Few-shot learning, tokeny i dalsze kwestie wstępne

Paweł Rychlikowski

Instytut Informatyki UWr

17 października 2024

Przypomnienie: 4 poziomy modelu językowego

- **Poziom 0**: aplikacja
- **Poziom 1**: API generujące teksty
- **Poziom 2**: rozkład prawdopodobieństwa na tokenach
- **Poziom 3**: sieć neuronowa

Przypomnienie: 4 poziomy modelu językowego

- **Poziom 0**: aplikacja
- **Poziom 1**: API generujące teksty
- **Poziom 2**: rozkład prawdopodobieństwa na tokenach
- **Poziom 3**: sieć neuronowa

Jeszcze o few-shots learning

(ważna technika z poziomu 1)

Few shots-learning in LMs

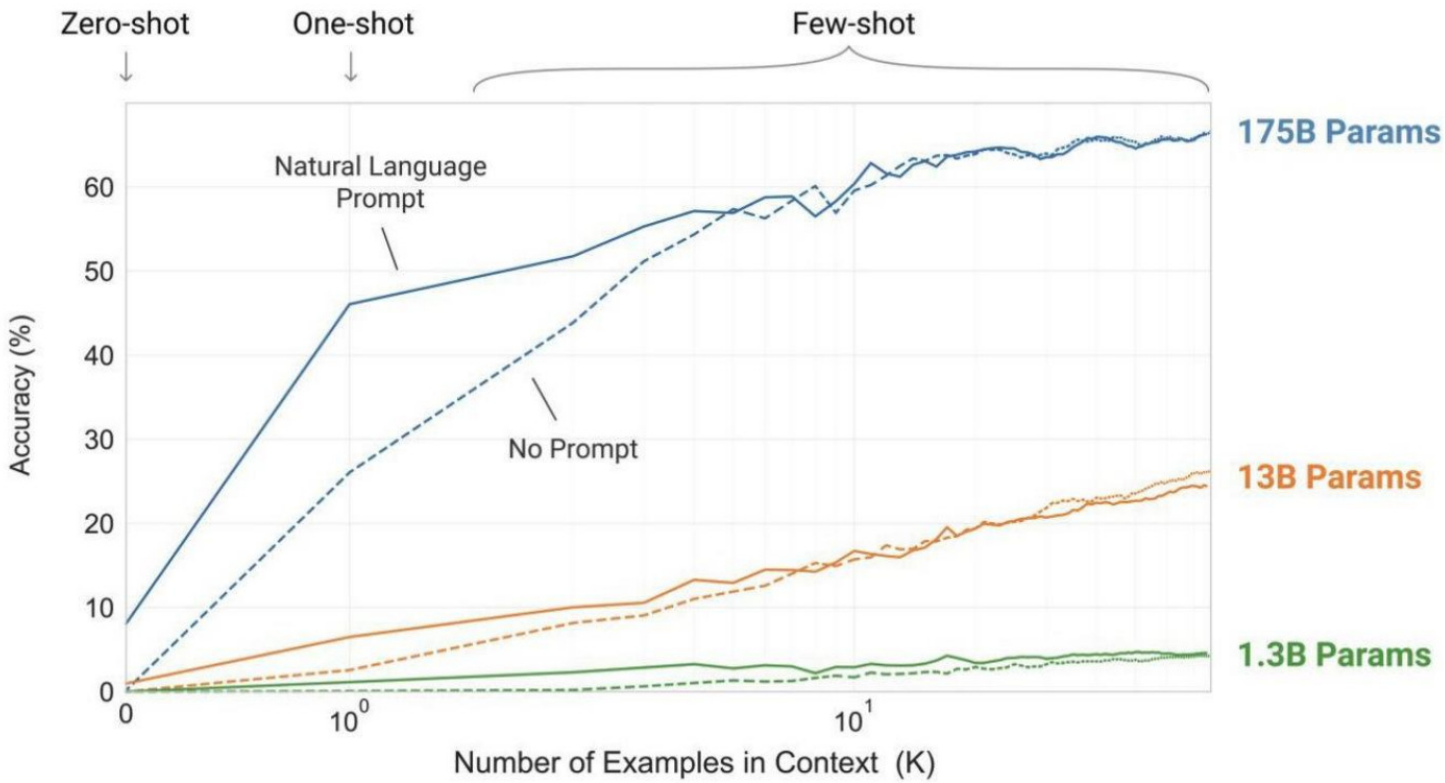
Główna idea:

- Wybrać kilka **demonstracji**
- i wkleić je do prompta.

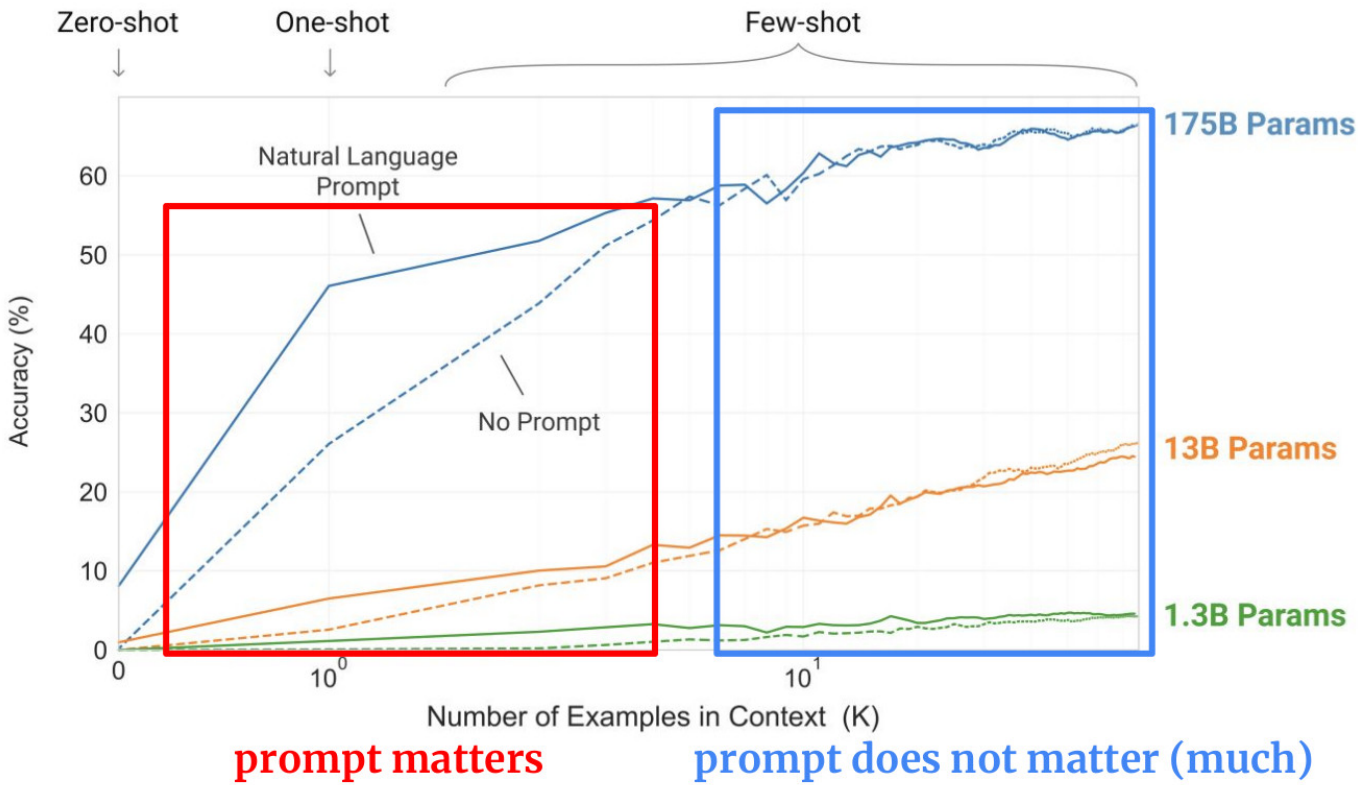
Kilka pytań:

- Jak wybrać przykłady do promptu (czy wszystkie?)
- Czy warto oprócz przykładów podać opis zadania?
- Jaka kolejność przykładów?
- Czy formatowanie ma znaczenie?

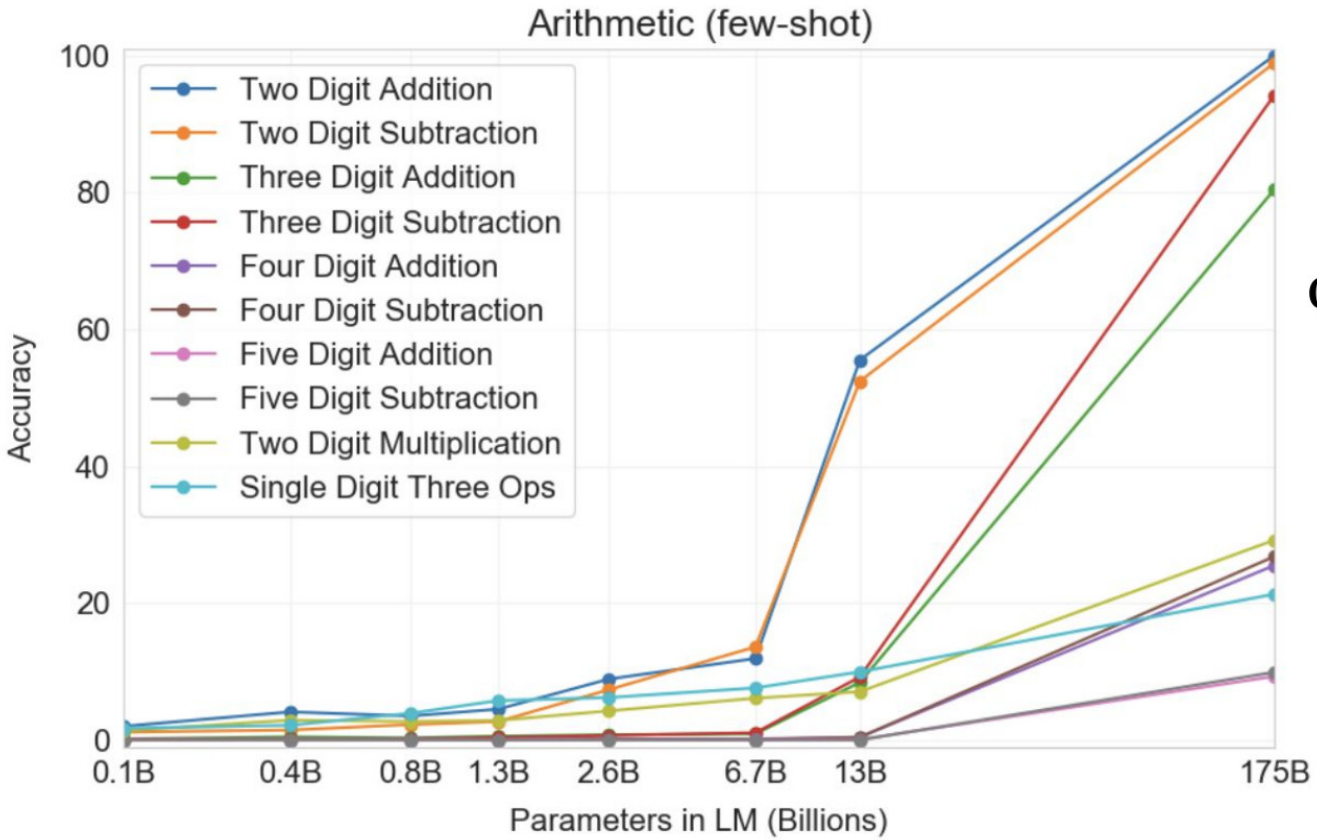
Emergent Capability - In-Context Learning



Larger Models Learn Better In-Context



Pushing GPT-3 Further: Arithmetic



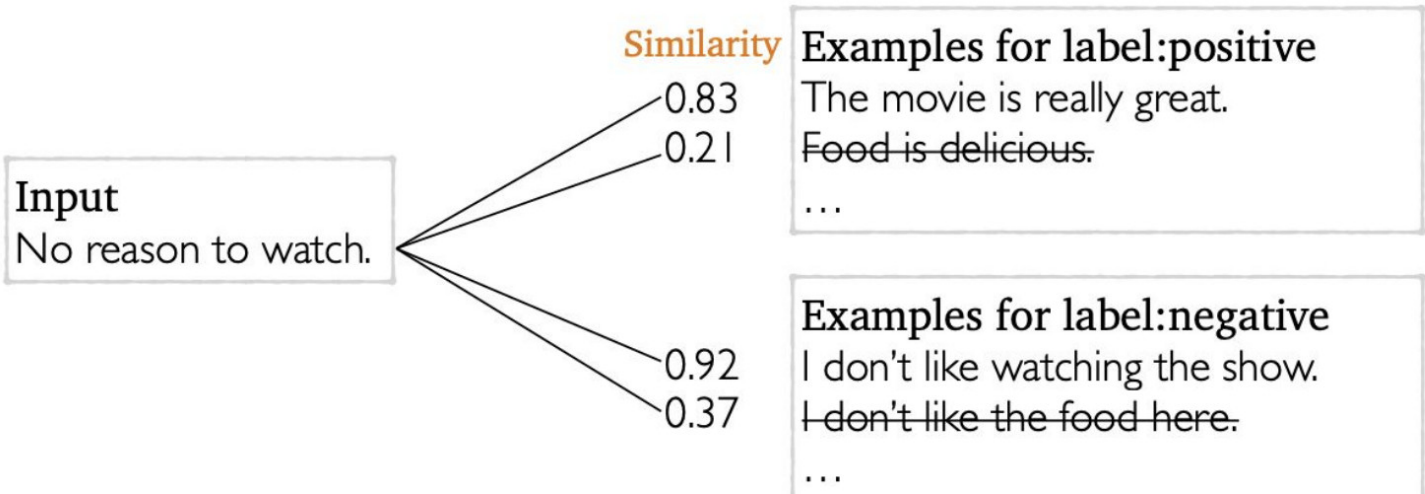
Observations:

1. Scale is important!
2. >1 operation or >3 digit numbers are much harder

Q: What is 48 plus 76?
A: 124.

Demonstracje pasujące do konkretnego przypadku wejściowego

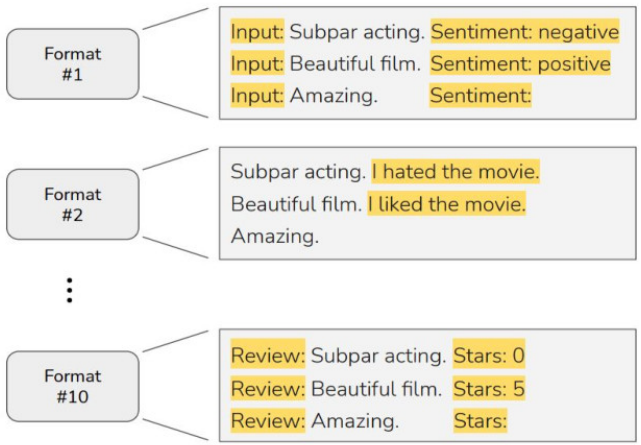
- Prosta intuicja: przykłady powinny być podobne do danych wejściowych
- Jak mierzyć podobieństwo – o tym będzie cały wykład, na razie możemy przyjąć, że jest to dowolna heurystyczna procedura, zliczająca powtarzające się wyrazy, mierząca odległość edycyjną, etc
- (dla zaawansowanych: podobieństwo cosinusowe osadzeń wyliczonych przez pretrenowany model typu BERT)

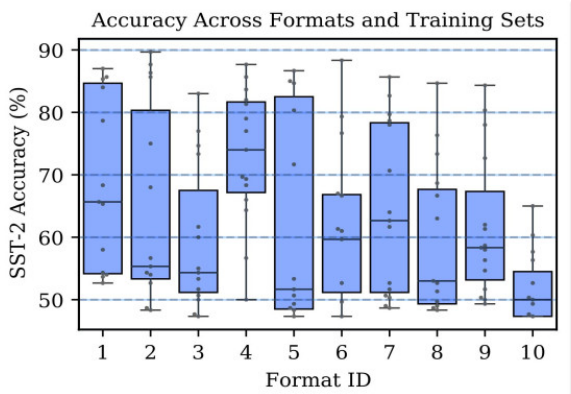


How important is the structure of the prompt for in-context learning?

Components of a prompt

1. **Prompt format**
2. Training example selection
3. Training example permutation



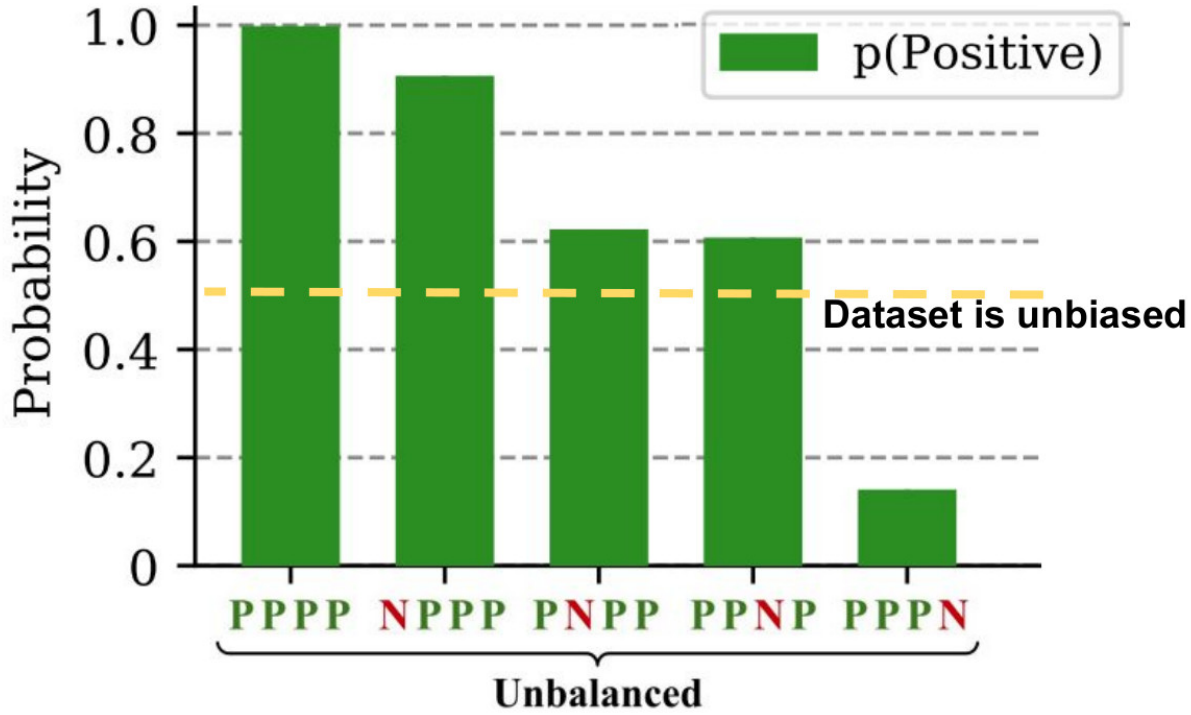


In-context learning is highly sensitive to prompt format

What causes this sensitivity?

Three main reasons

1. **Majority label bias**
2. Common token bias
3. Recency bias

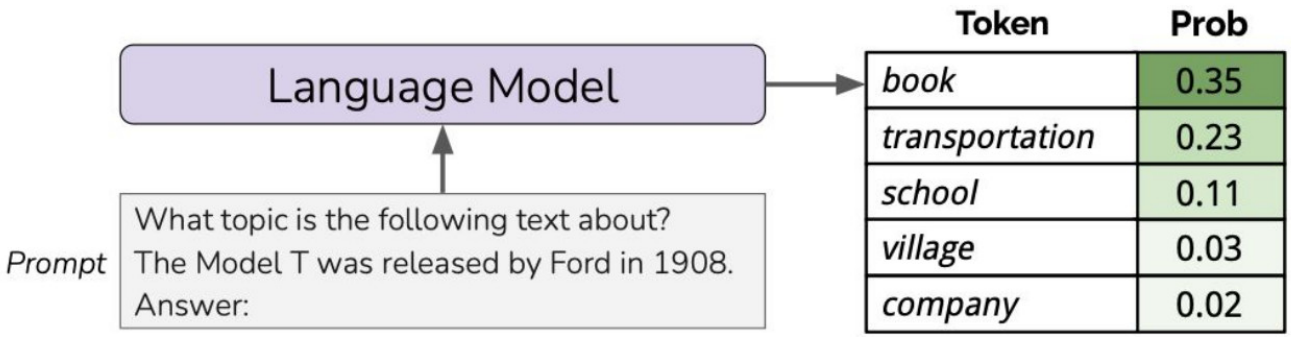


1. **Model prefers to predict positive when the majority labels is "P/Positive"**
2. **Surprising because the validation dataset is balanced!**

What causes this sensitivity?

Three main reasons

1. Majority label bias
2. **Common token bias**
3. Recency bias



	Token	Web (%)	Label (%)	Prediction (%)
❌	*book*	0.026	9	29
✅	*transportation*	0.0000006	9	4

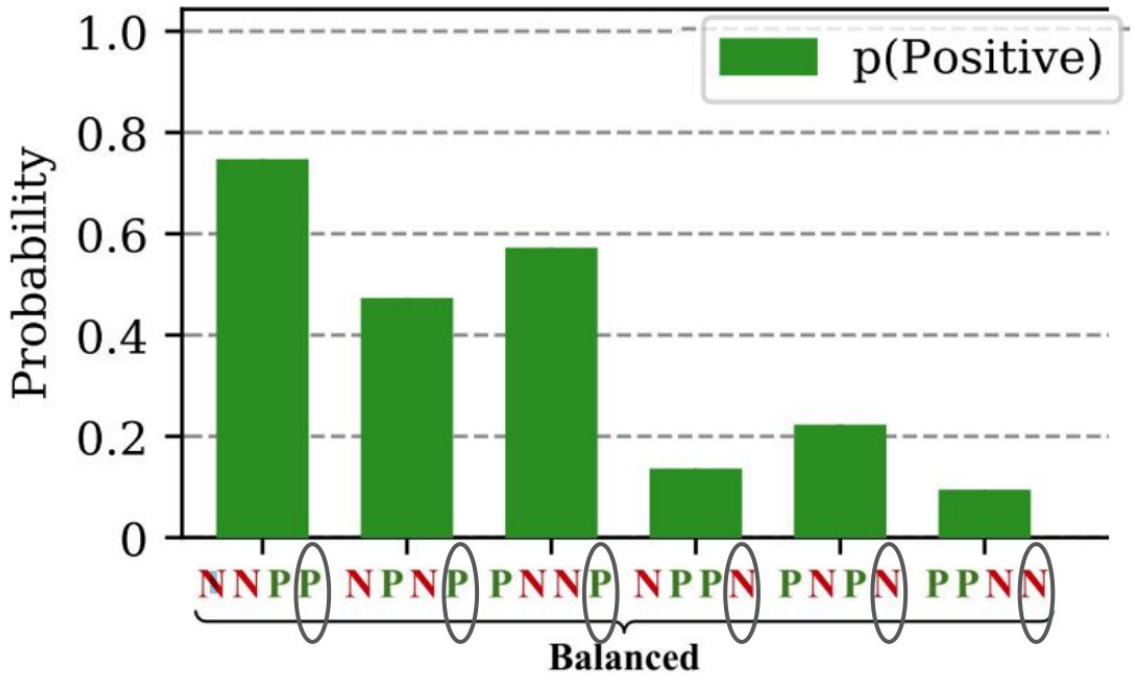
Model is biased towards predicting the incorrect frequent token "book" even when both "book" and "transportation" are equally likely labels in the dataset

20

What causes this sensitivity?

Three main reasons

1. Majority label bias
2. Common token bias
3. **Recency bias**



1. **Model is heavily biased towards the most recent label**
2. **Again, dataset is balanced!**

Przypomnienie: 4 poziomy modelu językowego

- **Poziom 0**: aplikacja
- **Poziom 1**: API generujące teksty
- **Poziom 2**: rozkład prawdopodobieństwa na tokenach
- **Poziom 3**: sieć neuronowa

Poziom 2 (przypomnienie)

- Prawdopodobieństwo sekwencji tokenów można obliczyć następująco:

$$P(w_1 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_n|w_1 \dots w_{n-1})$$

Popatrzmy na kod obliczający prawdopodobieństwo tekstu.

Ważna uwaga

Ten kod **jest deterministyczny!**

Wykorzystanie prawdopodobieństwa zdania

Gdzie może być wykorzystane:

- Zadania klasyfikacji: co jest bardziej prawdopodobne?
  [tekst-opinii-klienta] **Polecam!**
  [tekst-opinii-klienta] **Nie polecam!**

  Problem do rozwiązania: dłuższe teksty mają 'pod górkę'.
- Ustalanie autorstwa tekstu (X, czy Y):
  potrzebne są nam wówczas dwa modele językowe, jeden dla X-a, drugi dla Y-ka)
- Automatyczne poprawianie błędów (literówki, gramatyka, poprawianie szyku zdania)

Tokenizacja

Definicja

Tokenizacja jest zamianą ciągu znaków na odpowiadający mu ciąg tokenów.

Decyzja, czy dany ciąg jest jednym tokenem, czy wymaga podziału nie zawsze jest oczywista!

Tradycyjna tokenizacja w NLP

Wariant 1

Wykonujemy operację split na każdym wierszu
Wada: Przyklejona interpunkcja!

Wariant 2

Każdy znak interpunkcyjny otaczamy (wirtualnie) spacjami, następnie wykonujemy operację split.

- For every punctuation character c, do
  `s = s.replace(c, ' ' + c + ' ')`
- Return `s.split()` or `s.lower().split()`

Wada (?): yahoo! albo F-16

Tokenizacja (cd)

Wariant 3

Uznajemy, że ktoś to rozwiązał i znajdujemy bibliotekę (np. NLTK, spaCy, również frameworki neuronowe jak Pytorch i Keras) i korzystamy z bibliotecznego tokenizatora