

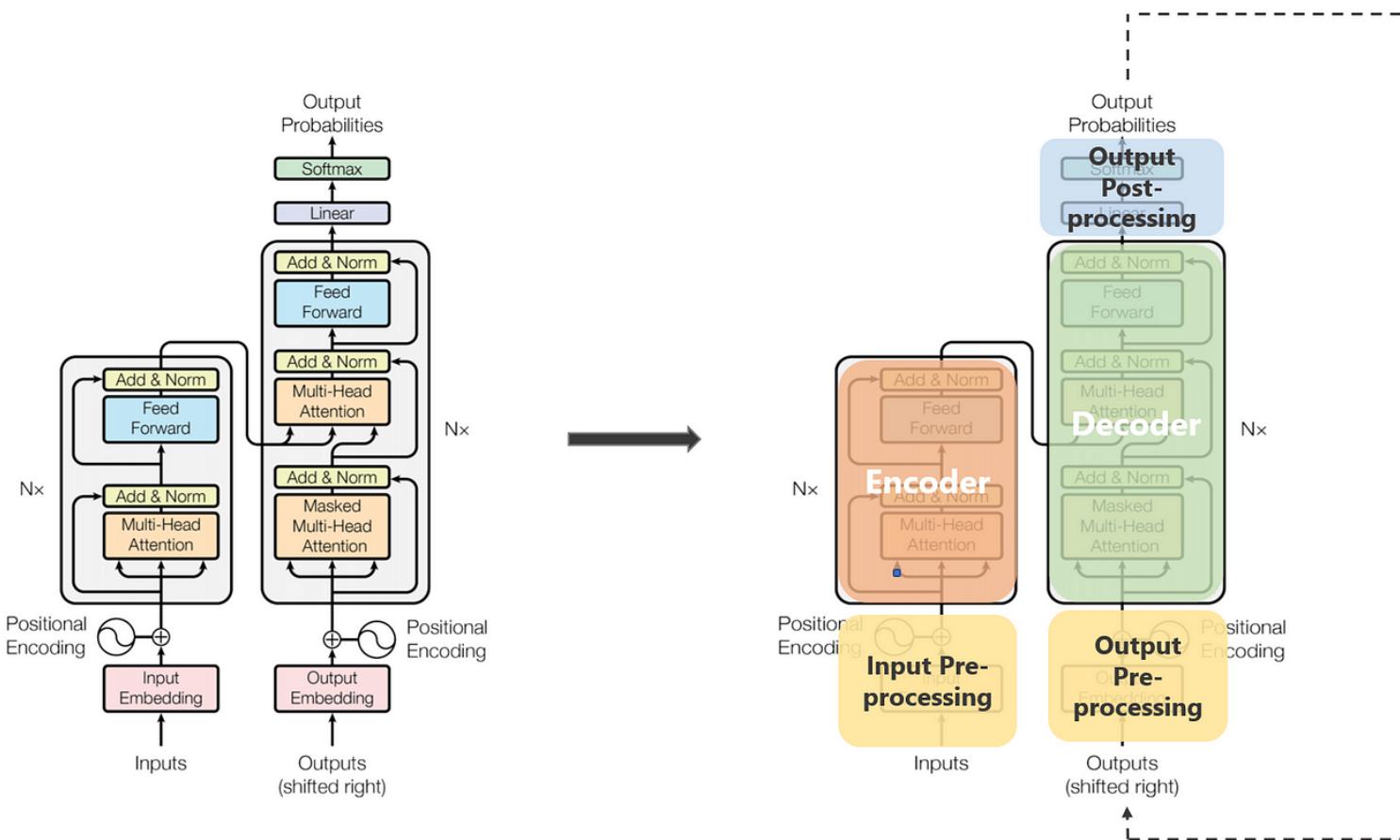
## Recurrent Neural Networks

Neural Networks that are able to process input sequences recurrently.



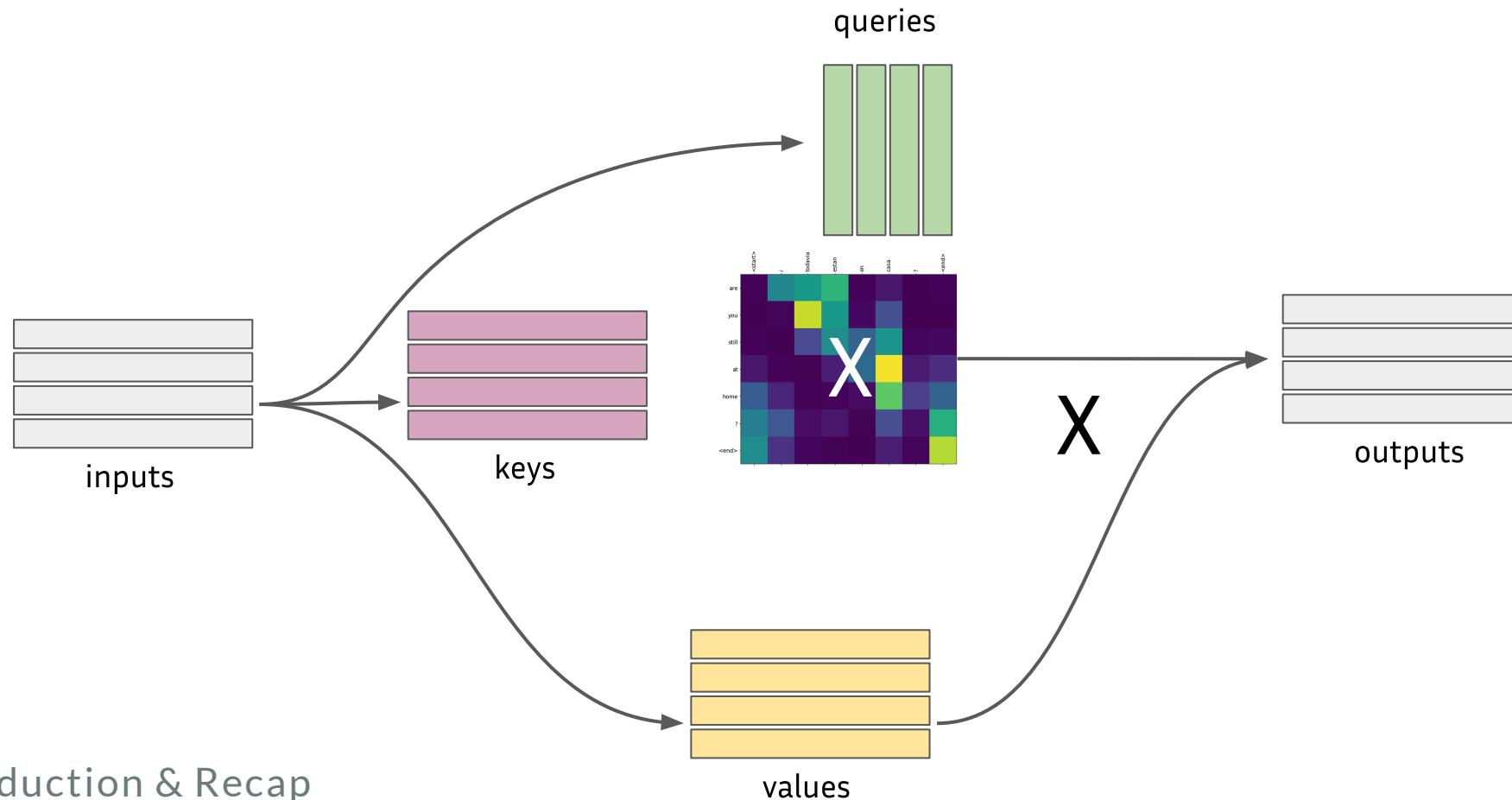
# Transformers

Neural Networks that are able to process input sequences directly.



# Transformers - Self-Attention

Self-attention allows comparing inputs and representing interactions.



# Fine-tuning

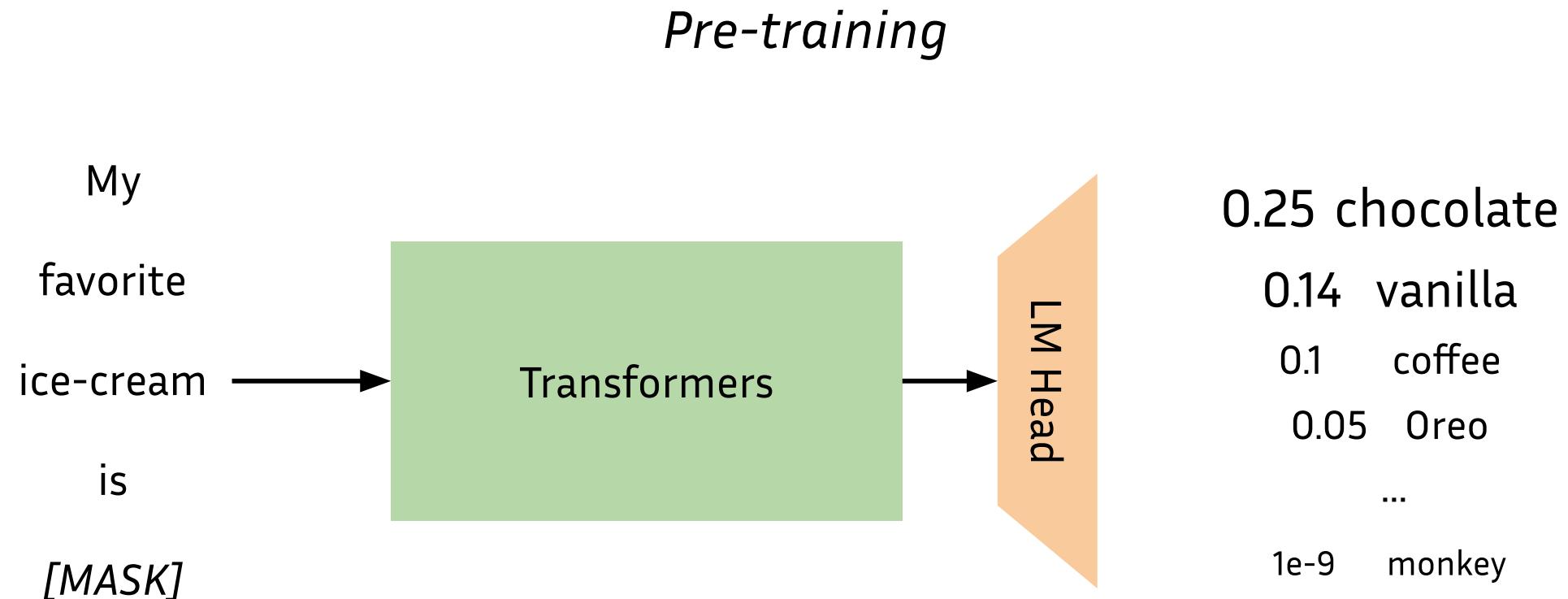
Modern NLP models are built in two separate steps:

- **Pretraining:** models are trained **without supervision** on raw text data (e.g.: next word prediction on all text from Wikipedia).
- **Fine-tuning:** pretrained models are *retrained* on a smaller annotated dataset that matches the final task.

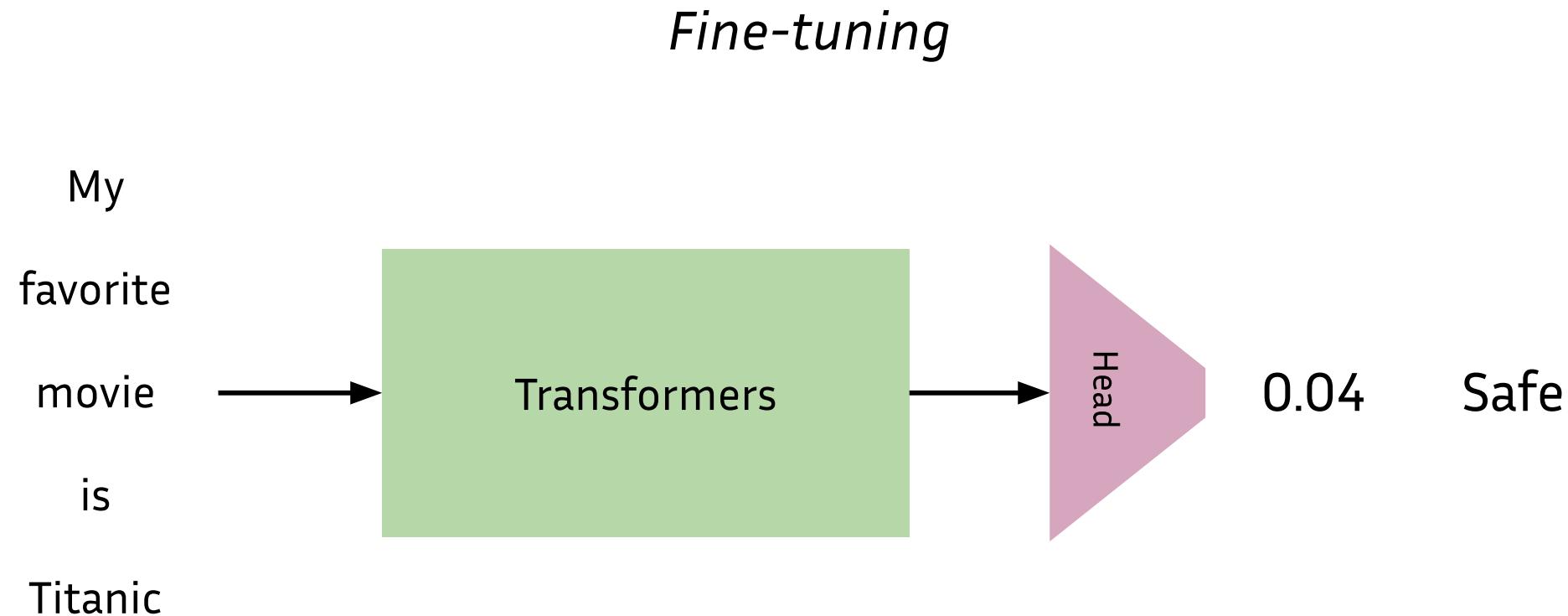
# Fine-tuning - example

1. Download a Transformers language model that was pretrained on data from the Internet (books, Wikipedia, ...) to predict future words
2. Take a list of tweets and mark them as *suspicious* or *safe*
3. Add a classifier on top of the LM that predicts a float in  $[0, 1]$
4. Train the whole model (LM + classifier) to predict *suspicious* or *safe* tweets

# Fine-tuning - example



# Fine-tuning - example



# Questions?

# Lab session