COMP0087 25/26
Lecture 1: Introduction

13/01/2026

Who are we?

Instructors

Yao Lu

Lecturer in Natural Language Processing

Contact: yao.lu@cs.ucl.ac.uk

Research: language model pretraining, prompt optimisation

Instructors

Ehsan Shareghi

Associate Professor in Natural Language Processing

Contact: ehsan.shareghi@gmail.com

Research: LLM safety, speech processing

Instructors

Pontus Stenetorp

Professor in Natural Language Processing

Contact: p.stenetorp@cs.ucl.ac.uk

Research: Anything that fits into NLP (Multilinguality, data, knowledge, generalisation, etc)

Instructors

Vasileios (Bill) Lampos

Associate Professor

Contact: v.lampos@ucl.ac.uk

Research: Sequence modelling (time series forecasting), LLMs training dynamics (e.g., mode collapse)

Assumptions about you

You already know the basics of applied ML:

- What the training/dev/test cycle is

- What a loss function is

- What an optimization technique does (at a high level)

- How to evaluate a model (given a standard benchmark)

Assumptions about you

You can already do these things:

- Perform basic data manipulation

- Build basic knowledge in Python

Who are you?

Level of experience in NLP

1. Beginner (no prior NLP experience)
2. Intermediate (coursework or projects)
3. Advanced (research or professional experience)

Who are you?

I'm taking this course to

1. Prepare for job searching
2. Learn how to conduct NLP research
3. Apply NLP to ongoing projects and research
4. Other

Module communication

General, non-personal, queries

Largest postgraduate module

General, non-personal, queries:

Please, pretty please use the Moodle forum

Personal queries

Personal queries:

comp0087@cs.ucl.ac.uk

General project-related queries

General, project-related queries:

Moodle forum

Your project-related queries

Specific, project-related queries:

Assigned teaching assistant

Administrative-related queries

Administrative logistics (registration, etc.):

cs.pgt-students@ucl.ac.uk

Module content

Learning goals

1. Foundational “building blocks” for LLMs

2. Advanced LLM development/applications

3. Practical, empirical understanding of building NLP systems

- Analysis, empirical experiments, etc.

- Group project

Teaching plan

Basic NLPs 3-4 weeks

Advanced NLPs (LLMs and applications) 3-4 weeks

Topic Lectures from Industry speakers (e.g., Meta AI, Microsoft etc) 2-3 weeks

NLP Course Projects (supervised by our awesome TAs and faculty members) 1-10 weeks

Teaching plan: Foundations

Tokenisation

What is your input?

Token representation

In what form?

Teaching plan: Foundations

Traditional LM

Neural LM

How to model the data?

Teaching plan: Foundations

Attention

Transformer

Which architecture and why?

Teaching plan: Foundations

Prompting

How to "talk" to the model?

Teaching plan: Advanced Topics

Advanced NLPs

- Tuning Large Language Models

- Alignment with Human Feedback

- Reasoning LLMs

- Retrieval Augmented Generation (RAG)

- LLMs Pretraining

…

Teaching plan: Industry Lectures

Please let us know which topics you're interested in.

Group project

Assessment

Coursework (Group): 100% of module mark

- Mandatory bi-weekly progress meetings
- Single-page progress reports (due date TBD)
- Eight-page final report
- Not including references

Group formation

- Groups of *five* or *six*
- * Exceptions will be rare
- Group formation due 18 January by midnight
- * Start looking *now*
- * Decide on group name
- * Post UCL e-mails in dedicated Moodle forum

Project requirements

- **Must** involve language
- Empirical investigation
- Designed by students, with the *help* of us
- * Ultimate responsibility and control is with *you*
- Outcomes *may* be publishable
- * But *no* guarantees

Project timeline sketch

- Week 1 to 2: Project design and literature review
- Week 3 to 4: Data collection, experimental design, initial implementation
- Week 5 to 6: Experiments not working out: Revisions, revisions, revisions,
- Week 7 to 8: Additional experiments and analysis
- Week 9 to 10: Report writing

Project design

- Minimum viable project
- Align with group interests and teaching assistant expertise
- Design to be "split"

Computational resources

- *Limited*: Design your projects with this in mind!
- - UCL Computer Science:
- * https://tsg.cs.ucl.ac.uk/gpus
- * ~30 x NVIDIA RTX 4060 cards (16GB)
- * ~25 x NVIDIA RTX 3090 cards (24GB)

Computational resources

- Google Colab:
- * https://colab.research.google.com
- * Web "notebooks" with free GPUs and TPUs (LLM access?)
- * Can pay for better service (GPU server rental)
- co:here credits
- * Application process announced shortly

Project design example

~~I want to train a language model.~~

I want to train a language model that better understands twitter posts.

Project design example

- What's broken? (Problem)
- How will you know it's fixed? (Metrics)
- What will you try? (Approach)
- Can you actually do it? (Feasibility)
- What if it fails? (Contingency)

Project design example

What's Broken?

- Why do current LLMs struggle with tweets?
- What evidence supports this? (Literature review)
- Show examples of failures

Project design example

How Will You Know It's Fixed?

- Which task? (sentiment, NER, sarcasm detection, etc.)
- Which dataset? (TweetEval? SemEval? Custom?)
- What's the baseline? (GPT-4? RoBERTa-Twitter? Prompting vs. fine-tuning?)

Project design example

What Will You Try?

- Which base model?
- Prompting, fine-tuning, or pretraining? (Keep budget in mind)
- What training data? (Source, size, licensing)
- Why should this approach work? (Hypothesis)

Project design example

Can You Actually Do It?

- GPU hours (check maximum available resources)
- Training data (size, cost)
- Evaluation data (quality, cost)
- Timeline (10 weeks in total)
- Split tasks

Project design example

What If It Fails?

- No improvement: Why? Data quality? Wrong task? Write negative result analysis
- Reframe as benchmark or dataset contribution
- Discuss with your TA in advance

Teaching assistants

Adam Oomerjee-Vawda (adam.vawda@gmail.com)

- PhD student at UCL AI centre
- Interests: Efficient models, reasoning
- Favourite paper:
- Bottlenecked Transformers: Periodic KV Cache Abstraction for Generalised Reasoning. https://arxiv.org/abs/2505.16950

Edan Toledo (edan.toledo.24@ucl.ac.uk)

- PhD student at UCL and Facebook
- Interests:
- * General search methods * Agentic systems * Meta-learning * Reinforcement learning
- Favourite paper:
- * "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play" by Silver et al. (2018) * https://www.science.org/doi/10.1126/science.aarг6404

Eduardo Sánchez (eduardo.sanchez.22@ucl.ac.uk)

- final-year PhD student at UCL and Facebook
- Interests:
- * Low-resource languages * Multilinguality
- Favourite paper:
- * "Linguini: A benchmark for language-agnostic linguistic reasoning" by Sánchez et al. (2024) https://arxiv.org/abs/2409.12126

Jiayi Wang (jiayi.lin.wang@ucl.ac.uk)

- final-year PhD student
- Interests:
- * Inclusive and efficient multilingual large language models
- Favourite paper:
- * "Multilingual Pretraining Using a Large Corpus Machine-Translated from a Single Source Language" by Wang et al. (2023)
- * https://arxiv.org/abs/2410.23956

Karen Hambardzumyan (karen.hambardzumyan.23@ucl.ac.uk)

- PhD student at UCL and Facebook
- Interests:
- * Mechanistic interpretability * Multi-agent systems
- Favourite paper:
- * "WARP: Word-level Adversarial ReProgramming" by Hambardzumyane et al. (2021) * https://aclanthology.org/2021.acl-long.381

Lovish Madaan (lovish.madaan.23@ucl.ac.uk)

- PhD student at UCL and Facebook (GenAI)
- Interests:
- * Generalisation behaviours in reinforcement learning * Better/harder evaluation
- Favourite paper:
- * "Amortizing intractable inference in large language models" by Hu et al. (2024) * https://arxiv.org/abs/2310.04363

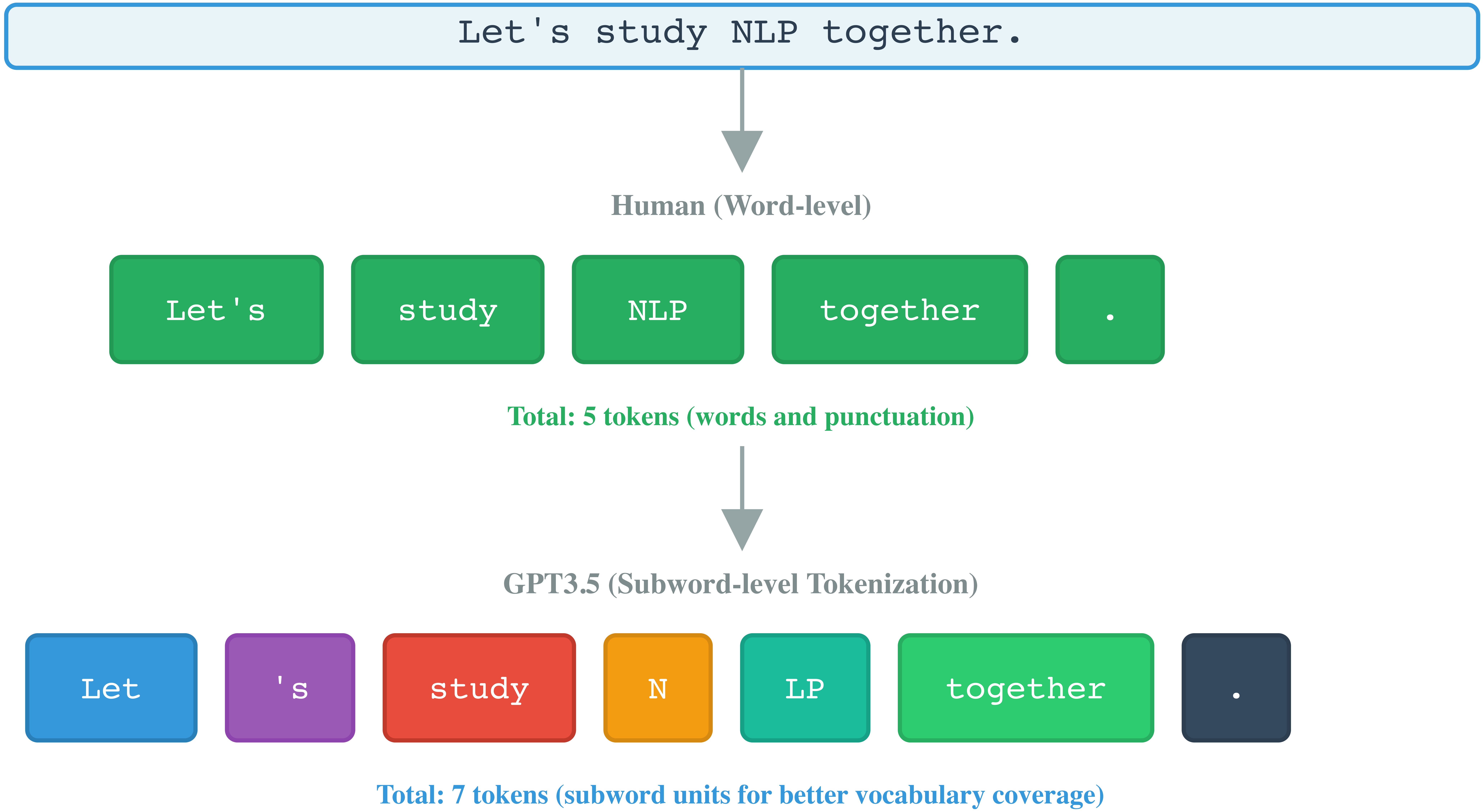
Ralph Tang (r-tang.25@ucl.ac.uk)

- PhD student at NLP group
- Interests: * Multimodality * LLM/VLM interpretability
- Favourite paper:
- What the DAAM: Interpreting Stable Diffusion Using Cross Attention. ACL 2023

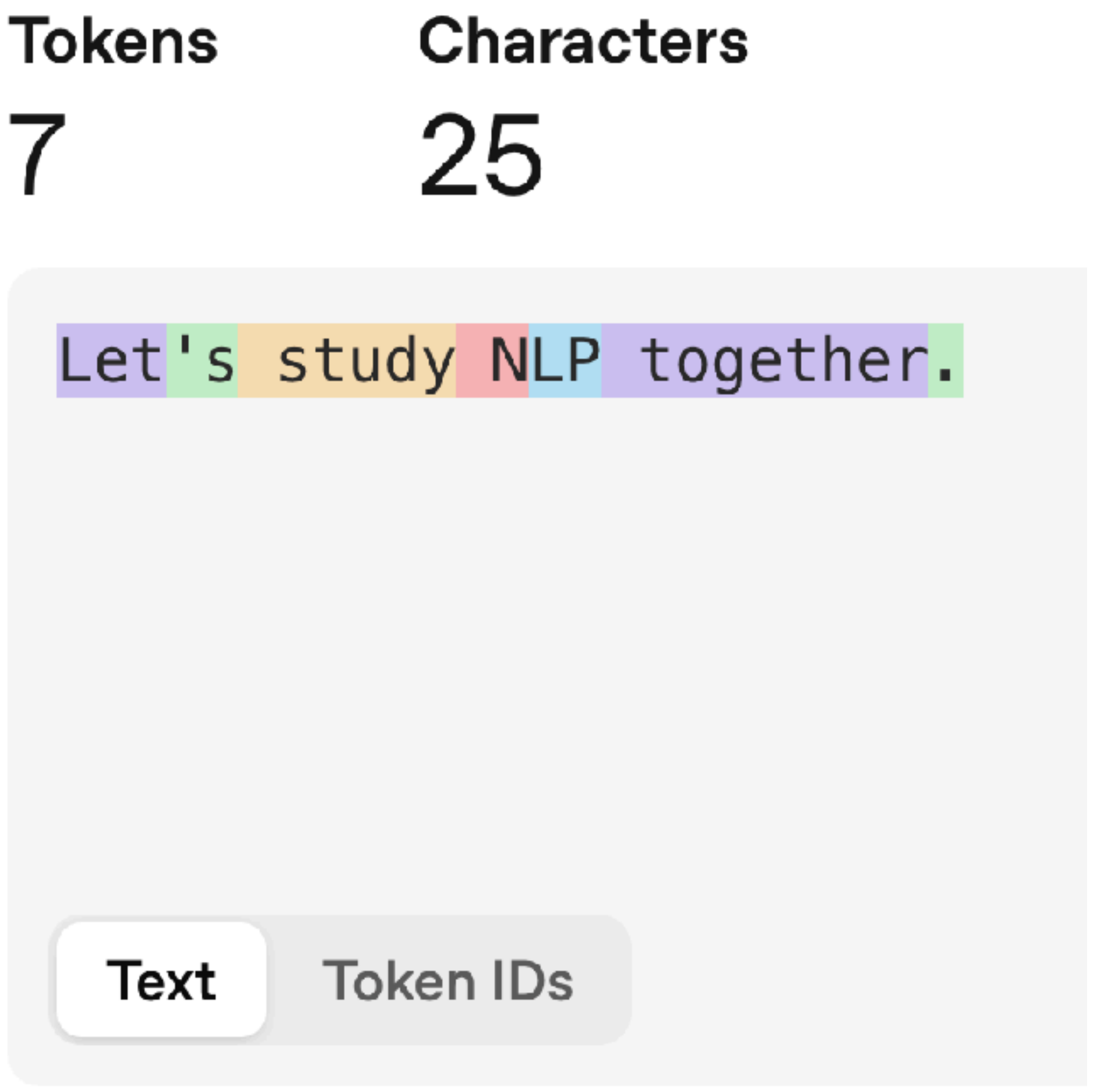
Hossein A (Saeed) Rahmani
(hossein.rahmani.22@ucl.ac.uk)

- PhD student at NLP group
- Interests: * Evaluations and benchmarks * Self-Improvement
- Favourite paper:
- "Large Language Models Cannot Self-Correct Reasoning Yet" by Huang et al. (2024) https://arxiv.org/abs/2310.01798

Tokenisation


Let's study NLP together.
Human (Word-level)
Let's
study
NLP
together
.
Total: 5 tokens (words and punctuation)
GPT3.5 (Subword-level Tokenization)
Let
's
study
N
LP
together
.
Total: 7 tokens (subword units for better vocabulary coverage)

From Text to Tokens

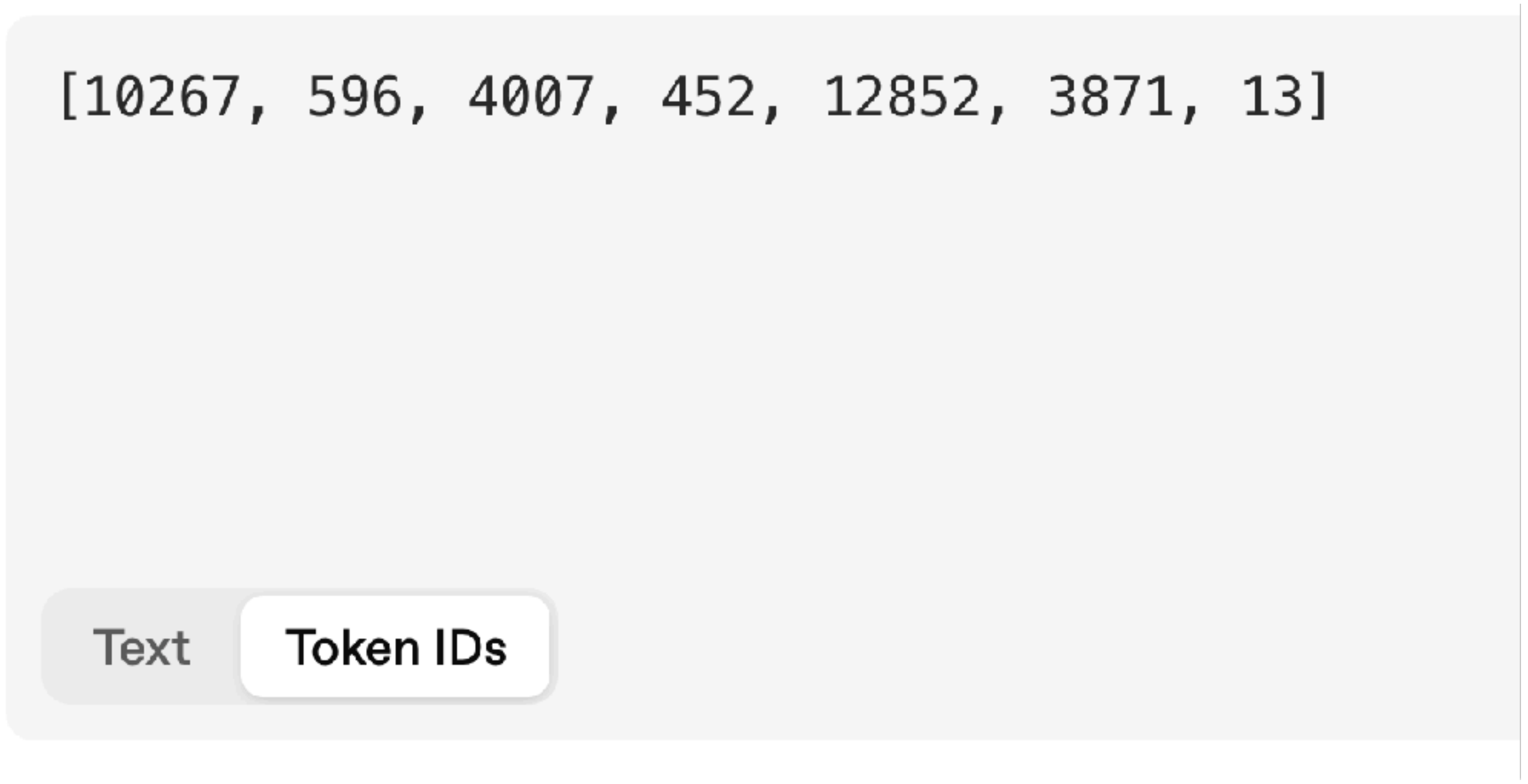


Tokens
7

Characters
25

```
Let's study NLP together.
```

Text Token IDs



```
[10267, 596, 4007, 452, 12852, 3871, 13]
```

Text Token IDs

The Reverse Spelling Problem

strawberry -> y-r-r-e-b-w-a-r-t-s

Why is this difficult?

The Reverse Spelling Problem

strawberry -> [“str”, “aw”, “berry”]

-> y-r-r-e-b-w-a-r-t-s?

Why is this difficult?

The Reverse Spelling Problem

strawberry -> [“str”, “aw”, “berry”]

-> [“yr”,“reb”,“warts”]

Why is this difficult?

The Reverse Spelling Problem

strawberry -> [496, 675, 15717]

-> [11160, 32575, 64156]

Why is this difficult?

The Number Comparison Problem

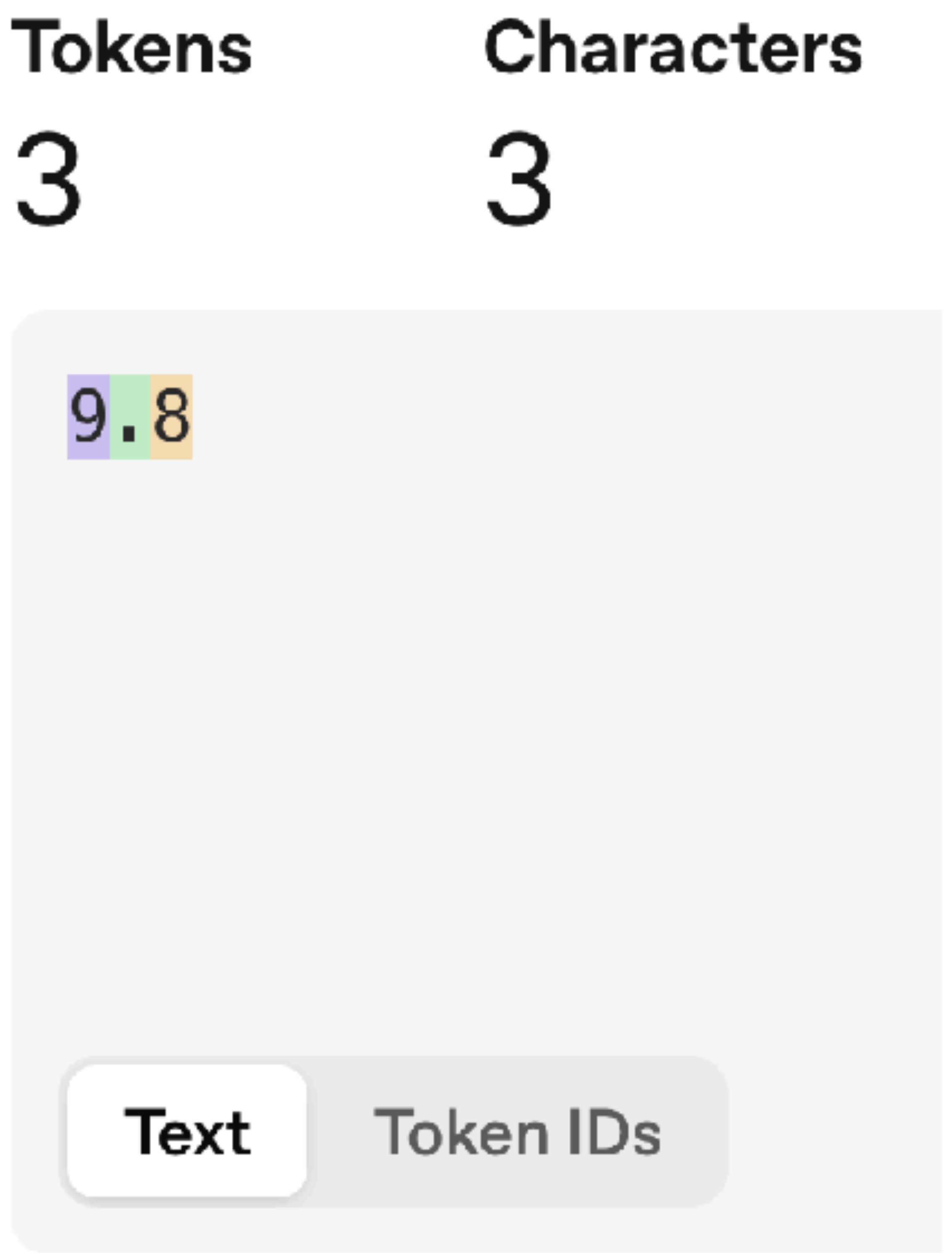
"Is 9.8 greater than 9.11?"

The Number Comparison Problem

Human: 9.8 > 9.11

The Number Comparison Problem

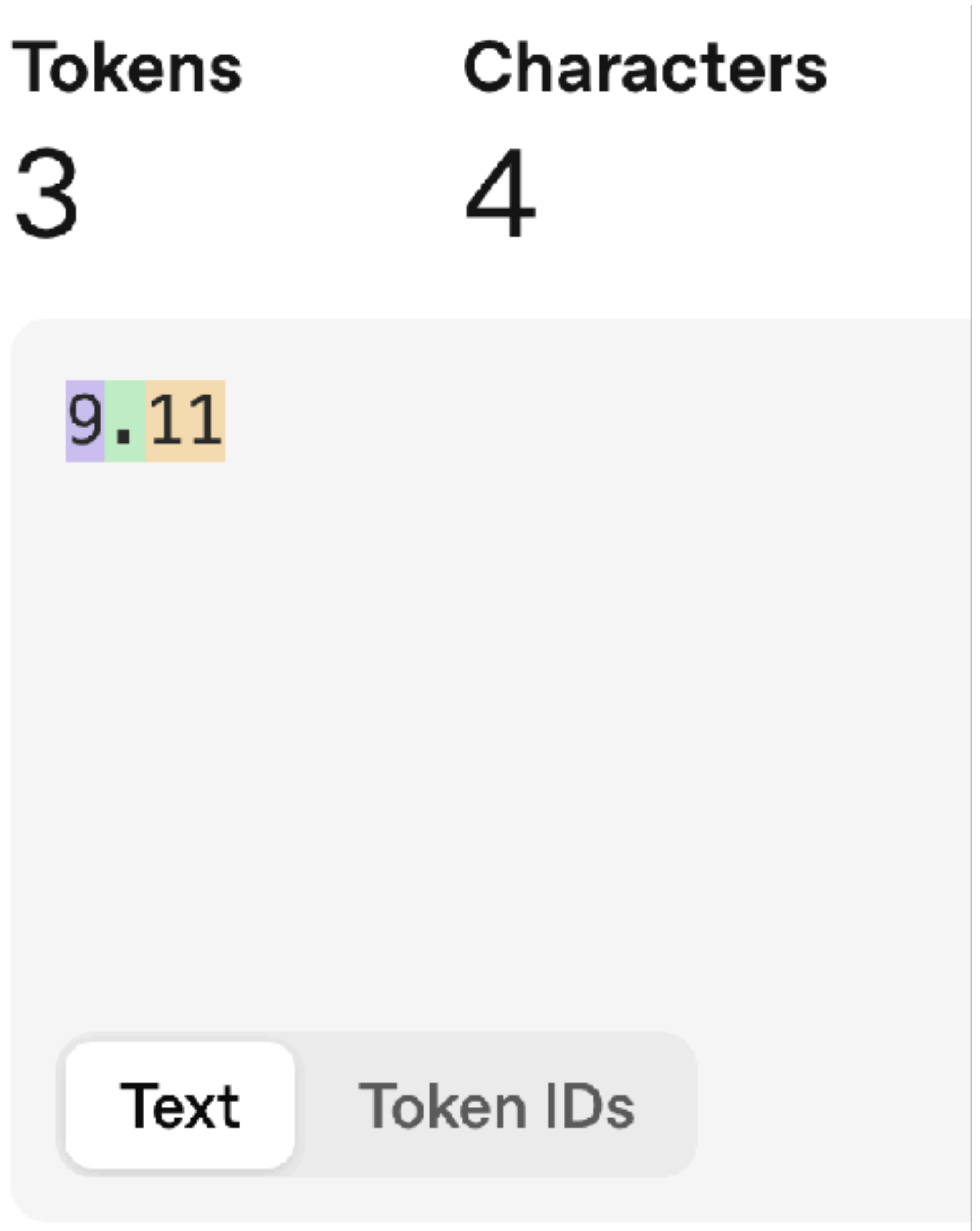
GPT4: 9.8 < 9.11



Tokens 3

Characters 3

9.8



Tokens 3

Characters 4

9.11

Definition of tokenisation

- The process of splitting **text** into **meaningful tokens**, which are then mapped to **numerical IDs**.

Classical tokenisation methods

Character-level tokenisation: strawberry -> [s, t, r, a, w, b, e, r, r, y]

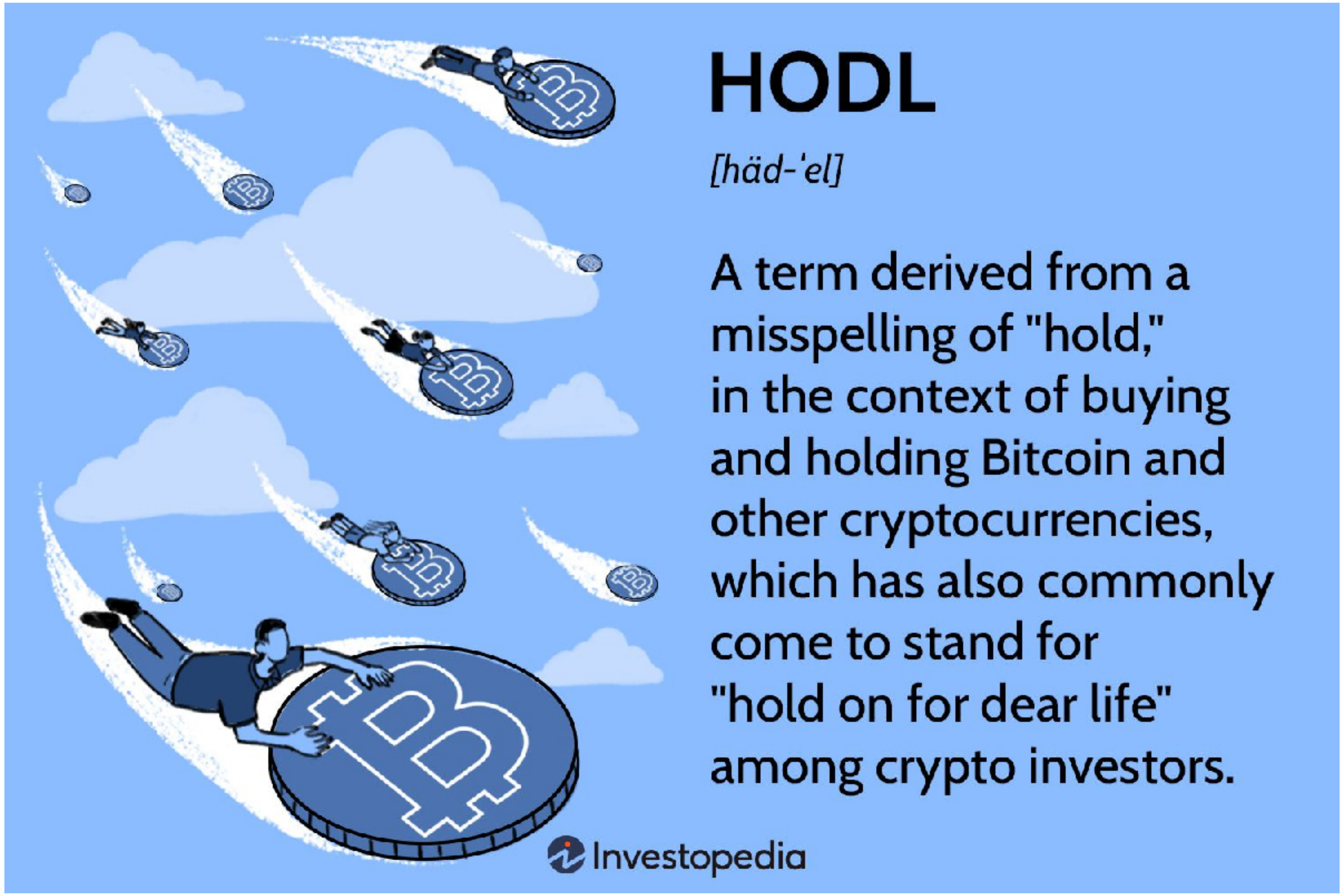
- Pros: a very small set of symbols (100?) can represent almost everything
- Cons: less meaningful segmentation

Classical tokenisation methods

Word-level tokenisation: “I am happy” -> [Index(I), Index(am), Index(happy)]

- Pros: meaningful and natural segmentation (by whitespace)
- Cons: huge vocabulary space

Out of Vocabulary Tokens



HODL

[häd-ˈel]

A term derived from a misspelling of "hold," in the context of buying and holding Bitcoin and other cryptocurrencies, which has also commonly come to stand for "hold on for dear life" among crypto investors.

Investopedia

Out of Vocabulary Tokens

Solution: replace out of vocabulary tokens with <UNK>

“My favourite food is <UNK>”

“My favourite <UNK> is <UNK>”

“<UNK> favourite <UNK> is <UNK>”

Tokenisation design

	Character-based method	Word-level method	?
Vocabulary size	~100	100,000	1k-100k
Corpus coverage	High (any text)	Medium (OOV issues)	High
Meaningfulness	Low	High	High
Sequence length	Very long	Short	Medium

Subword Tokenisation

- Frequent words stay as single tokens (“cat").
- Rare/complex words split into meaningful sub-units ("tokenisation" -> "token" + "isation").
- Handles OOV: Can decompose new words ("ChatGPT" -> "Chat" + "G" + "PT").

Byte-Pair Encoding (BPE)

strawberry -> [“str”, “aw”, “berry”]

Byte-Pair Encoding (BPE)

“London”

“Londres”

“Londra”

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

(L, o) 3

{L, o, n, d, r, e, s, a}

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3

Byte-Pair Encoding (BPE)

"London"
"Londres"
"Londra"

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1
(d, r)	2

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1
(d, r)	2
(r, e)	1

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1
(d, r)	2
(r, e)	1
(e, s)	1

Byte-Pair Encoding (BPE)

"London"
"Londres"
"Londra"

{L, o, n, d, r, e, s, a}

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1
(d, r)	2
(r, e)	1
(e, s)	1
(r, a)	1

Byte-Pair Encoding (BPE)

“London”
“Londres”
“Londra”

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

(L, o)	3
(o, n)	4
(n, d)	3
(d, o)	1
(d, r)	2
(r, e)	1
(e, s)	1
(r, a)	1

Byte-Pair Encoding (BPE)

"L <u>on</u> d <u>on</u>"

"L <u>on</u> d r e s "

"L <u>on</u> d r a"

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

Byte-Pair Encoding (BPE)

“L on d on”

“L on d r e s ”

“L on d r a”

(L, on) 3

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

Byte-Pair Encoding (BPE)

“L <u>on</u> d <u>on</u>”

“L <u>on</u> d r e s ”

“L <u>on</u> d r a”

(L, on) 3

(on, d) 3

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

Byte-Pair Encoding (BPE)

“L on d on”

“L on d r e s ”

“L on d r a”

(L, on) 3

(on, d) 3

(d, on) 1

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

Byte-Pair Encoding (BPE)

“L <u>on</u> d <u>on</u>”

“L <u>on</u> d r e s ”

“L <u>on</u> d r a”

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

(L, on)	3
(on, d)	3
(d, on)	1
(d, r)	2

Byte-Pair Encoding (BPE)

“L on d on”

“L on d r e s ”

“L on d r a”

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

(L, on) 3

(on, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

Byte-Pair Encoding (BPE)

“L on d on”

“L on d r e s ”

“L on d r a”

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

(L, on) 3

(on, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

(e, s) 1

Byte-Pair Encoding (BPE)

“L on d on”

“L on d r e s ”

“L on d r a”

{L, o, n, d, r, e, s, a, on}

rule #1: (o, n) -> on

(L, on) 3

(on, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

Byte-Pair Encoding (BPE)

"L on d on"

"L on d r e s "

"L on d r a"

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

(L, on) 3

(on, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

Byte-Pair Encoding (BPE)

“Lon d on”

“Lon d r e s ”

“Lon d r a”

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

Byte-Pair Encoding (BPE)

“<u>Lon</u> d <u>on</u>”

“<u>Lon</u> d r e s ”

“<u>Lon</u> d r a”

(Lon, d) 3

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

Byte-Pair Encoding (BPE)

“Lon d on”

“Lon d r e s ”

“Lon d r a”

(Lon, d) 3

(d, on) 1

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

Byte-Pair Encoding (BPE)

“Lon d on”

“Lon d r e s ”

“Lon d r a”

(Lon, d) 3

(d, on) 1

(d, r) 2

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

Byte-Pair Encoding (BPE)

“Lon d on”

“Lon d r e s ”

“Lon d r a”

{L, o, n, d, r, e, s, a, on, Lon}

(Lon, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

Byte-Pair Encoding (BPE)

"Lon d on"

"Lon d r e s "

"Lon d r a"

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

(Lon, d)	3
(d, on)	1
(d, r)	2
(r, e)	1
(e, s)	1

Byte-Pair Encoding (BPE)

“Lon d on”

“Lon d r e s ”

“Lon d r a”

{L, o, n, d, r, e, s, a, on, Lon}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

(Lon, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

Byte-Pair Encoding (BPE)

“<u>Lon</u> d <u>on</u>”

“<u>Lon</u> d r e s ”

“<u>Lon</u> d r a”

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on
rule #2: (L, on) -> Lon
rule #3: (Lon, d) -> Lond

(Lon, d) 3

(d, on) 1

(d, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

Byte-Pair Encoding (BPE)

“Lond on”

“Lond r e s ”

“Lond r a”

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

Byte-Pair Encoding (BPE)

“<u>Lond</u> <u>on</u>”
“<u>Lond</u> r e s ”
“<u>Lond</u> r a”

(Lond, on) 1

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on
rule #2: (L, on) -> Lon
rule #3: (Lon, d) -> Lond

Byte-Pair Encoding (BPE)

“Lond on”

“Lond r e s ”

“Lond r a”

(Lond, on) 1

(Lond, r) 2

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

Byte-Pair Encoding (BPE)

“Lond on”

“Lond r e s ”

“Lond r a”

(Lond, on) 1

(Lond, r) 2

(r, e) 1

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

Byte-Pair Encoding (BPE)

"Lond on"
"Lond r e s "
"Lond r a"

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on
rule #2: (L, on) -> Lon
rule #3: (Lon, d) -> Lond

(Lond, on) 1

(Lond, r) 2

(r, e) 1

(e, s) 1

Byte-Pair Encoding (BPE)

“Lond on”
“Lond r e s ”
“Lond r a”

{L, o, n, d, r, e, s, a, on, Lon, Lond}

rule #1: (o, n) -> on
rule #2: (L, on) -> Lon
rule #3: (Lon, d) -> Lond

(Lond, on) 1

(Lond, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

Byte-Pair Encoding (BPE)

“<u>Lond</u> <u>on</u>”

“<u>Lond</u> r e s ”

“<u>Lond</u> r a”

(Lond, on) 1

(Lond, r) 2

(r, e) 1

(e, s) 1

(r, a) 1

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

Vocabulary

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

Merge Rules

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

rondLon

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

r <u>on</u> d L <u>on</u>

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

r <u>on</u> d L <u>on</u>

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

r <u>on</u> d <u>Lon</u>

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

r <u>on</u> d <u>Lon</u>

No merges for #3 and #4

Byte-Pair Encoding (BPE)

{L, o, n, d, r, e, s, a, on, Lon, Lond, Londr}

rule #1: (o, n) -> on

rule #2: (L, on) -> Lon

rule #3: (Lon, d) -> Lond

rule #4: (Lond, r) -> Londr

rondLon

r <u>on</u> d <u>Lon</u>

Byte-Pair Encoding (BPE)

{0: L, 1: o, 2:n, 3: d,

4: r, 5: e, 6: s,

7: a, 8: on, 9: Lon,

10: Lond, 11: Londr}

rondLon

r <u>on</u> d <u>Lon</u>

[4, 8, 3, 9]

Why is BPE important?

	GPT-2	Mistral
“London”	[23421]	[4222]
“Londres”	[43, 623, 411]	[26432]
“Londra”	[43, 623, 430]	[17681, 520]

BPE is data sensitive

90% general web	80% general web
5% programming	10% programming
5% math	10% math
st raw berry	str aw berry

BPE is data sensitive

90% general web

5% programming

5% math

+ emoji

build tokenizer

general web

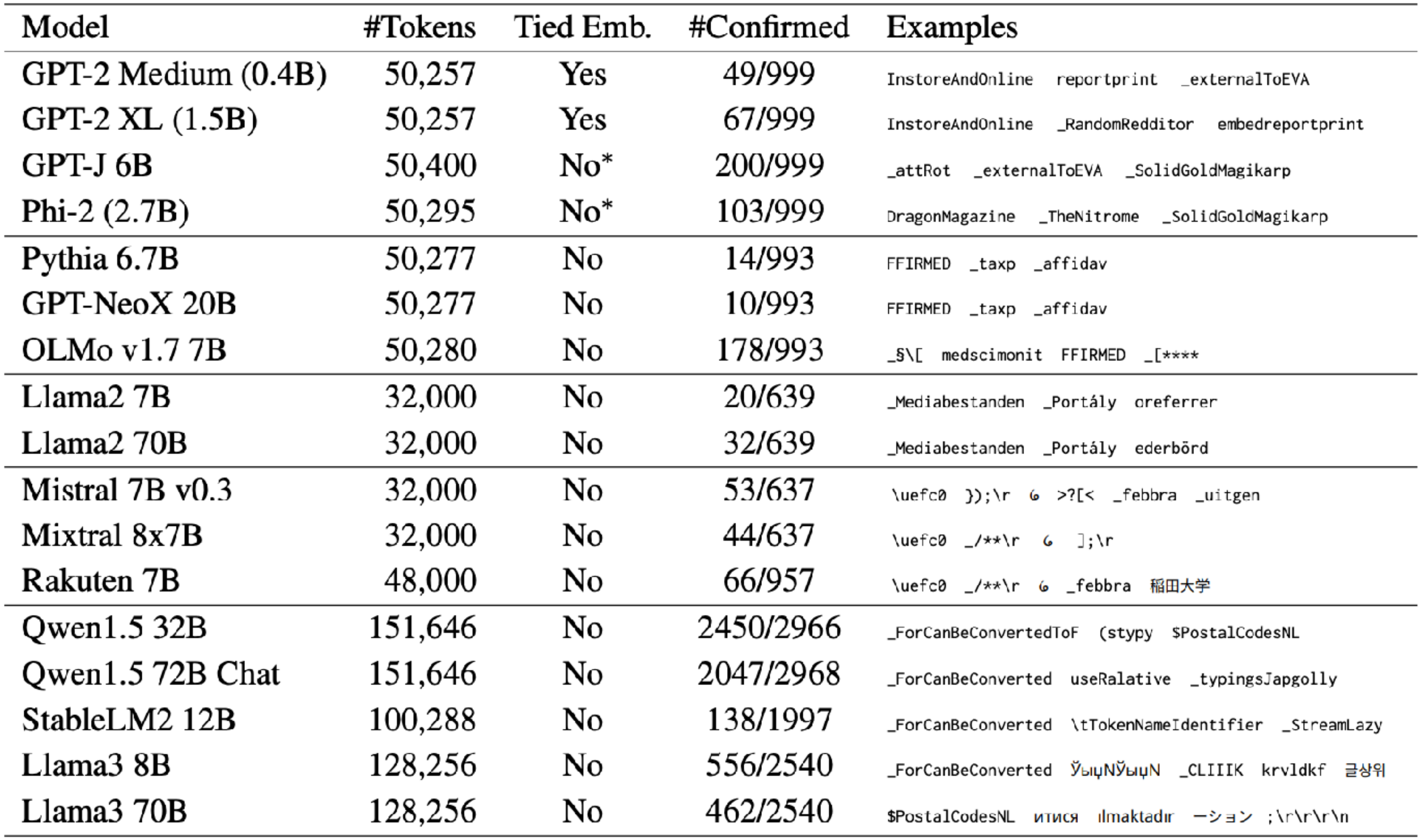
programming

math

(remove all emojis)

pretraining data

BPE is data sensitive



Model	#Tokens	Tied Emb.	#Confirmed	Examples
GPT-2 Medium (0.4B)	50,257	Yes	49/999	InstoreAndOnline reportprint _externalToEVA
GPT-2 XL (1.5B)	50,257	Yes	67/999	InstoreAndOnline _RandomRedditor embedreportprint
GPT-J 6B	50,400	No*	200/999	_attRot _externalToEVA _SolidGoldMagikarp
Phi-2 (2.7B)	50,295	No*	103/999	DragonMagazine _TheNitrome _SolidGoldMagikarp
Pythia 6.7B	50,277	No	14/993	FFIRMED _taxp _affidav
GPT-NeoX 20B	50,277	No	10/993	FFIRMED _taxp _affidav
OLMo v1.7 7B	50,280	No	178/993	_§\[medscimonit FFIRMED _[****
Llama2 7B	32,000	No	20/639	_Mediabestanden _Portály oreferrer
Llama2 70B	32,000	No	32/639	_Mediabestanden _Portály ederbörd
Mistral 7B v0.3	32,000	No	53/637	\uefc0 });\r ๖ >?[< _febbra _uitgen
Mixtral 8x7B	32,000	No	44/637	\uefc0 _/**\r ๖];\r
Rakuten 7B	48,000	No	66/957	\uefc0 _/**\r ๖ _febbra 稲田大学
Qwen1.5 32B	151,646	No	2450/2966	_ForCanBeConvertedToF (stypy $PostalCodesNL
Qwen1.5 72B Chat	151,646	No	2047/2968	_ForCanBeConverted useRalative _typingsJapgolly
StableLM2 12B	100,288	No	138/1997	_ForCanBeConverted \tTokenNameIdentifier _StreamLazy
Llama3 8B	128,256	No	556/2540	_ForCanBeConverted ЎыџNЎыџN _CLIIIK krvldkf 글상위
Llama3 70B	128,256	No	462/2540	$PostalCodesNL итися ılmaktadır ーション ;\r\r\r\n

Thank you!