Large Language Models wykorzystanie w tłumaczeniu

Kamil Schlagowski

Kacper Budniak

Co to LLM?

Jest to rodzaj systemu sztucznej inteligencji (AI), który jest w stanie generować tekst podobny do ludzkiego w oparciu o wzorce i relacje, których uczy się z ogromnych ilości danych. ~Nvidia

Jest to model uczenia maszynowego, którego celem jest przewidywanie i generowanie wiarygodnego języka. Na przykład autouzupełnianie jest modelem językowym. Modele te działają poprzez szacowanie prawdopodobieństwa wystąpienia tokena lub sekwencji tokenów w dłuższej sekwencji tokenów. ~Google

*Def - Nvidia

Co to LLM?

Rozważmy takie zdanie:

When I hear rain on my roof, I _____ in my kitchen.

Prawdopodobieństwa:

- cook soup 9.4%
- warm up a kettle 5.2%
- cower 3.6%
- nap 2.5%
- relax 2.2%

Czemu „Large”?

Nie jest dokładnie powiedziane od jakiego momentu model jest duży, ale na przykładzie modeli googla:

- BERT 110 milionów parametrów
- PaLM 340 miliardów parametrów

Parametry to wagi, których model nauczył się podczas szkolenia, używane do przewidywania następnego tokena w sekwencji. „Duży” może odnosić się albo do liczby parametrów w modelu, albo czasami do liczby słów w zbiorze uczącym.

Jak działają LLM

Surowe dane wejściowe

Input

Podział na mniejsze jednostki

Tokeniser

Przekład danych na język sieci neuronowej

Embedding

Encoder

Wyodrębnienie kontekstu

Attention Mechanism

Przypisanie rożnych wag tokenom

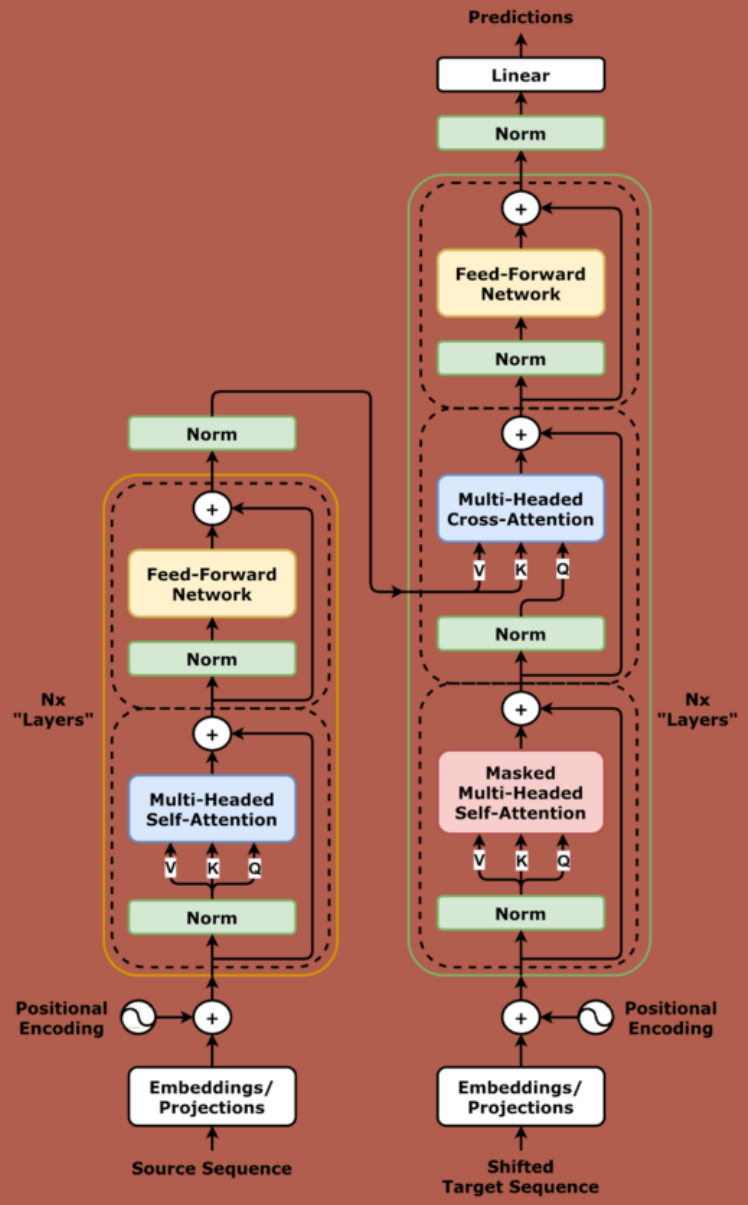
Decoder

Tłumaczenie na język ludzi

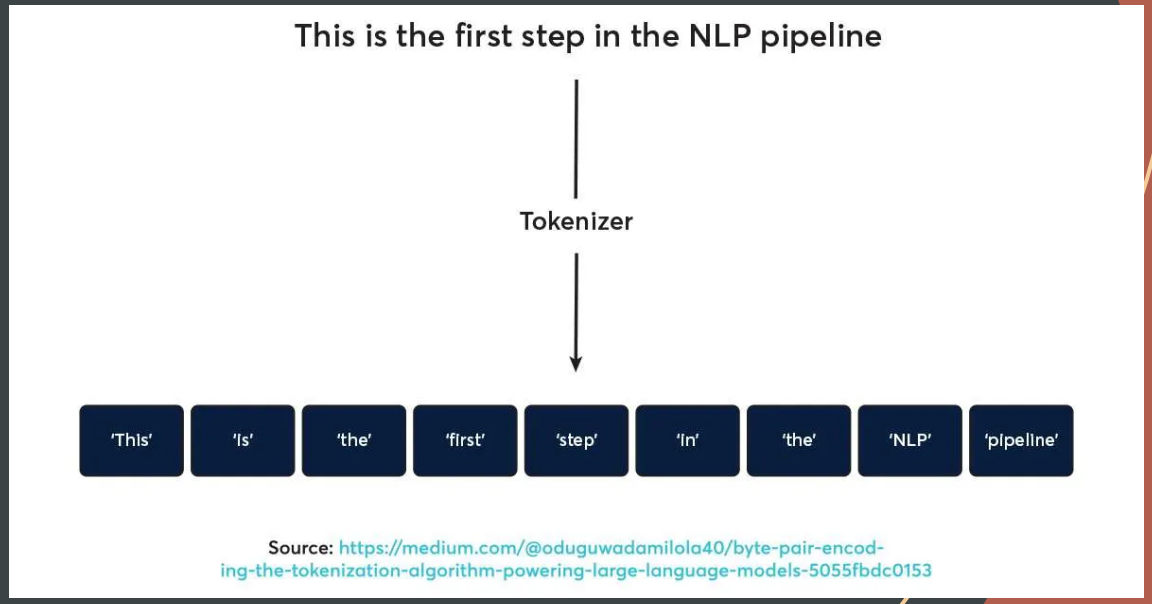
Output

Odpowiedź ostatecznie wyświetlona użytkownikowi

Transformer



Standardowa architektura Transformera, pokazująca po lewej Encoder, a po prawej Decoder



Tokenizacja

- Tokenizacja tekstu to proces podziału tekstu na mniejsze jednostki (zwane tokenami), które są podstawowymi elementami wejściowymi dla modeli językowych. Tokenujące modele językowe, takie jak LLM, przekształcają tekst w tokeny, aby lepiej przetwarzać i analizować język naturalny. Tokeny mogą być słowami, częściami słów, znakami lub nawet pojedynczymi literami, w zależności od zastosowanej metody tokenizacji.

Tokenizacja

Przykłady działania tokenizatora

Tekst: „The lighthouse!”

Słowa: [„ The”, „ lighthouse”, „!”]

Subword (BPE): [„ The”, „light”, „house”, „!”]

- Znaki: [„T”, „h”, „e”, „l”, „i”, „g”, „h”, „t”, „h”, „o”, „u”, „s”, „e”, „!”]

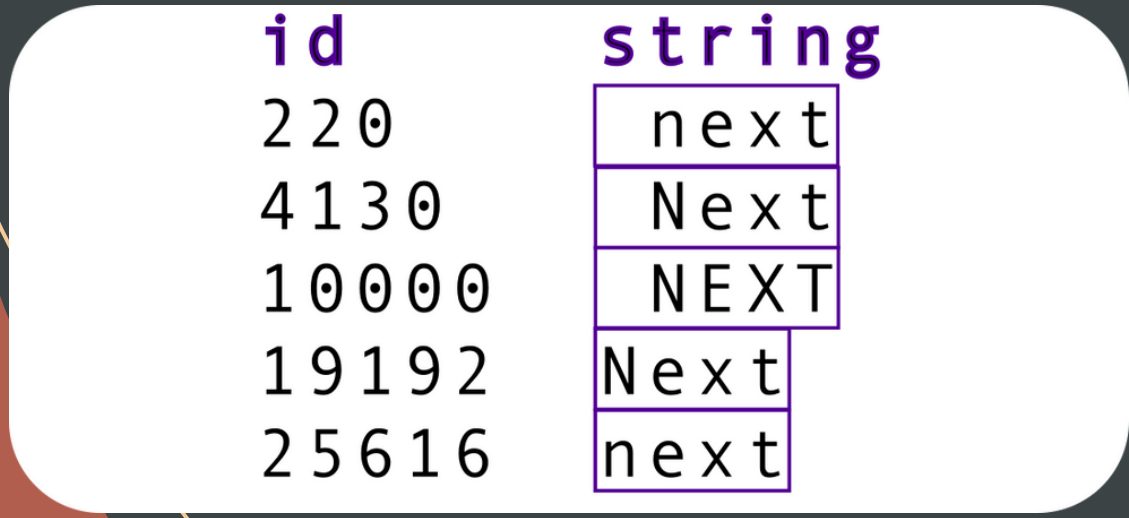
Tokenizacja

Następnie tokenizator patrzy na ciąg tekstu, dzieli go na ciągi tokenów i przekształca w listę identyfikatorów tokenów. Jest to zasadniczo tabela wyszukiwania/słownik: Na przykład:

Tekst : “ The cat went up the stairs.”

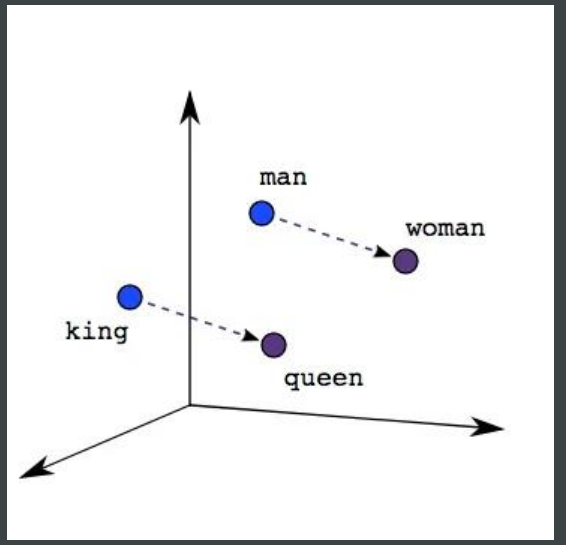
Podział: [“ The”, “ cat”, “ went”, “ up”, “ the”, “ stairs”, “.”]

Przydzielenie ID [20, 4758, 439, 62, 5,16745, 4]



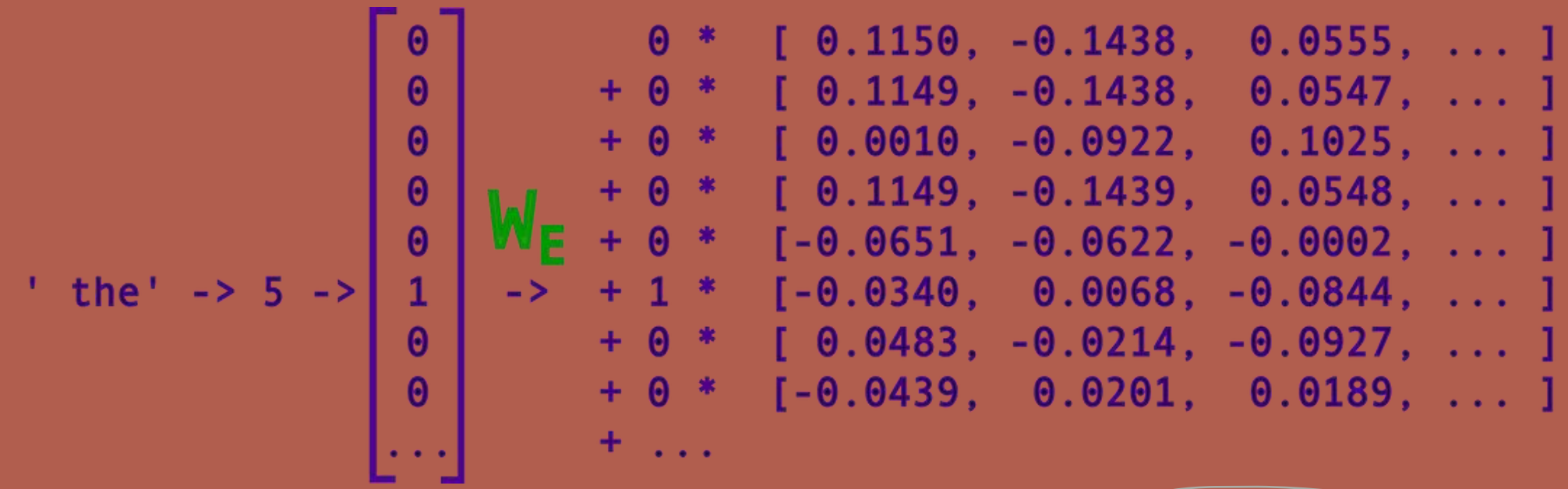
Embeddings

Numeryczne reprezentacje danych (najczęściej tekstowych), które umożliwiają ich przetwarzanie przez modele uczenia maszynowego. W kontekście języka naturalnego (NLP), embeddingi zamieniają słowa, frazy, czy inne jednostki językowe na wektory liczbowe w przestrzeni o określonej liczbie wymiarów. W tej przestrzeni słowa o podobnym znaczeniu znajdują się blisko siebie.



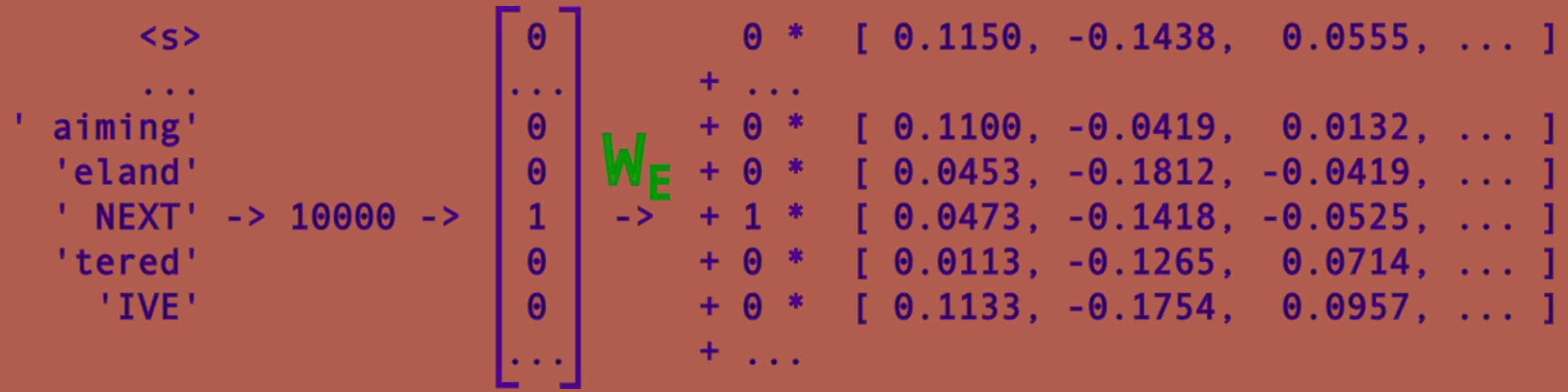
Embeddings

Wykorzystuje się macierz embeddingów do konwersji identyfikatorów tokenów w na wektory tokentów



Embeddings

Wykorzystuje się macierz embeddingów do konwersji identyfikatorów tokenów w na wektory tokentów



Podusmowanie Transformer

Transformery stosowane w tłumaczeniu maszynowym mają zwykle architekturę **encoder-decoder**, gdzie:

Encoder: Przetwarza zdanie w języku źródłowym, tworząc jego wektorową reprezentację.

Decoder: Na podstawie tej reprezentacji generuje tłumaczenie w języku docelowym

1. Wejście do modelu (Tokenizacja i reprezentacja)

1. Zdanie w języku źródłowym, np. „Kot goni mysz.”, jest poddawane tokenizacji:

   Przykładowe tokeny: ["Kot", "goni", "mysz", "."]

   Tokeny są zamieniane na liczby za pomocą słownika (vocabulary), np.:

   "Kot" → 123, "goni" → 456, "mysz" → 789.

2. Do każdego tokenu dodawane są wektory pozycyjne, które informują model o kolejności słów (ponieważ Transformer nie ma wbudowanego pojęcia sekwencji).

2. Przetwarzanie przez Encoder

1. Każdy token wejściowy jest zamieniany na **embedding** – wektor liczbowy reprezentujący znaczenie słowa.
2. Mechanizm **self-attention** w encoderze analizuje, jak każde słowo odnosi się do innych słów w zdaniu, uwzględniając kontekst:

   „Kot" ma silną zależność z „goni" (podmiot i czasownik).

   „goni" jest związane z „mysz" (czasownik i dopełnienie).
3. Wyjście z encodera to zestaw wektorów (jeden dla każdego tokenu), które reprezentują znaczenie słów i ich kontekst w całym zdaniu.

3. Przetwarzanie przez Decoder

1. Start dekodowania:

   Dekoder zaczyna od specjalnego tokenu startowego, np. [START], i generuje słowo wyjściowe token po tokenie.

2. Self-Attention w dekoderze:

   Dekoder analizuje już wygenerowane słowa, aby określić, co generować dalej. Na początku bierze pod uwagę tylko [START].

3. Przetwarzanie przez Decoder

3. Cross-Attention (Uwaga na encoder):

   Dekoder korzysta z informacji dostarczonej przez encoder (reprezentacja całego zdania źródłowego). Mechanizm ten pozwala dekoderowi "zapytać" encoder o odpowiednie fragmenty zdania źródłowego.

4. Generowanie tokenów:

   Dla każdego kroku dekoder przewiduje najbardziej prawdopodobny następny token w języku docelowym na podstawie:

   - Wcześniej wygenerowanych tokenów (np. słowa w tłumaczeniu do tej pory).
   - Wyjścia encodera (informacja o zdaniu źródłowym).

   Przykład: [START] → "The" → "cat" → "chases" → "the" → "mouse" → [END].

4. Generowanie tłumaczenia (Wyjście z dekodera)

Tokeny wygenerowane przez dekoder są zamieniane z powrotem na słowa za pomocą słownika modelu (vocabulary), np.:

[The, cat, chases, the, mouse] → „The cat chases the mouse.”

Kluczowe mechanizmy w tłumaczeniu Transformerów

a) Self-Attention

Mechanizm **self-attention** pozwala modelowi zrozumieć, które słowa w zdaniu są istotne dla siebie nawzajem. Działa w obu komponentach:

- W encoderze: „Kot" zwraca uwagę na „goni" i „mysz".
- W dekoderze: Wygenerowane słowo „cat" wpływa na to, że kolejne słowo to „chases".

b) Cross-Attention

Cross-attention umożliwia dekoderowi korzystanie z informacji zakodowanej przez encoder. Przykład:

- Podczas generowania „cat", dekoder zwraca uwagę na wektor „Kot" w wyjściu encodera.

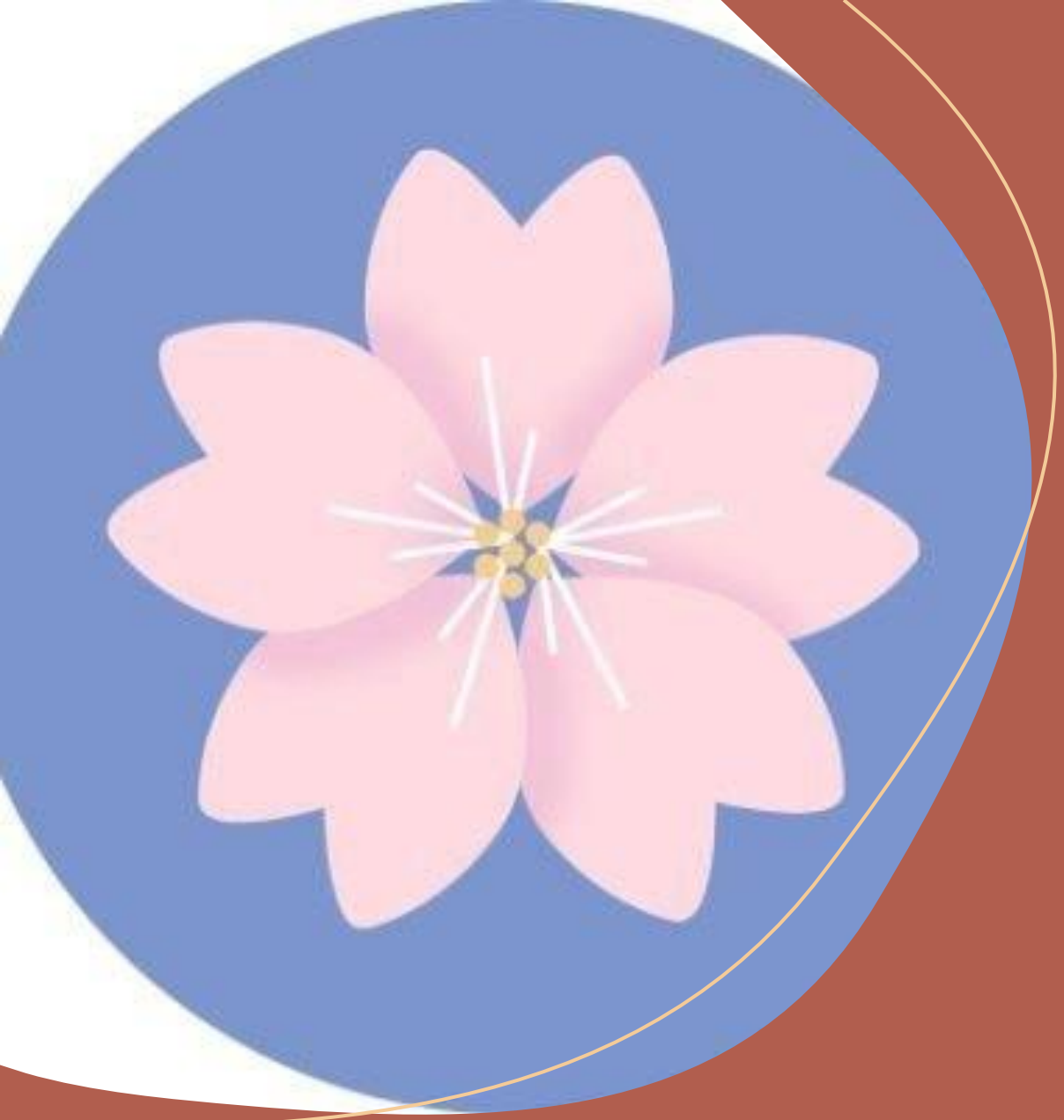
c) Predykcja tokenów

Dla każdego kroku dekoder oblicza prawdopodobieństwo wystąpienia każdego tokenu w słowniku i wybiera najbardziej prawdopodobny (lub korzysta z metod takich jak beam search, aby znaleźć najlepsze tłumaczenie).

Przegląd LLMów

LLaMA

Model językowy od Meta AI, dostępny dla społeczności badawczej.



BLOOM

Wielojęzyczny model językowy stworzony przez konsorcjum BigScience.

GPT-NeoX

Otwartoźródłowy model językowy rozwijany przez EleutherAI.

Falcon

Potężny model językowy open-source stworzony przez Technology Innovation Institute.

MPT

MPT - MosaicML Pretrained Transformer, skalowalny model językowy open-source.

Dziękujemy za uwagę

Bibliografia

- https://www.llama.com
- https://bigscience.huggingface.co/blog/bloom
- https://huggingface.co/docs/transformers/model_doc/gpt_neox
- https://www.eleuther.ai
- https://www.databricks.com/blog/mpt-7b
- https://falconllm.tii.ae