Few-shot learning, tokeny i dalsze kwestie wstępne

Paweł Rychlikowski

Instytut Informatyki UWr

22 października 2025

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.

Jeszcze o few-shots learning

(ważna technika z poziomu 1)

Few shots-learning in LMs

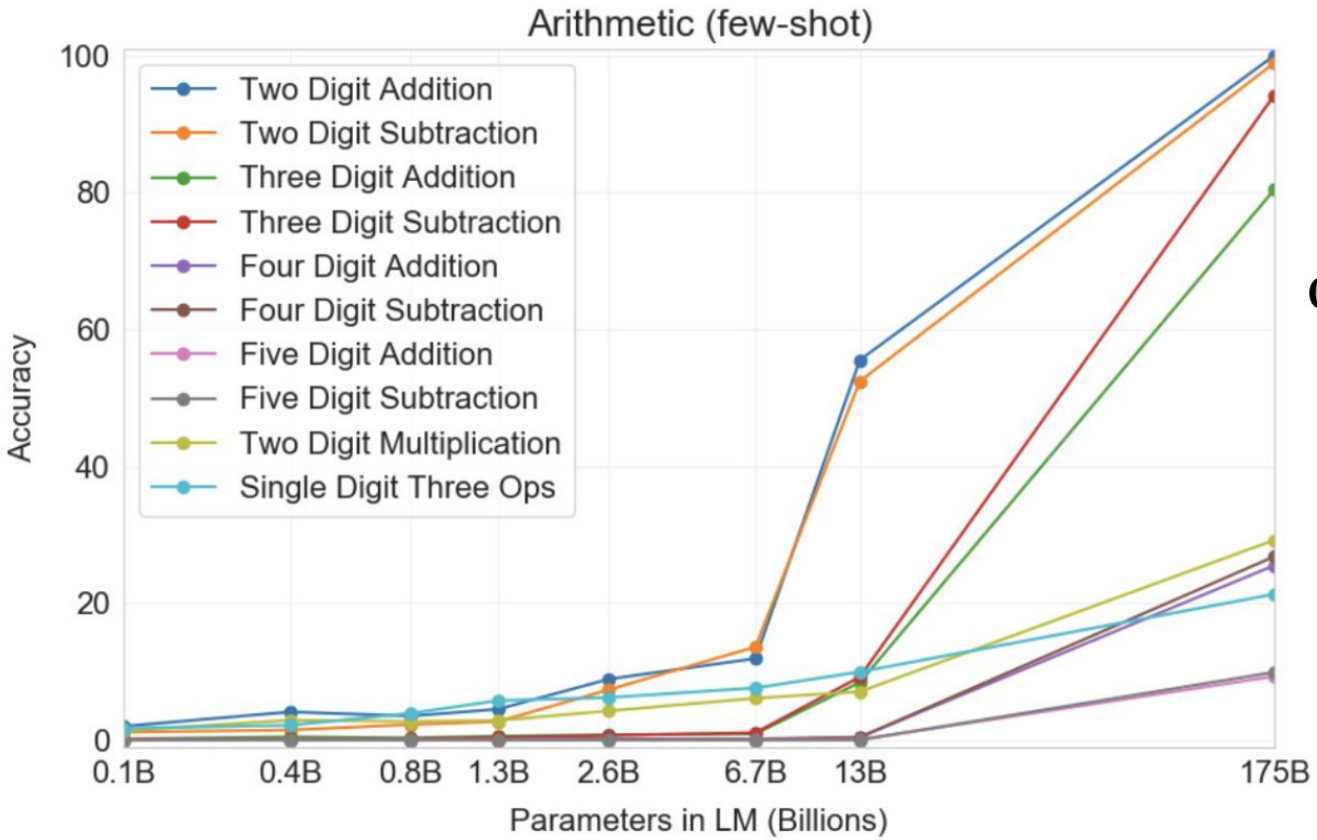
Główna idea:

- 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?

Pushing GPT-3 Further: Arithmetic



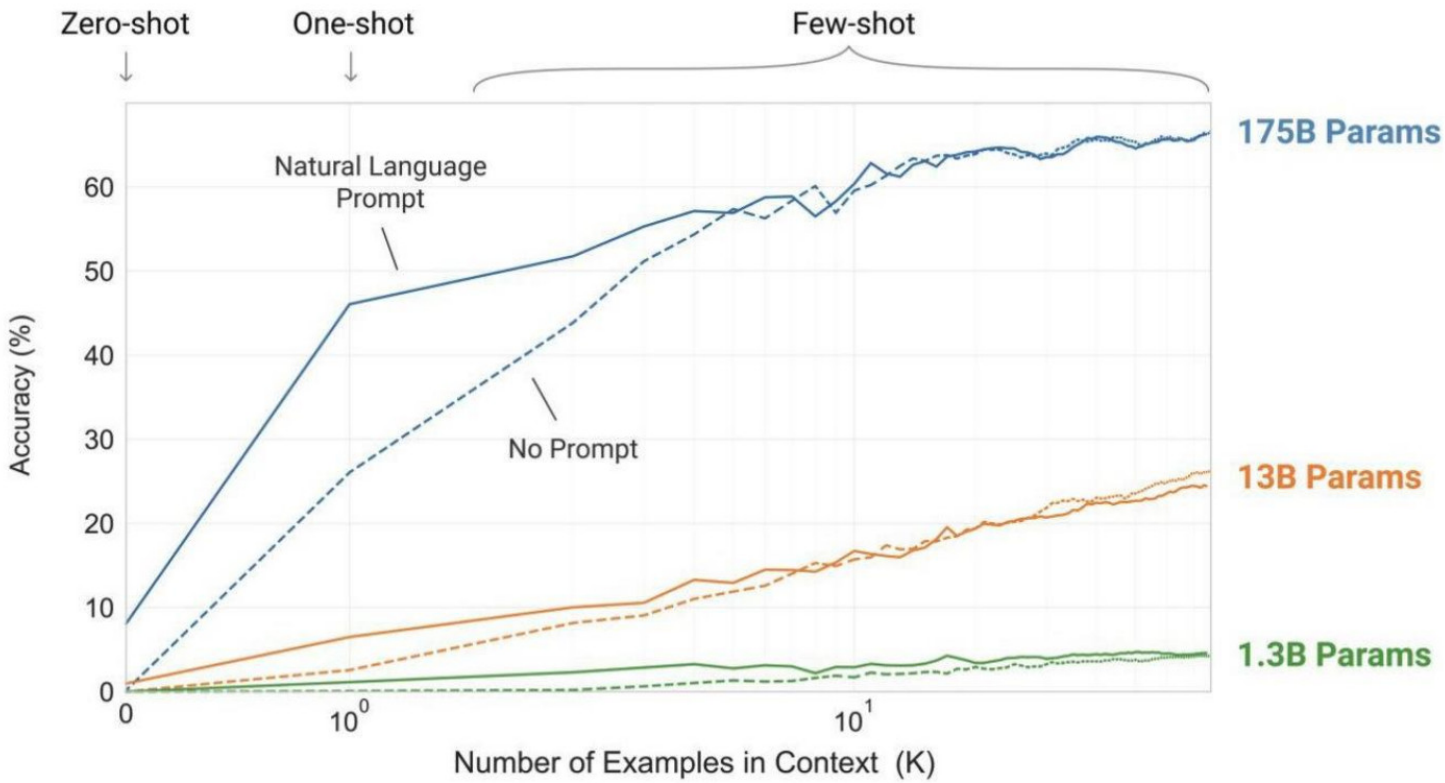
Observations:

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

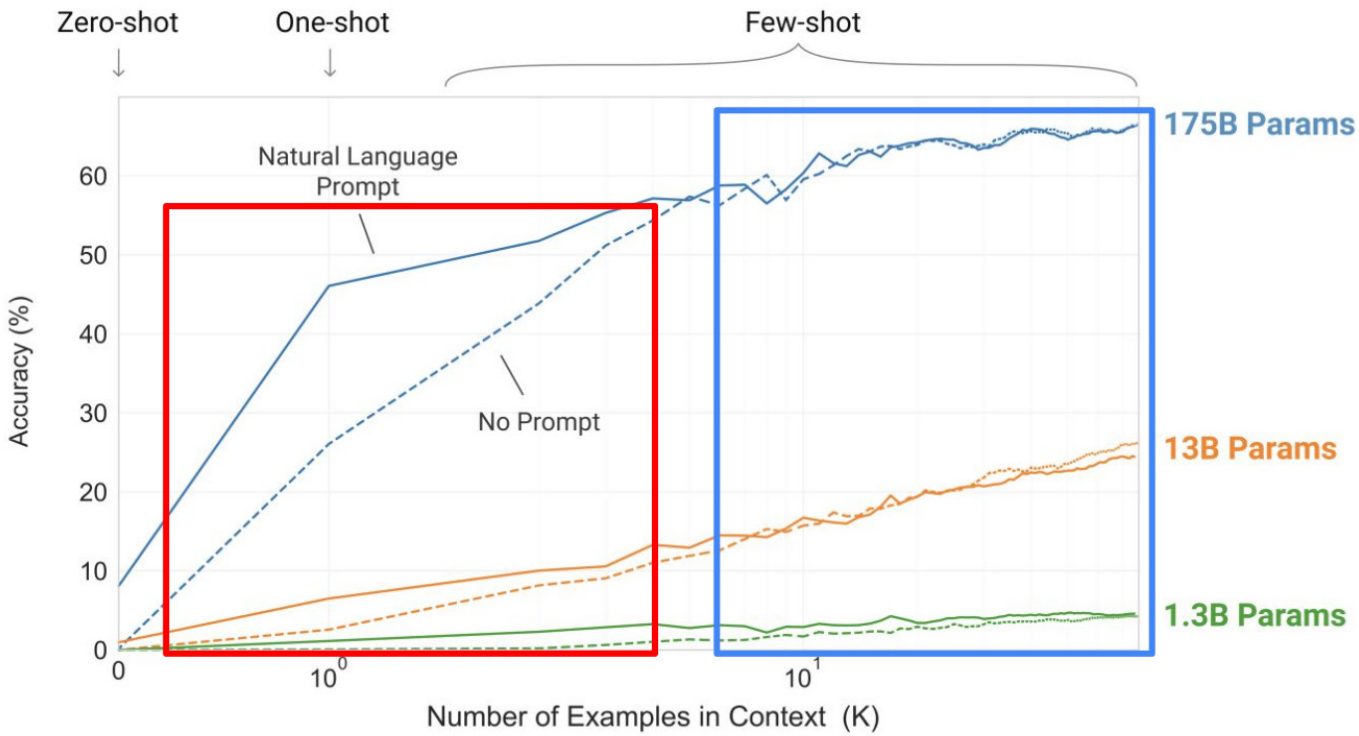
Q: What is 48 plus 76?
A: 124.

63

Emergent Capability - In-Context Learning



Larger Models Learn Better In-Context



Demonstracje pasujące do konkretnego przypadku wejściowego

- Prosta intuicja: przykłady powinny być podobne *wejścia*

Demonstracje pasujące do konkretnego przypadku wejściowego

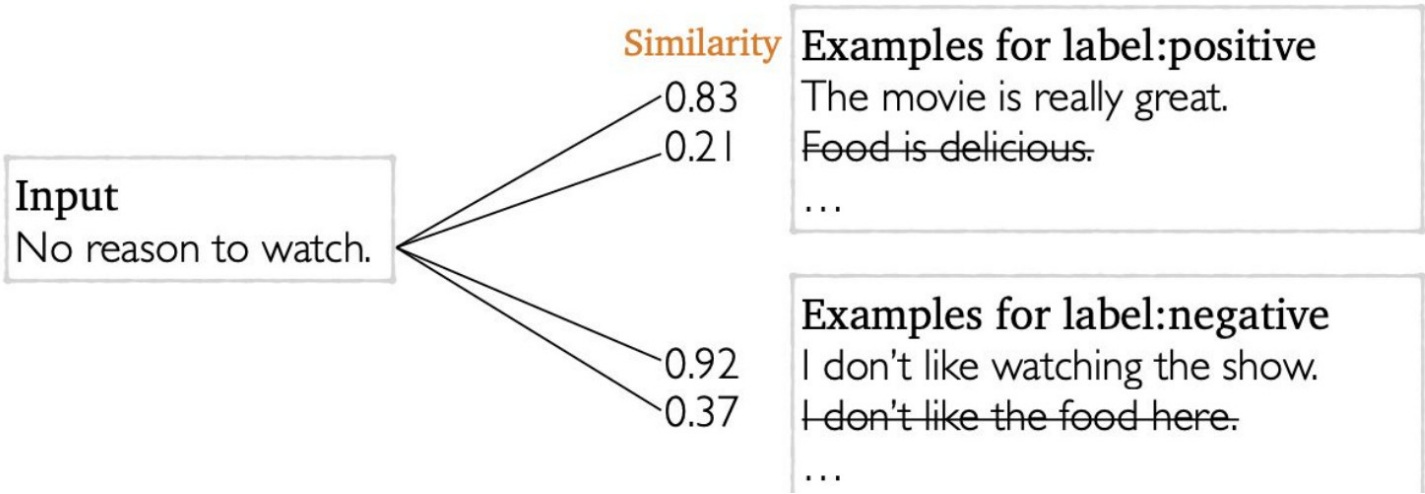
- Prosta intuicja: przykłady powinny być podobne *wejścia*
- Jak mierzyć podobieństwo – o tym będzie cały wykład, na razie możemy przyjąć, że jest to dowolna heurystyczna procedura, zliczająca powtarzające się wyrazy, mierząca odległość edycyjną, patrząca na początek pytania (W którym roku), etc

Demonstracje pasujące do konkretnego przypadku wejściowego

- Prosta intuicja: przykłady powinny być podobne *wejścia*
- 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, patrząca na początek pytania (W którym roku), etc
- Dla zaawansowanych: podobieństwo cosinusowe osadzeń wyliczonych przez pretrenowany model typu BERT

Demonstracje pasujące do konkretnego przypadku wejściowego

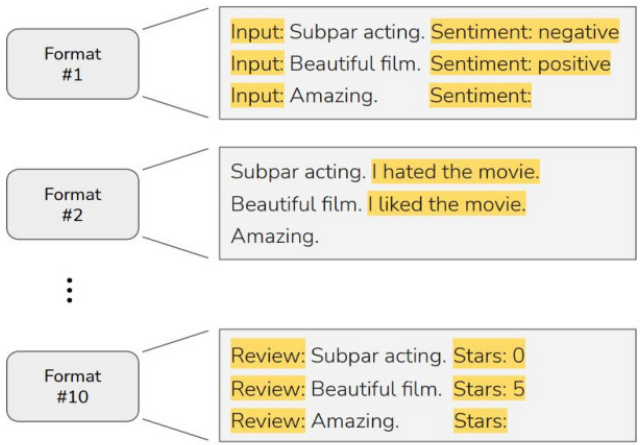
- Prosta intuicja: przykłady powinny być podobne *wejścia*
- Jak mierzyć podobieństwo – o tym będzie cały wykład, na razie możemy przyjąć, że jest to dowolna heurystyczna procedura, zliczająca powtarzające się wyrazy, mierząca odległość edycyjną, patrząca na początek pytania (W którym roku), etc
- Dla zaawansowanych: podobieństwo cosinusowe osadzeń wyliczonych przez pretrenowany model typu BERT

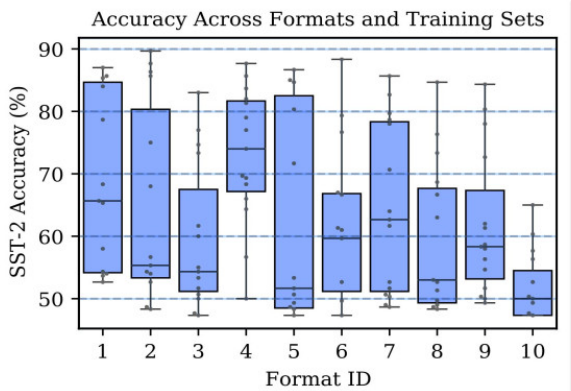


How important is the structure of the prompt for in-context learning?

Components of a prompt

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



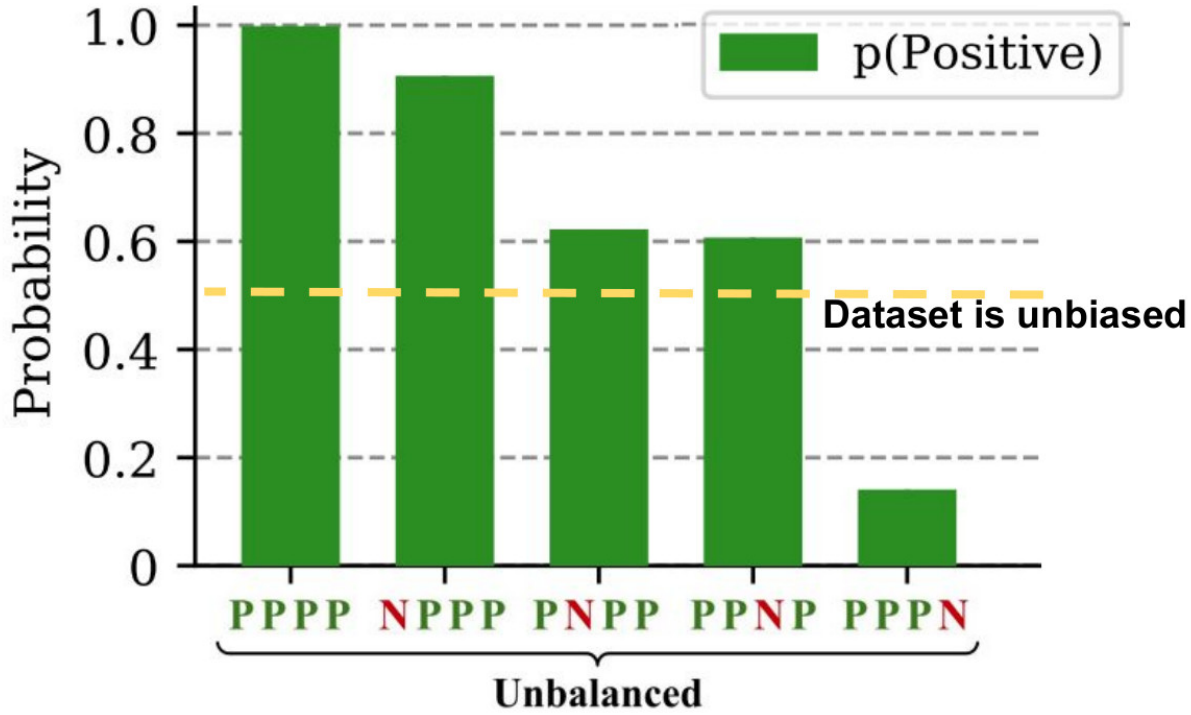


In-context learning is highly sensitive to prompt format

What causes this sensitivity?

Three main reasons

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



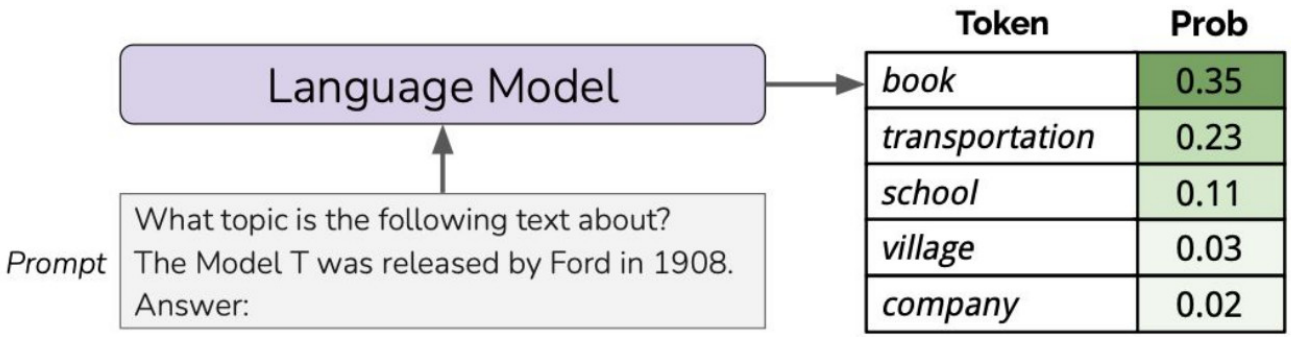
1. **Model prefers to predict positive when the majority labels is "P/Positive"**
2. **Surprising because the validation dataset is balanced!**

19

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

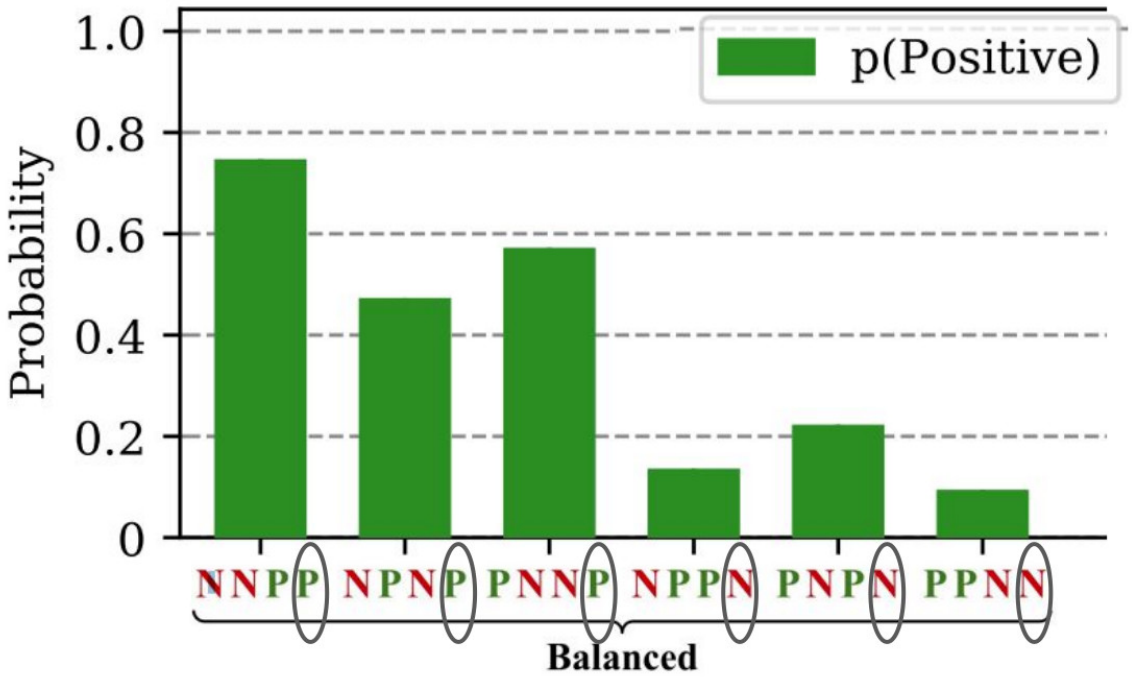
Model is biased towards predicting the incorrect frequent token "book" even when both "book" and "transportation" are equally likely labels in the dataset

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

Tokeny

- Prawdopodobieństwo sekwencji tokenów można obliczyć następująco:

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

Tokeny

- Prawdopodobieństwo sekwencji tokenów można obliczyć następująco:

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

Ale czym są te tokeny?

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!

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()`

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), czy biblikoteka transformers i korzystamy z bibliotecznego tokenizatora

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!**

Wykorzystanie prawdopodobieństwa zdania

Gdzie może być wykorzystane:

- Zadania klasyfikacji: co jest bardziej prawdopodobne?
  [tekst-opinii-klienta] **Polecam!**
  [tekst-opinii-klienta] **Nie polecam!**
- Problemy do rozwiązania:
  - dłuższe teksty mają 'pod górkę'.
  - Nawet jak teksty są równej długości, to któryś z nich jest bardziej prawdopodobny

Wykorzystanie prawdopodobieństwa zdania

Gdzie może być wykorzystane:

- Zadania klasyfikacji: co jest bardziej prawdopodobne?
  [tekst-opinii-klienta] **Polecam!**
  [tekst-opinii-klienta] **Nie polecam!**
- Problemy do rozwiązania:
  - dłuższe teksty mają 'pod górkę'.
  - Nawet jak teksty są równej długości, to któryś z nich jest bardziej prawdopodobny
- Przykładowe rozwiązanie: ustalić optymalny próg na różnice log-prawdopodobieństw

Jeszcze o tokenach

Czasem pomija się tokenizację, traktując język np. jako:

- Ciąg znaków (ASCII, Unicode)
- Ciąg bajtów (kodowanie utf-8)

Jeszcze o tokenach

Czasem pomija się tokenizację, traktując język np. jako:

- Ciąg znaków (ASCII, Unicode)
- Ciąg bajtów (kodowanie utf-8)

Uwaga

Jak uczymy model od zera, nie należy bać sie własnej tokenizacji, wykorzystującej naszą wiedzę o dziedzinie:

- Jak stokenizować DNA?
- Jak stokenizować wzory chemiczne?
- Jak stokenizować partie szachów?

Jeszcze o tokenach

Czasem pomija się tokenizację, traktując język np. jako:

- Ciąg znaków (ASCII, Unicode)
- Ciąg bajtów (kodowanie utf-8)

Uwaga

Jak uczymy model od zera, nie należy bać sie własnej tokenizacji, wykorzystującej naszą wiedzę o dziedzinie:

- Jak stokenizować DNA?
- Jak stokenizować wzory chemiczne?
- Jak stokenizować partie szachów?
- Czasem wiedza lingwistyczna daje lepszą tokenizację niż generyczne algorytmy (dla standardowych tekstów, nie dla DNA).

Bytes Pair Encoding

W wielkim skrócie:

Bytes Pair Encoding

W wielkim skrócie:

a) Liczymy słowa w korpusie (czyli dużym, reprezentatywnym, zbiorze tekstów)
b) W słowach liczymy częstości par liter
   - Jeżeli abrakadabra występowało 15 razy, to zwiększamy licznik **ra** o 30.
c) Zamieniamy najczęstszą parę na nową (pseudo)literę
d) Czynności powtarzamy aż do otrzymania pożądanej liczby pseudoliter.

Każde słowo reprezentujemy jako ciąg pseudoliter (szczegóły na kolejnych sjaldach).

Byte Pair Encoding

- Originally a compression algorithm:
  - Most frequent byte pair ↦ a new byte.

Replace bytes with character ngrams

(though, actually, some people have done interesting things with bytes)

Rico Sennrich, Barry Haddow, and Alexandra Birch. ***Neural Machine Translation of Rare Words with Subword Units****. ACL 2016.*
https://arxiv.org/abs/1508.07909
https://github.com/rsennrich/subword-nmt
https://github.com/EdinburghNLP/nematus

Byte Pair Encoding

- A word segmentation algorithm:
  - Though done as bottom up clusering
  - Start with a unigram vocabulary of all (Unicode) characters in data
  - Most frequent ngram pairs $\mapsto$ a new ngram

19

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pairs ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d

Start with all characters in vocab

20

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pairs ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9

21

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pairs ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w **est**
3 w i d **est**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9

22

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pairs ↦ a new ngram

Dictionary

5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7

23

(Example from Sennrich)

Byte Pair Encoding

- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer (commonly Moses tokenizer for MT)
- Automatically decides vocab for system
  - No longer strongly “word” based in conventional way

Top places in WMT 2016!
Still widely used in WMT 2018

24

https://github.com/rsennrich/nematus

Modele N-gramowe

Definicja

N-gramem nazywamy ciąg kolejnych słów o długości N. 1-gramy to unigramy, 2-gramy to bigramy, 3-gramy to trigramy.

Za pomocą N-gramów tworzymy model języka, w którym staramy się przewidzieć kolejne słowo (N-te) na podstawie $N-1$ słów poprzednich.

Modele N-gramowe

Definicja

N-gramem nazywamy ciąg kolejnych słów o długości N. 1-gramy to unigramy, 2-gramy to bigramy, 3-gramy to trigramy.

Za pomocą N-gramów tworzymy model języka, w którym staramy się przewidzieć kolejne słowo (N-te) na podstawie $N-1$ słów poprzednich.

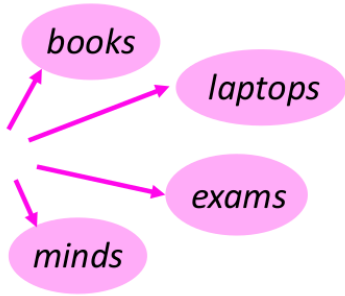
Uwaga

Na kolejnych slajdach (z wykładu na Stanfordzie, CS/...) powiemy krótko o modelach n-gramowych i próbkowaniu.

Language Modeling

- **Language Modeling** is the task of predicting what word comes next

the students opened their _____



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

- A system that does this is called a **Language Model**

Language Modeling

- You can also think of a Language Model as a system that assigns a probability to a piece of text

- For example, if we have some text $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the probability of this text (according to the Language Model) is:

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$

$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

This is what our LM provides

n-gram Language Models

- First we make a Markov assumption: $x^{(t+1)}$ depends only on the preceding n-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}}) \quad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(definition of conditional prob)}$$

- **Question:** How do we get these n-gram and (n-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(statistical approximation)}$$

24