

Trening (nie tylko dla tekstu))

Paweł Rychlikowski

Instytut Informatyki UWr

14 stycznia 2026

OthelloGPT – wnioski

- Przewiduje poprawne ruchy z b. dużą dokładnością – nauczył się zasad

OthelloGPT – wnioski

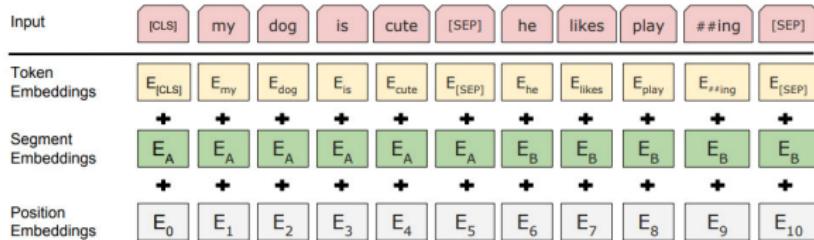
- Przewiduje poprawne ruchy z b. dużą dokładnością – nauczył się zasad
- Tworzy wewnętrzny model planszy (ew. na tablicy o **probingu**)

OthelloGPT – wnioski

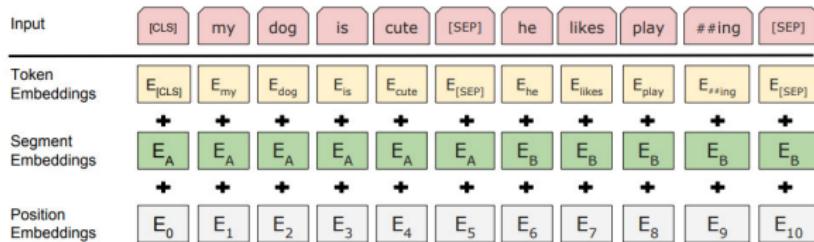
- Przewiduje poprawne ruchy z b. dużą dokładnością – nauczył się zasad
- Tworzy wewnętrzny model planszy (ew. na tablicy o **probingu**)

Czy umie grać sensownie? Autorzy nie stawiali tego pytania...

Jeszcze o kodowaniu pozycji

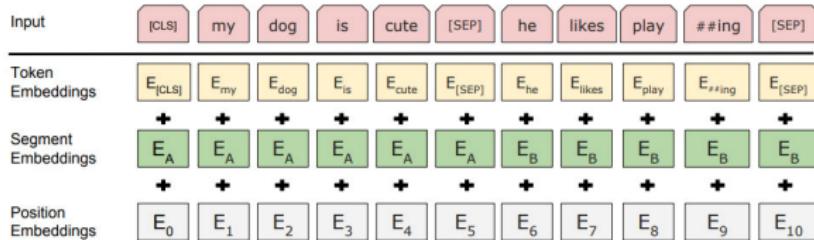


Jeszcze o kodowaniu pozycji



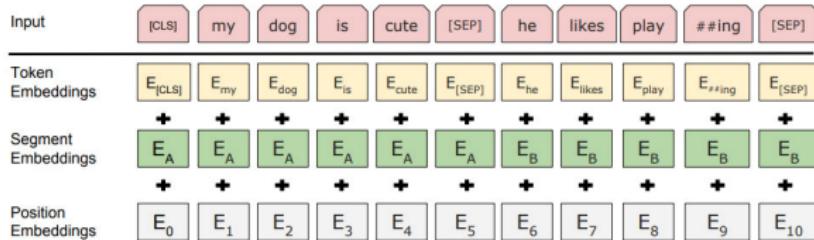
- Nie zawsze jest konieczne: w grafach mechanizm atencji określa przepływy informacji i relacje między węzłami

Jeszcze o kodowaniu pozycji



- Nie zawsze jest konieczne: w grafach mechanizm atencji określa przepływy informacji i relacje między węzłami
- Również w wersji GPT sieć jest w stanie się uczyć (trocę wolniej) również wówczas, gdy jedynie atencja określa relacje między słowami.

Jeszcze o kodowaniu pozycji



- Nie zawsze jest konieczne: w grafach mechanizm atencji określa przepływy informacji i relacje między węzłami
- Również w wersji GPT sieć jest w stanie się uczyć (trocę wolniej) również wówczas, gdy jedynie atencja określa relacje między słowami.
- Można określać **równocześnie** różne rodzaje pozycji (jak w BERT-cie, które słowo i które zdanie)

Zagrajmy w szachy!



Zagrajmy w szachy!



Zadanie

Zaprojektować wejście do transformera w zadaniu, w którym dla pozycji szachowej mamy powiedzieć, kto ma przewagę (dla uproszczenia załóżmy, że ruch przypada na białe).

Zagrajmy w szachy!



White to move. What is a strong continuation?

- Transformer będzie działał w stylu BERT-a, 64 pola + jedno dodatkowe na token **[CLS]**

Zagrajmy w szachy!



White to move. What is a strong continuation?

- Transformer będzie działał w stylu BERT-a, 64 pola + jedno dodatkowe na token [CLS]
- Tokenami będą

Zagrajmy w szachy!



- Transformer będzie działał w stylu BERT-a, 64 pola + jedno dodatkowe na token [CLS]
- Tokenami będą bierki (czarny skoczek, biały król, itd)
- Kodowanie pozycji (są różne opcje):

Zagrajmy w szachy!



- Transformer będzie działał w stylu BERT-a, 64 pola + jedno dodatkowe na token **[CLS]**
- Tokenami będą bierki (czarny skoczek, biały król, itd)
- Kodowanie pozycji (są różne opcje):
 - ▶ **[a5], [c3]**, ... mają osobne osadzenia (mamy nadzieję, że sieć się nauczy struktury 2D)
 - ▶ osadzenie dla **[a5]** to osadzenie dla **[a] + [5]**

Hipoteza: MLM for chess (?)

- Maskowany model językowy dla szachów może pozwolić modelowi otrzymać lepsze rozumienie gry (w porównaniu do sytuacji, gdy mamy nauczyć się jedynie, czy ruch jest dobry, czy zły).
- Zamiast: *Zgadnij co na tym polu mamy*: zgadnij, jak to **biała** bierka jest (albo **czarna**)



Hipoteza: MLM for chess (?)

- Maskowany model językowy dla szachów może pozwolić modelowi otrzymać lepsze rozumienie gry (w porównaniu do sytuacji, gdy mamy nauczyć się jedynie, czy ruch jest dobry, czy zły).
- Zamiast: *Zgadnij co na tym polu mamy*: zgadnij, jak to **biała** bierka jest (albo **czarna**)



GATO: w stronę ogólnej AI

A Generalist Agent

Jeden model, rozwiązujący wiele różnych zadań

- Różne benchmarkowe zadania, związane z wnioskowaniem w uproszczonych światach (kolorowe klocki tworzące wieże)

A Generalist Agent

Jeden model, rozwiązujący wiele różnych zadań

- Różne benchmarkowe zadania, związane z wnioskowaniem w uproszczonych światach (kolorowe klocki tworzące wieże)
- Modelowanie dialogu

A Generalist Agent

Jeden model, rozwiązujący wiele różnych zadań

- Różne benchmarkowe zadania, związane z wnioskowaniem w uproszczonych światach (kolorowe klocki tworzące wieże)
- Modelowanie dialogu
- Sokoban (łamigłówka z labiryntem i przesuwaniem skrzynek)

A Generalist Agent

Jeden model, rozwiązujący wiele różnych zadań

- Różne benchmarkowe zadania, związane z wnioskowaniem w uproszczonych światach (kolorowe klocki tworzące wieże)
- Modelowanie dialogu
- Sokoban (łamigłówka z labiryntem i przesuwaniem skrzynek)
- Gry zręcznościowe dla Atari

GATO: w stronę ogólnej AI

A Generalist Agent

Jeden model, rozwiązujący wiele różnych zadań

- Różne benchmarkowe zadania, związane z wnioskowaniem w uproszczonych światach (kolorowe klocki tworzące wieże)
- Modelowanie dialogu
- Sokoban (łamigłówka z labiryntem i przesuwaniem skrzynek)
- Gry zręcznościowe dla Atari
- Generowanie podpisów pod rysunkami
- Zadania związane z robotyką (ruch ramienia robota realizujący jakiś cel)

GATO: w stronę ogólnej AI

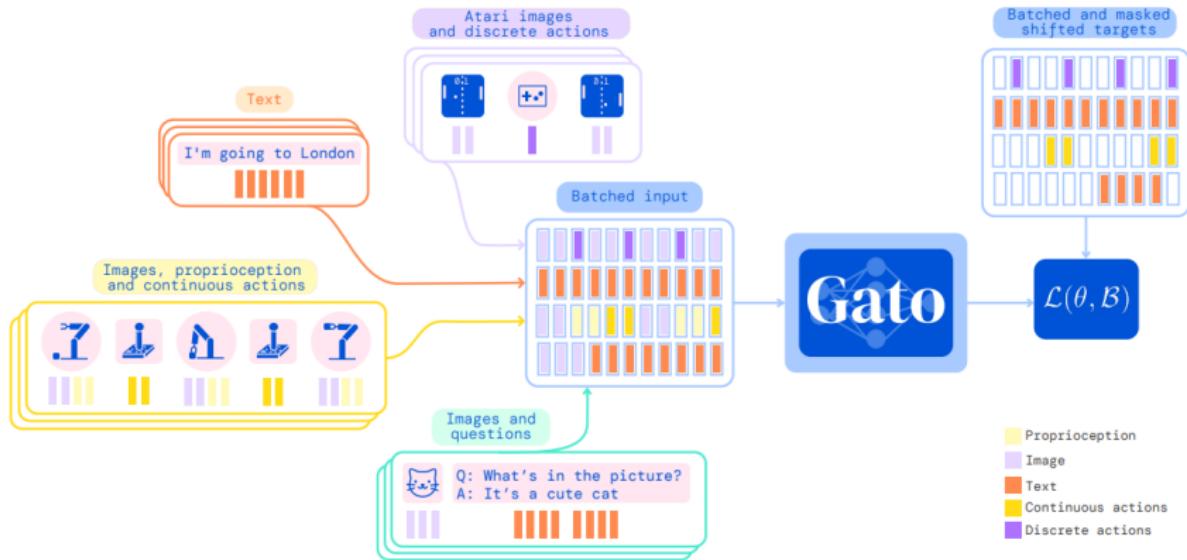


Figure 2: **Training phase of Gato.** Data from different tasks and modalities is serialized into a flat sequence of tokens, batched, and processed by a transformer neural network akin to a large language model. Masking is used such that the loss function is applied only to target outputs, i.e. text and various actions.

GATO: w stronę ogólnej AI

Given a sequence of tokens $s_{1:L}$ and parameters θ , we model the data using the chain rule of probability:

$$\log p_\theta(s_1, \dots, s_L) = \sum_{l=1}^L \log p_\theta(s_l | s_1, \dots, s_{l-1}), \quad (1)$$

Let b index a training batch of sequences \mathcal{B} . We define a masking function m such that $m(b, l) = 1$ if the token at index l is either from text or from the logged action of an agent, and 0 otherwise. The training loss for a batch \mathcal{B} can then be written as

$$\mathcal{L}(\theta, \mathcal{B}) = - \sum_{b=1}^{|\mathcal{B}|} \sum_{l=1}^L m(b, l) \log p_\theta \left(s_l^{(b)} | s_1^{(b)}, \dots, s_{l-1}^{(b)} \right) \quad (2)$$

GATO: w stronę ogólnej AI

Given a sequence of tokens $s_{1:L}$ and parameters θ , we model the data using the chain rule of probability:

$$\log p_\theta(s_1, \dots, s_L) = \sum_{l=1}^L \log p_\theta(s_l | s_1, \dots, s_{l-1}), \quad (1)$$

Let b index a training batch of sequences \mathcal{B} . We define a masking function m such that $m(b, l) = 1$ if the token at index l is either from text or from the logged action of an agent, and 0 otherwise. The training loss for a batch \mathcal{B} can then be written as

$$\mathcal{L}(\theta, \mathcal{B}) = - \sum_{b=1}^{|\mathcal{B}|} \sum_{l=1}^L m(b, l) \log p_\theta \left(s_l^{(b)} | s_1^{(b)}, \dots, s_{l-1}^{(b)} \right) \quad (2)$$

- Czyli nie modelujemy opisu stanu, obrazków itd, tylko wynikowy tekst, względnie akcję
- Warto pamiętać o maskowaniu części przewidywań

GATO: w stronę ogólnej AI

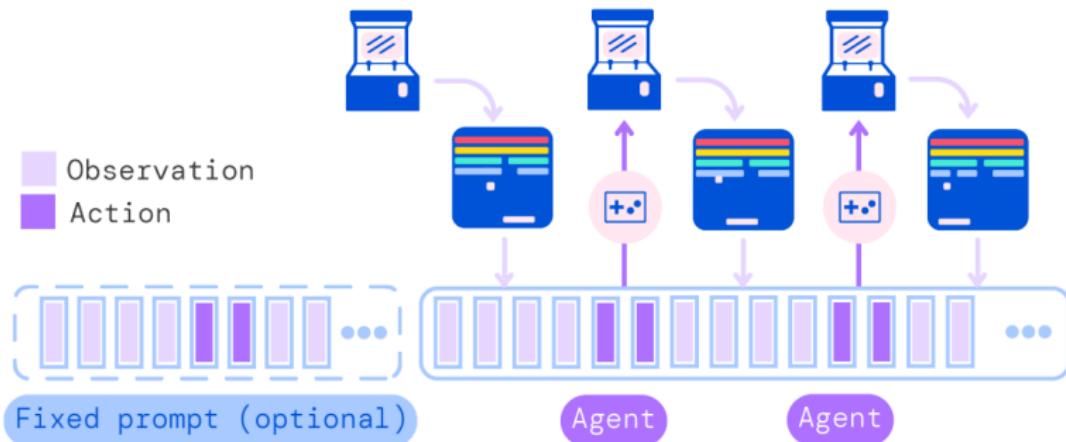


Figure 3: **Running Gato as a control policy.** Gato consumes a sequence of interleaved tokenized observations, separator tokens, and previously sampled actions to produce the next action in standard autoregressive manner. The new action is applied to the environment – a game console in this illustration, a new set of observations is obtained, and the process repeats.

GATO: w stronę ogólnej AI

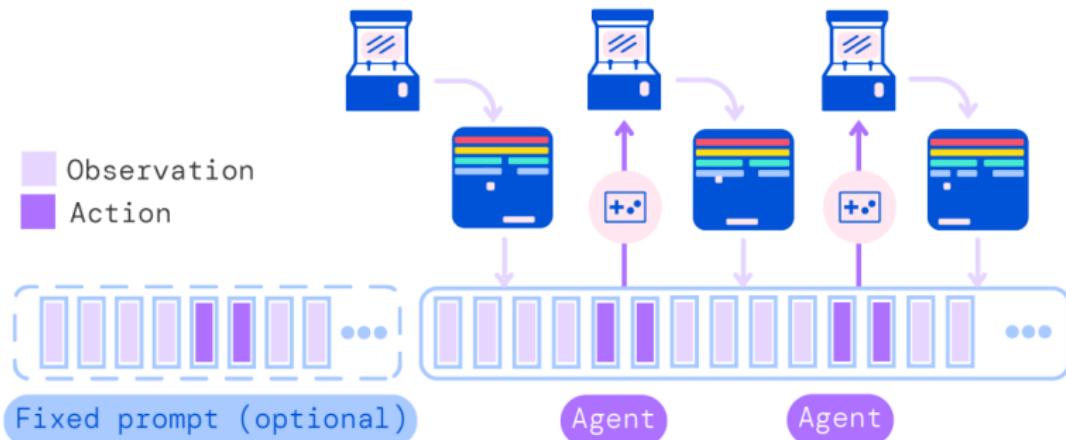


Figure 3: **Running Gato as a control policy.** Gato consumes a sequence of interleaved tokenized observations, separator tokens, and previously sampled actions to produce the next action in standard autoregressive manner. The new action is applied to the environment – a game console in this illustration, a new set of observations is obtained, and the process repeats.

Uwaga

Jak widać można włączyć twardy generator symboli (tu: program gry Arkanoid) w potok modelowania języka.

Ohtello vs Atari

- W Atari do ciągu symboli wrzuciliśmy **cały** opis stanu.
- W Othello był **pusty** opis stanu
- Pytanie: czy są możliwe rozwiązania pośrednie?

Ohtello vs Atari

- W Atari do ciągu symboli wrzuciliśmy **cały** opis stanu.
- W Othello był **pusty** opis stanu
- Pytanie: czy są możliwe rozwiązania pośrednie?

Odpowiedź

Są i co ciekawe, dają różne możliwości: przykładowo w Othello łatwiej jest grać modelowi, gdy otrzymuje informację o różnicy liczby białych i czarnych pionków po ruchu (co jeszcze ciekawsze, może ją sam obliczyć) – zachęcam do eksperymentów.

Transformery dla obrazów



Vision Transformer (uzupełnienie)

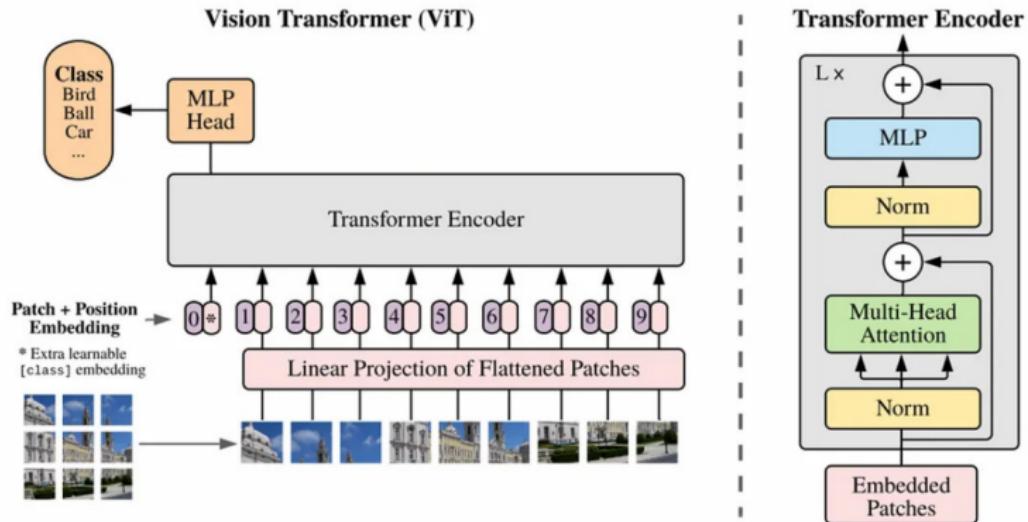


Image from Paper: "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale"

Vision Transformer (uzupełnienie)

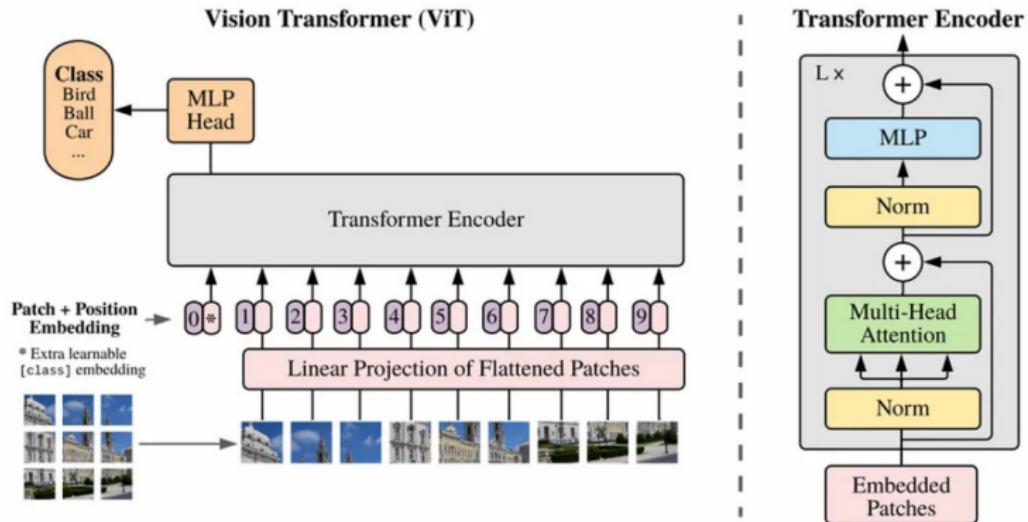


Image from Paper: "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale"

- Trenowany do zadania klasyfikacji całego obrazu (ImageNet)

Vision Transformer (uzupełnienie)

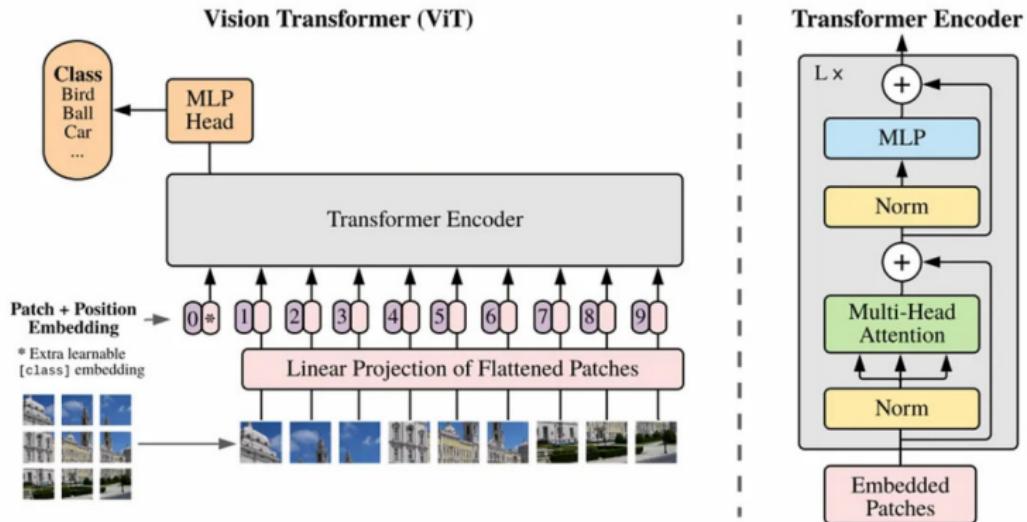


Image from Paper: "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale"

- Trenowany do zadania klasyfikacji całego obrazu (ImageNet)
- Osadzenia pozycji trenujemy

Vision Transformer (uzupełnienie)

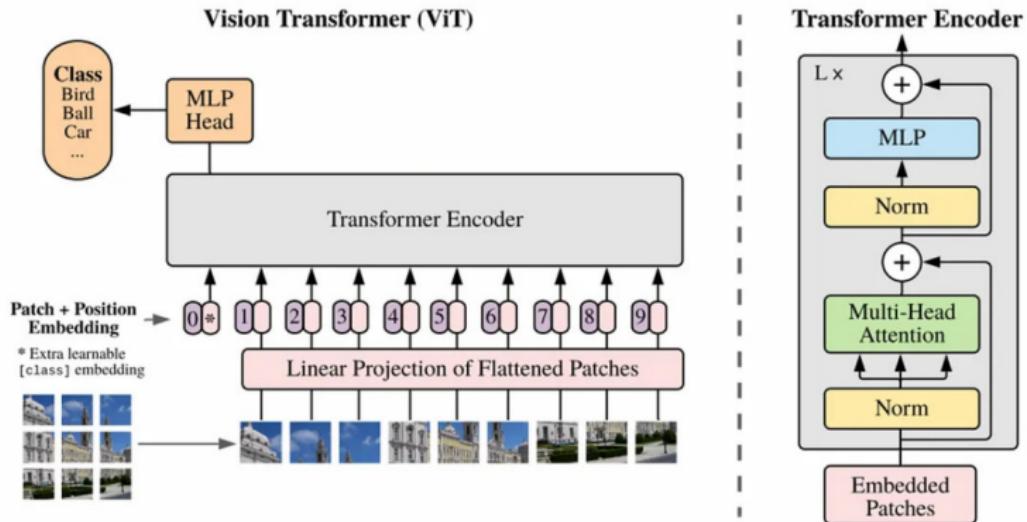


Image from Paper: "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale"

- Trenowany do zadania klasyfikacji całego obrazu (ImageNet)
- Osadzenia pozycji trenujemy
- Nie są potrzebne tokeny wejściowe (wrzucamy całe kawałki obrazka, przemnożone przez macierz projekcji)

Transformery dla obrazów (wersja dyskretna)

- Moglibyśmy klasteryzować obrazki (zwykły k-means)

Transformery dla obrazów (wersja dyskretna)

- Moglibyśmy klasteryzować obrazki (zwykły k-means)
- Wówczas można trenować MLM bez dodatkowych danych

Transformery dla obrazów (wersja dyskretna)

- Moglibyśmy klasteryzować obrazki (zwykły k-means)
- Wówczas można trenować MLM bez dodatkowych danych
 - ▶ Schematy maskowania mogą być różne, czasem utrudnia się zadanie wycinając spójne fragmenty, zarówno tekstu (SpanBERT), jak i obrazu

Transformery dla obrazów (wersja dyskretna)

- Moglibyśmy klasteryzować obrazki (zwykły k-means)
- Wówczas można trenować MLM bez dodatkowych danych
 - ▶ Schematy maskowania mogą być różne, czasem utrudnia się zadanie wycinając spójne fragmenty, zarówno tekstu (SpanBERT), jak i obrazu
- Trening właściwy takiego modelu mógłby korzystać z niewielkiej ilości danych

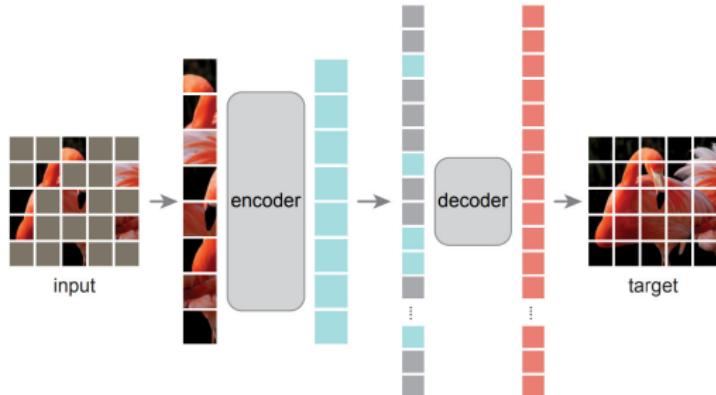
Transformery dla obrazów (wersja dyskretna)

- Moglibyśmy klasteryzować obrazki (zwykły k-means)
- Wówczas można trenować MLM bez dodatkowych danych
 - ▶ Schematy maskowania mogą być różne, czasem utrudnia się zadanie wycinając spójne fragmenty, zarówno tekstu (SpanBERT), jak i obrazu
- Trening właściwy takiego modelu mógłby korzystać z niewielkiej ilości danych

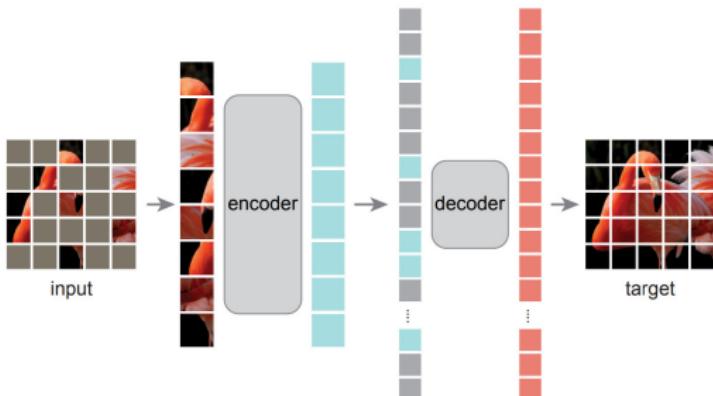
Ale można lepiej!

Na kolejnych slajdach będzie inne, prawdopodobnie lepsze podejście

Masked auto-encoder (MAE)

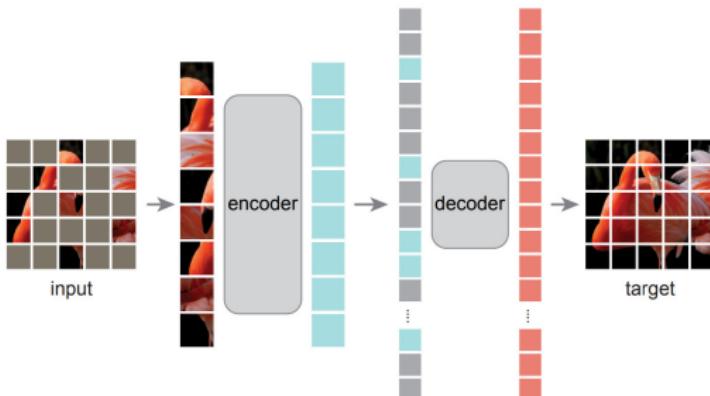


Masked auto-encoder (MAE)



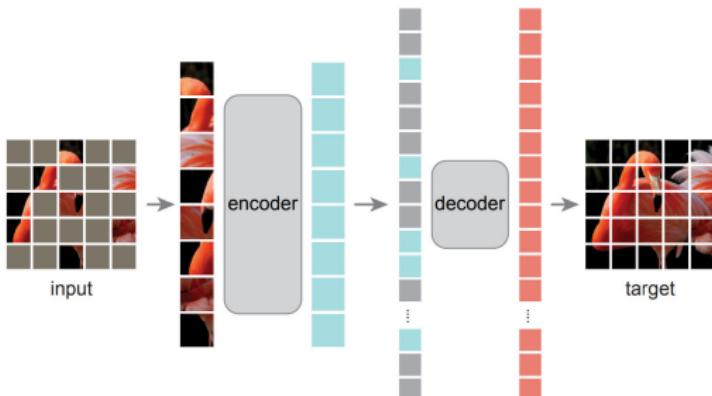
- Zamazujemy dużo kwadratów, do kodera przekazujemy tylko niezamazane (wraz z pozycją)

Masked auto-encoder (MAE)



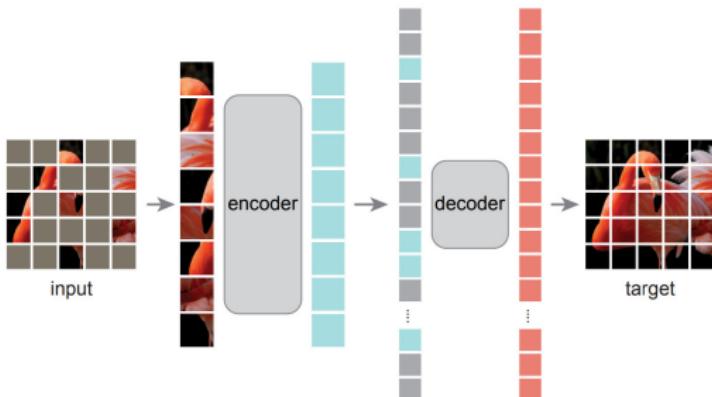
- Zamazujemy dużo kwadratów, do kodera przekazujemy tylko niezamazane (wraz z pozycją)
- Musi odtworzyć te niezamazane (funkcja kosztu: błąd średniokwadratowy na pikselach)

Masked auto-encoder (MAE)



- Zamazujemy dużo kwadratów, do kodera przekazujemy tylko niezamazane (wraz z pozycją)
- Musi odtworzyć te niezamazane (funkcja kosztu: błąd średniokwadratowy na pikselach)
- Funkcja kosztu nie nagradza ostrych krawędzi (ważniejszy jest odcień nieba, niż kontur samolotu – bo ten kontur to bardzo mało pikseli)

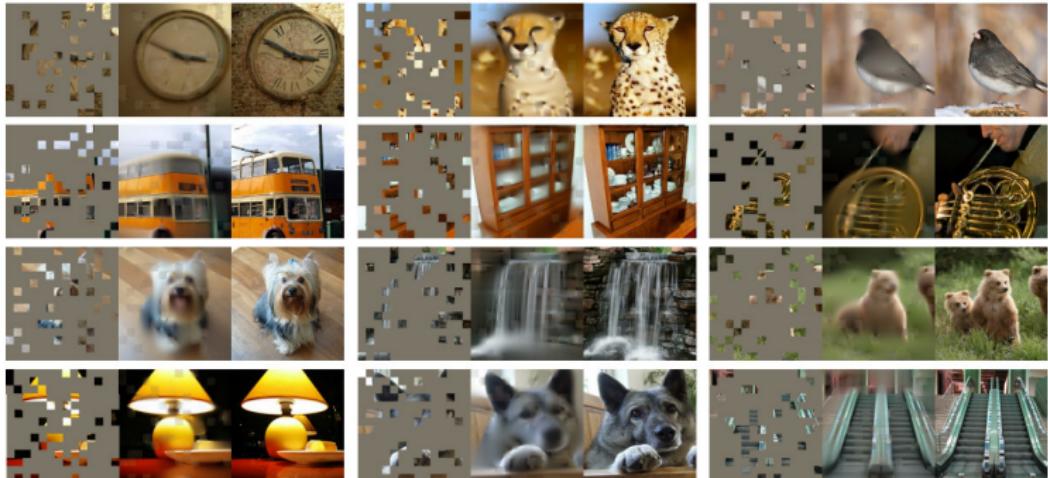
Masked auto-encoder (MAE)



- Zamazujemy dużo kwadratów, do kodera przekazujemy tylko niezamazane (wraz z pozycją)
- Musi odtworzyć te niezamazane (funkcja kosztu: błąd średniokwadratowy na pikselach)
- Funkcja kosztu nie nagradza ostrych krawędzi (ważniejszy jest odcień nieba, niż kontur samolotu – bo ten kontur to bardzo mało pikseli)

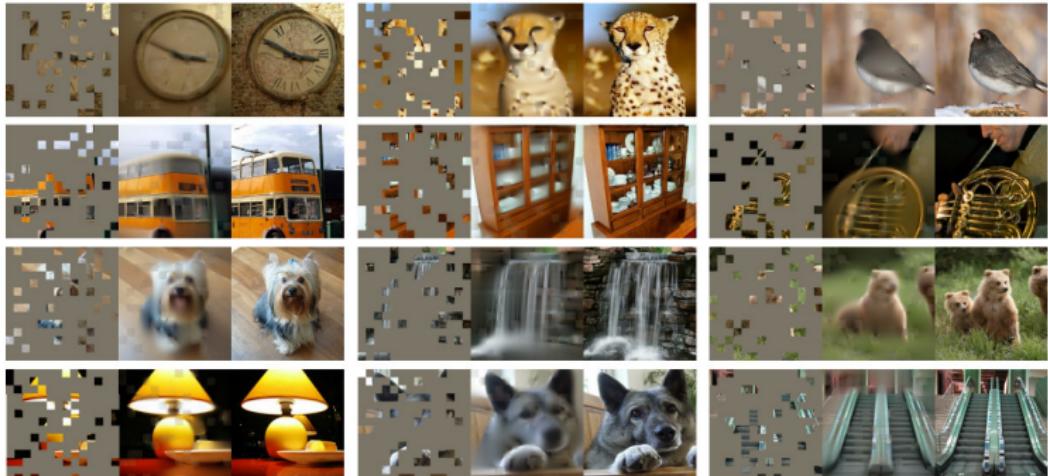
Nie jest naszym celem generowanie *ładnych* obrazków, ale wytrenowanie uniwersalnego kodera (który potem będzie dostrajany)

Masked auto-encoder (MAE)



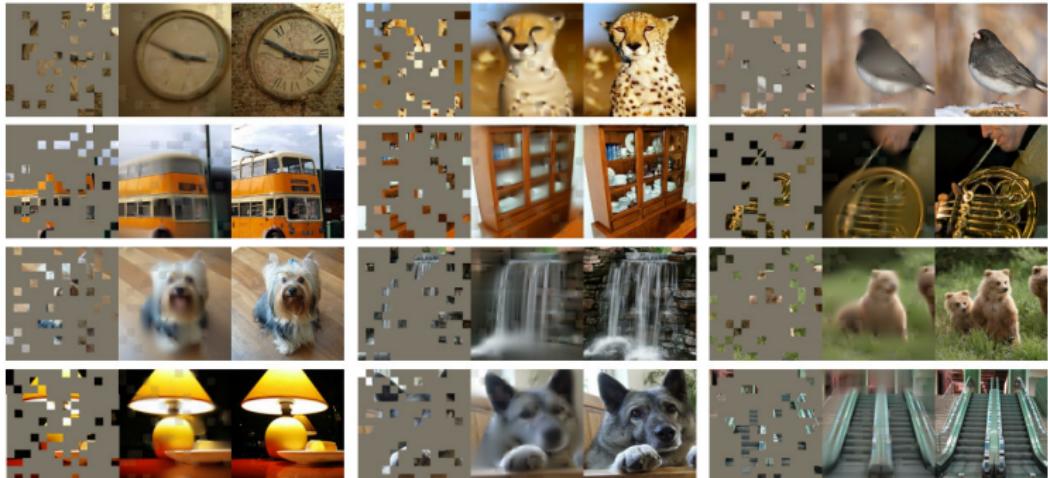
- Lewy obrazek to to, co widzi koder, środkowy – wynik dekodowania, prawy – oryginał

Masked auto-encoder (MAE)



- Lewy obrazek to to, co widzi koder, środkowy – wynik dekodowania, prawy – oryginał
- Wszystkie obrazki były **spoza** zbioru treningowego!

Masked auto-encoder (MAE)



- Lewy obrazek to to, co widzi koder, środkowy – wynik dekodowania, prawy – oryginał
- Wszystkie obrazki były **spoza** zbioru treningowego!
- Widzimy, że semantyczne wygląda to bardzo dobrze: nie mamy kłopotu ze zgadnięciem, co jest na obrazku (widząc tylko kwadraty mielibyśmy kłopot)

NLP w HuggingFace



- To już ostatnie z ważnych zadań z NLP, które omówimy.

NLP w HuggingFace



- To już ostatnie z ważnych zadań z NLP, które omówimy.
- (małe przypomnienie: BertScore jest fajny, ale nie nadaje się do porównywania tekstu z milionem innych tekstów)

NLP w HuggingFace



- To już ostatnie z ważnych zadań z NLP, które omówimy.
- (małe przypomnienie: BertScore jest fajny, ale nie nadaje się do porównywania tekstu z milionem innych tekstów)
- Reszta zadań: pewnie sobie o nich trochę powiemy na ostatnich ćwiczeniach.

Sentence transformers

Zadanie

Chcemy mieć osadzenie zdania w przestrzeni R^n , które jak najlepiej oddaje jego semantykę.

Sentence transformers

Zadanie

Chcemy mieć osadzenie zdania w przestrzeni R^n , które jak najlepiej oddaje jego semantykę.

Problemy

- Wektor dla **<CLS>** nie jest idealny – w szczególności zawiera głównie te informacje, które były istotne w treningu wstępny.

Sentence transformers

Zadanie

Chcemy mieć osadzenie zdania w przestrzeni R^n , które jak najlepiej oddaje jego semantykę.

Problemy

- Wektor dla **<CLS>** nie jest idealny – w szczególności zawiera głównie te informacje, które były istotne w treningu wstępny.
- Uśrednienie wszystkich wektorów nie jest złe, ale średnia trochę rozmywa wektory, a ponadto problem z poprzedniego punktu nie znika

Bliźniacze BERT-y



Sieci syjamskie

Trening wstępny

Trenujemy standardowego (pojedynczego) BERT-a

Trening wstępny

Trenujemy standardowego (pojedynczego) BERT-a

Dostrajanie. Wariant klasyfikacji

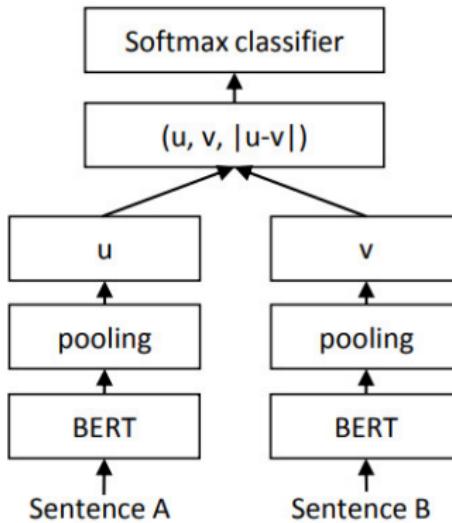
- Dane (opcja 1):
 - ▶ Przykłady pozytywne: pary zdań, będące swoimi parafrazami.
 - ▶ Przykłady negatywne: pary zdań, **niebędące** swoimi parafrazami (mimo podobnego słownictwa).
- Dane (opcja 2):
 - ▶ The Stanford Natural Language Inference (SNLI) Corpus

Korpus SNLI

Text	Judgments	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

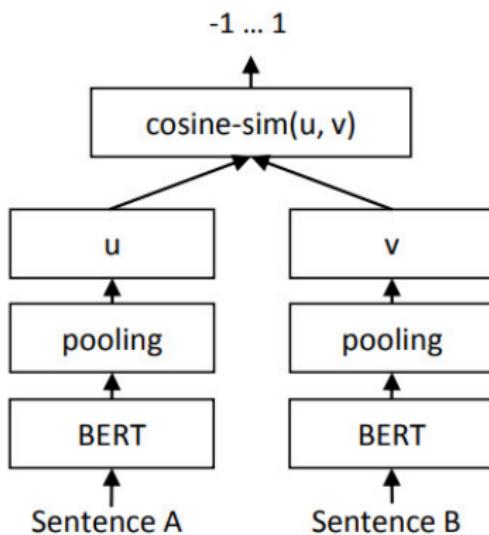
- Zwróćmy uwagę na niezgodność osób oznaczających relacje (co jest dość typowe w trudniejszych zadaniach NLP)

Sieci syjamskie. Uczenie klasyfikatora



- $|v - u|$ oznacza wektor wartości bezwględnych różnic współrzędnych
- Na trzech wektorach jest trenowany prosty klasyfikator (softmax) ze standardową funkcją kosztu

Sieci syjamskie. Uczenie regresora i inferencja



- Gdy używamy takich sieci, po prostu porównujemy (cos) wyliczone reprezentacje wektorowe
- Możemy też trenować miarę podobieństwa, jeżeli mamy dostęp do takich danych uczących

Sieci syjamskie. Testy ABX i triplet loss

Triplet Objective Function. Given an anchor sentence a , a positive sentence p , and a negative sentence n , triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n . Mathematically, we minimize the following loss function:

$$\max(\|s_a - s_p\| - \|s_a - s_n\| + \epsilon, 0)$$

with s_x the sentence embedding for $a/n/p$, $\|\cdot\|$ a distance metric and margin ϵ . Margin ϵ ensures that s_p is at least ϵ closer to s_a than s_n . As metric we use Euclidean distance and we set $\epsilon = 1$ in our experiments.

Osadzenia pozycji niepodlegające uczeniu

Uwaga

Używa się też kodowań pozycji, które nie są trenowane (tak było w oryginalnym transformerze)

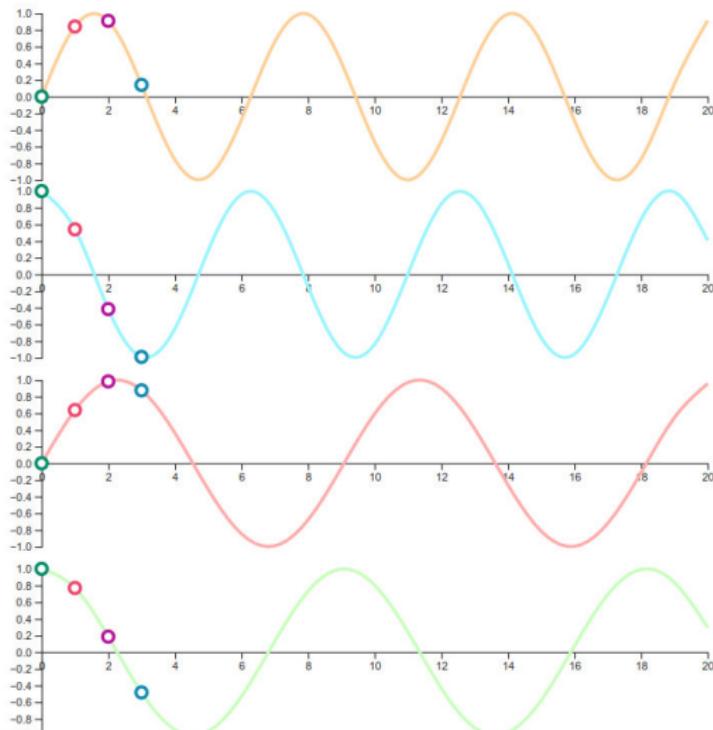
Osadzenia pozycji niepodlegające uczeniu

Uwaga

Używa się też kodowań pozycji, które nie są trenowane (tak było w oryginalnym transformerze)

- Bazują one na funkcjach sin i cos z różnymi okresami
- Pewne analogie dla kodowania binarnego: jak kolumny są kodami kolejnych liczb

Positional encoding visualization



	p0	p1	p2	p3	
i=0	0.000	0.841	0.909	0.141	
i=1	1.000	0.540	-0.416	-0.990	
i=2	0.000	0.638	0.983	0.875	
i=3	1.000	0.770	0.186	-0.484	

Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Settings: $d = 50$

The value of each positional encoding depends on the *position* (pos) and *dimension* (d). We calculate result for every *index* (i) to get the whole vector.



RoPE – zapowiedź

- Inną, popularną obecnie opcją są Rottary Positional Embeddings
- Nie dodajemy tam zanurzeń pozycji, ale mnożymy przez pewną (stałą i prawie przekatniową) macierz

nanoGPT:krótka instrukcja użycia

- ① Pobieramy kod z <https://github.com/karpathy/nanoGPT>
- ② Próbujemy wytrenować model Szekspirowski, zgodnie z instrukcjami na stronie
- ③ Sprawdzamy generowanie (skrypt sample.py)

nanoGPT: krótka instrukcja użycia

- ① Pobieramy kod z <https://github.com/karpathy/nanoGPT>
- ② Próbujemy wytrenować model Szekspirowski, zgodnie z instrukcjami na stronie
- ③ Sprawdzamy generowanie (skrypt sample.py)

Co może pojść nie tak?

- Brak wsparcia dla **bfloat16**?
Rozwiązanie: zamieniamy na **float16**

nanoGPT: Tokenizacja

Propozycja

Przygotowujemy korpus w ten sposób, by `tokenize == split`

Popatrzmy na kod

```
stoi, itos = meta['stoi'], meta['itos']
encode = lambda s: [stoi[c] for c in s.split()]
decode = lambda l: ' '.join([itos[i] for i in l])
```

nanoGPT: wielkość modelu i inne parametry

Konwencja: tworzymy nowy plik konfiguracyjny dla zadania.

Popatrzmy na kod config/config_X.py

nanoGPT: dataloader

Popatrzmy na kod: `train.py`, funkcja `get_batch.py`

nanoGPT: dataloader

Popatrzmy na kod: `train.py`, funkcja `get_batch.py`

- Wybieramy dane z dowolnego miejsca w korpusie
- Pytanie: Jakie to ma konsekwencje?

nanoGPT: dataloader

Popatrzmy na kod: `train.py`, funkcja `get_batch.py`

- Wybieramy dane z dowolnego miejsca w korpusie
- Pytanie: Jakie to ma konsekwencje?
- Odpowiedź: w jednym ciągu mogą być rzeczy z różnych 'artykułów' (piosenek, partii gier, ...) i system musi się tego nauczyć

Connect Four



- Fajny wariant kółka i krzyżyka
- W większości sytuacji 7 możliwych ruchów
- Łatwo stworzyć agenta, który gra sensownie, a jednocześnie niedeterministycznie.
- Taki agent może nam wygenerować partie, jest też naturalnym przeciwnikiem.

Agent-0

- Minimax na poziomie 5, bez heurystyki (tylko zwycięstwa i przegrane, reszta stanów ma wartość 0)
- Remisy rozwiązuje prostymi symulacjami Monte Carlo
- Niepowtarzające się partie, sensowna jakość

Agent-0

- Minimax na poziomie 5, bez heurystyki (tylko zwycięstwa i przegrane, reszta stanów ma wartość 0)
- Remisy rozwiązuje prostymi symulacjami Monte Carlo
- Niepowtarzające się partie, sensowna jakość

Agent0 wygenerował 20K rozgrywek (nawiasem mówiąc użytych na przedmiocie sieci neuronowe)

Agent-0

- Minimax na poziomie 5, bez heurystyki (tylko zwycięstwa i przegrane, reszta stanów ma wartość 0)
- Remisy rozwiązuje prostymi symulacjami Monte Carlo
- Niepowtarzające się partie, sensowna jakość

Agent0 wygenerował 20K rozgrywek (nawiasem mówiąc użytych na przedmiocie sieci neuronowe)

Jakość rozgrywek

- Agent-0 vs Random: **(500-0-0)**

Korpus C4

Tokeny == znaki

S11326354624002233001B

S003001141011222434423321602345355546666655D

S1015114330400505542233641446331665560622B

S3430400600143364144122131130666622B

S323352465224555134411136200A

S514323233215121142245354435500104000B

Korpus C4

Tokeny == znaki

S11326354624002233001**B**

S003001141011222434423321602345355546666655**D**

S1015114330400505542233641446331665560622**B**

S3430400600143364144122131130666622**B**

S323352465224555134411136200**A**

S514323233215121142245354435500104000**B**

Zadanie dla modelu językowego

LM traktuje to jako ciąg symboli (11-elementowy alfabet). Podczas treningu musi

- nauczyć się reguł,
- nauczyć się warunków końcowych,
- nauczyć się rozsądnie grać.

Korpus C4

Tokeny == znaki

S11326354624002233001B

S003001141011222434423321602345355546666655D

S1015114330400505542233641446331665560622B

S3430400600143364144122131130666622B

S323352465224555134411136200A

S514323233215121142245354435500104000B

Zadanie dla modelu językowego

LM traktuje to jako ciąg symboli (11-elementowy alfabet). Podczas treningu musi

- nauczyć się reguł,
- nauczyć się warunków końcowych,
- nauczyć się rozsądnie grać.

Ustawienia nano-GPT z Szekspira



Wyniki dla Agenta-LM1

- Dokładność przewidywania stanów końcowych: około 50% (czyli skuteczność odpowiedzi na pytanie czy gra się skończyła i kto wygrał)
- Wyniki pojedynku Agenta0 and Agenta-LM1: **(887-90-23)** (około 10% zwycięstw)

Wyniki dla Agenta-LM1

- Dokładność przewidywania stanów końcowych: około 50% (czyli skuteczność odpowiedzi na pytanie czy gra się skończyła i kto wygrał)
- Wyniki pojedynku Agenta0 and Agenta-LM1: **(887-90-23)** (około 10% zwycięstw)

Komentarz

- Pokonanie swojego mistrza można uważać za sukces (nawet jeżeli tylko w 10% gier)
- Przewidywanie ruchów lepsze niż księga otwarć (46% vs 30%)
- Topline dla przewidywania ruchów jest 57.6% (bo rozgrywki mają sporo elementu losowości)

Korpus C4-2C

Poprzednia wersja

S11326354624002233001B

S323352465224555134411136200A

W C4-2C używamy innych symboli dla obu graczy

S11326354624002233001B

S323352465224555134411136200A

Wyniki Agenta-LM2

- Dokładność przewidywania stanów końcowych: około 75%
- Pojedynek Agent0 and Agent-LM2: **(844-134-22)**
- Losowy gracz kontra Agent-LM2: **(8-992-0)**

Wyniki Agenta-LM2

- Dokładność przewidywania stanów końcowych: około 75%
- Pojedynek Agent0 and Agent-LM2: **(844-134-22)**
- Losowy gracz kontra Agent-LM2: **(8-992-0)**

Komentarz

- Więcej informacji w tokenie == lepsza gra

Wyniki Agenta-LM2

- Dokładność przewidywania stanów końcowych: około 75%
- Pojedynek Agent0 and Agent-LM2: **(844-134-22)**
- Losowy gracz kontra Agent-LM2: **(8-992-0)**

Komentarz

- Więcej informacji w tokenie == lepsza gra
- Czy możemy upakować jeszcze więcej informacji w tokenie?

Mała wycieczka do świata szachów

Praca magisterska p. Kamila Michalaka proponowała taką tokenizację ruchów (która dała przyzwoitą jakość rozgrywki)

Type	Examples	Count
King moves	Ka1a2, Kf8g7	420
Queen moves	Qd1h6, Qd8e7	1456
Rook moves	Ra1b1, Re8e1	896
Bishop moves	Bf1g5, Bg7b2	560
Knight moves	Ng1f3, Nb7c6	336
Pawn moves	Pe2e4, Pb4c3	236
Castles	Ke1g1, Ke8c8	4
Pawn promotions	Pe7e8q, Pa2b1n	176
Special symbols	[EOS], [BOS], [PAD]	3
Total:		4087

Korpus C4-XY

Korpus C4-XY

- Ruch zawiera informację: player, x, y
- Rozmiar słownika: $42 \times 2 + 4$

- Ruch zawiera informację: player, x, y
- Rozmiar słownika: $42 \times 2 + 4$

Dodatkowa informacja

Testowane modele mają około 10.5M parametrów (drobne różnice wynikają z wielkości tablicy osadzeń)

Agent-LM3 results

- Wyniki meczów pomiędzy Agentem0 and Agentem-LM3:
(583-349-68)
- Wyniki meczów pomiędzy agentem losowym a Agentem-LM3:
(3-997-0)

Agent-LM3 results

- Wyniki meczów pomiędzy Agentem0 and Agentem-LM3:
(583-349-68)
- Wyniki meczów pomiędzy agentem losowym a Agentem-LM3:
(3-997-0)

Uwaga

Jak starczy czasu, to opowiemy sobie o śmiesznej idei **notatek** w kontekście gry C4.

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research [†]Universitat Pompeu Fabra

Abstract

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that LMs can teach themselves to *use external tools* via simple APIs and achieve the best of both worlds. We introduce *Toolformer*, a model

Toolformer: zestaw narzędzi

- ① Kalkulator – oblicza wyrażenia

Toolformer: zestaw narzędzi

- ① Kalkulator – oblicza wyrażenia
- ② System odpowiadania na pytania w języku naturalnym (wykorzystujący bazy wiedzy)

Toolformer: zestaw narzędzi

- ① Kalkulator – oblicza wyrażenia
- ② System odpowiadania na pytania w języku naturalnym (wykorzystujący bazy wiedzy)
- ③ Prosta wyszukiwarka do Wikipedii

Toolformer: zestaw narzędzi

- ① Kalkulator – oblicza wyrażenia
- ② System odpowiadania na pytania w języku naturalnym (wykorzystujący bazy wiedzy)
- ③ Prosta wyszukiwarka do Wikipedii
- ④ Kalendarz (chyba nie całkiem potrzebny)

Toolformer: zestaw narzędzi

- ① Kalkulator – oblicza wyrażenia
- ② System odpowiadania na pytania w języku naturalnym (wykorzystujący bazy wiedzy)
- ③ Prosta wyszukiwarka do Wikipedii
- ④ Kalendarz (chyba nie całkiem potrzebny)
- ⑤ Słownik wielojęzyczny

Toolformer: zestaw narzędzi

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

- Te kolorowe wstawki zostają umieszczone automatycznie w korpusie.
- Składają się z wywołania API, strzałki i wyniku zwróconego prez API
- Wszystko w nawiasach kwadratowych, żeby łatwo usunąć przed prezentowaniem użytkownikowi

Toolformer: zestaw narzędzi

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

- Te kolorowe wstawki zostają umieszczone automatycznie w korpusie.
- Składają się z wywołania API, strzałki i wyniku zwróconego prez API
- Wszystko w nawiasach kwadratowych, żeby łatwo usunąć przed prezentowaniem użytkownikowi

Uwaga

W ostatecznym treningu analizując tokeny po **that** model widzi cały ten tekst (tekst oryginalny i treść w nawiasie)

Toolformer: zestaw narzędzi

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Toolformer: zestaw narzędzi

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

We wszystkich przypadkach odpowiedź pomaga generować kolejne symbole

Toolformer: jak generować wywołania

Jakieś pomysły?

Toolformer: jak generować wywołania

Jakieś pomysły?

- Tradycyjny schemat few-shot learning
- Dla każdego API trzeba przygotowujemy szablon

Toolformer: jak generować wywołania

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: x

Output:

Toolformer: wzbogacanie korpusu

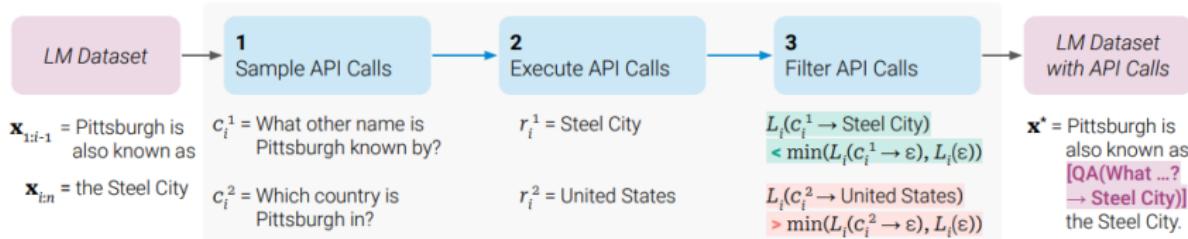


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text \mathbf{x} , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text \mathbf{x}^* .

- Zakładamy, że między *known as* a *the Steel* będzie API-call (skąd to wiemy – na początku losujemy, może z jakimiś heurystykami)
- Możemy też wziąć zdanie i wygenerować wywołania (jak na poprzednim slajdzie)

Toolformer: wzbogacanie korpusu

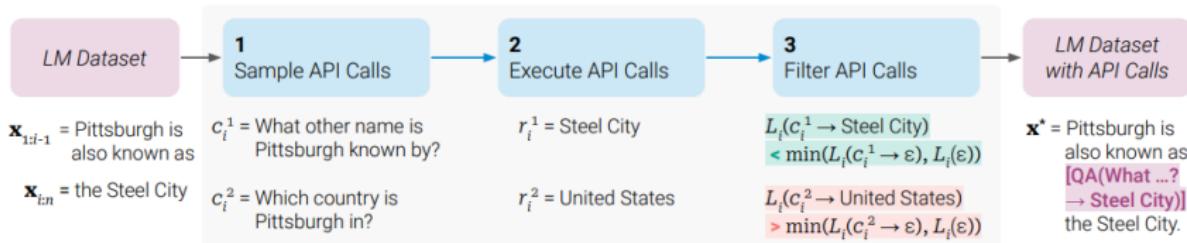


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- W danym miejscu generujemy, za pomocą (innego?) LLM-a wywołania do API (punkt 1)

Toolformer: wzbogacanie korpusu

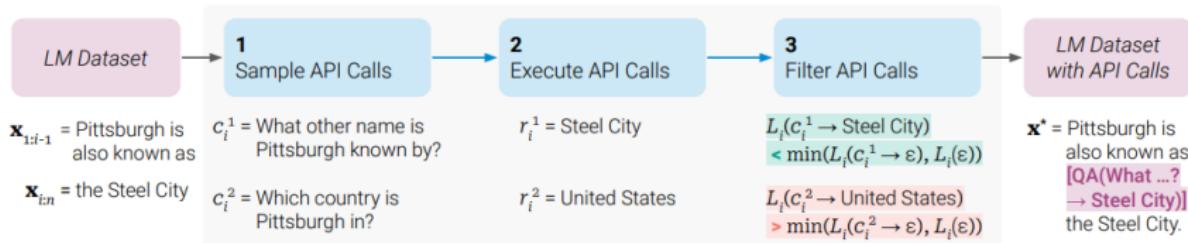


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- W danym miejscu generujemy, za pomocą (innego?) LLM-a wywołania do API (punkt 1)
- ... które następnie wykonujemy (punkt 2)

Toolformer: wzbogacanie korpusu

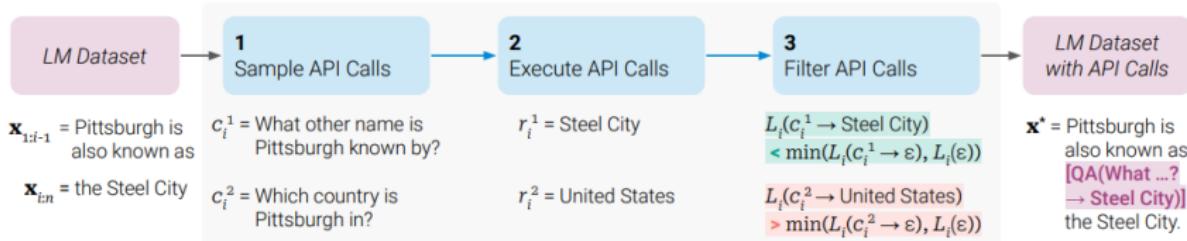


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- W danym miejscu generujemy, za pomocą (innego?) LLM-a wywołania do API (punkt 1)
- ... które następnie wykonujemy (punkt 2)

Pozostała decyzja, który (i czy jakiś) wybrać.

Toolformer: wzbogacanie korpusu

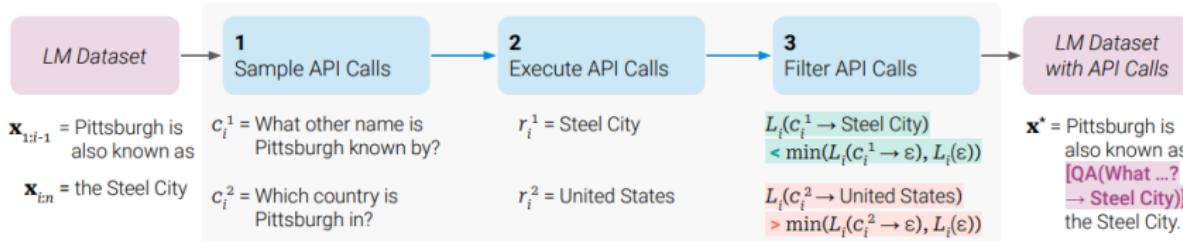


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- Wklejamy (punkt 3) **przed** naszym zdaniem/akapitem:
 - ▶ albo **nic**
 - ▶ albo **samo pytanie**
 - ▶ albo **pytanie z odpowiedzią**

Toolformer: wzbogacanie korpusu

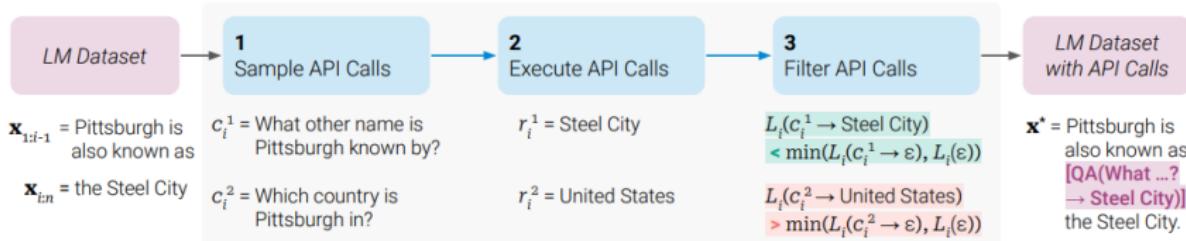


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- Wklejamy (punkt 3) **przed** naszym zdaniem/akapitem:
 - ▶ albo **nic**
 - ▶ albo **samo pytanie**
 - ▶ albo **pytanie z odpowiedzią**
- Liczymy koszt wygenerowania kontynuacji w każdym wariantów

Toolformer: wzbogacanie korpusu

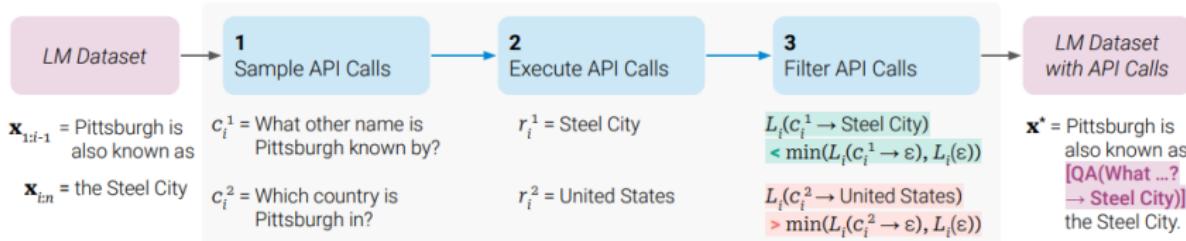


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

- Wklejamy (punkt 3) **przed** naszym zdaniem/akapitem:
 - ▶ albo **nic**
 - ▶ albo **samo pytanie**
 - ▶ albo **pytanie z odpowiedzią**
- Liczymy koszt wygenerowania kontynuacji w każdym wariantów
- Jeżeli trzeci jest najniższy, to znaczy, że to dobre miejsce do wywołania API

Toolformer: inferencja



- Mamy model, który generuje teksty z „kolorowymi” zapytaniami.
- Jak wygeneruje znacznik API to generujemy dalej aż do wygenerowania strzałki i następnie wywołujemy API z odpowiednim argumentem.

Toolformer: inferencja



- Mamy model, który generuje teksty z „kolorowymi” zapytaniami.
- Jak wygeneruje znacznik API to generujemy dalej aż do wygenerowania strzałki i następnie wywołujemy API z odpowiednim argumentem.
- ... a wynik wklejamy do tekstu (nie musimy go generować, choć model by chciał :)

Toolformer: inferencja



- Mamy model, który generuje teksty z „kolorowymi” zapytaniami.
- Jak wygeneruje znacznik API to generujemy dalej aż do wygenerowania strzałki i następnie wywołujemy API z odpowiednim argumentem.
- ... a wynik wklejamy do tekstu (nie musimy go generować, choć model by chciał :)

Zwróćmy uwagę, że teraz generują się **mądrzejsze** wywołania API niż podczas treningu, bo

Toolformer: inferencja



- Mamy model, który generuje teksty z „kolorowymi” zapytaniami.
- Jak wygeneruje znacznik API to generujemy dalej aż do wygenerowania strzałki i następnie wywołujemy API z odpowiednim argumentem.
- ... a wynik wklejamy do tekstu (nie musimy go generować, choć model by chciał :)

Zwróćmy uwagę, że teraz generują się **mądrzejsze** wywołania API niż podczas treningu, bo w korpusie zostały tylko te użyteczne.

Transformery wnioskujące

Rozpoczynamy nowy wątek, związany z możliwościami wnioskowania przeprowadzonymi przez transformery

Transformery wnioskujące

Rozpoczynamy nowy wątek, związany z możliwościami wnioskowania przeprowadzonymi przez transformery

Quasi-rozumowanie na poziomie GPT-2

Suppose I have 4 apples, and I gave two of them to my sister,

Transformery wnioskujące

Rozpoczynamy nowy wątek, związany z możliwościami wnioskowania przeprowadzonymi przez transformery

Quasi-rozumowanie na poziomie GPT-2

Suppose I have 4 apples, and I gave two of them to my sister, and she gave one of them to her sister, and it started to grow. Would I be able to see how much time it had taken for the apple to grow?

We will first give you a number which you will later use to work out how much time it has taken for the apple to grow:

(Time it took to grow 2 apples, shown as a clock) \times (Size of apple) = "time taken"

What this tells us is that:

$$2 \times 3 \times 2 = 44 \text{ (seconds in 24 hours)}$$

So the question is, what does 44 seconds look like?

Let's do a quick experimen

Transformery wnioskujące

Oczywiście GPT-3 i dalsze modele są sporo lepsze, ale daleko do doskonałości!

Transformery wnioskujące

Oczywiście GPT-3 i dalsze modele są sporo lepsze, ale daleko do doskonałości!

Uwaga

Jeden źle wygenerowany token może rozwalić całe rozumowanie!

Transformery wnioskujące

Oczywiście GPT-3 i dalsze modele są sporo lepsze, ale daleko do doskonałości!

Uwaga

Jeden źle wygenerowany token może rozwalić całe rozumowanie!

Przyjrzymy się na kolejnych slajdach bardzo „płodnej” idei **łańcucha myśli**, która ułatwia/umożliwia rozumowanie w modelach autoregresywnych (takich jak GPT).

Chain of Thought (CoT)

(a) Few-shot

Examples

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

Chain of Thought (CoT)

Examples

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

CoT Examples

(b) Few-shot-CoT (Wei et al., 2022)

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Step-by-step Answer

Chain of Thought (CoT)

Examples

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

b) Few-shot-CoT (Wei et al., 2022)

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

CoT Examples

Step-by-step Answer

Chain of Thought (CoT)

Examples

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(b) Few-shot-CoT (Wei et al., 2022)

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

CoT Examples

Step-by-step Answer

(d) Zero-shot-CoT (Kojima et al., 2022)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

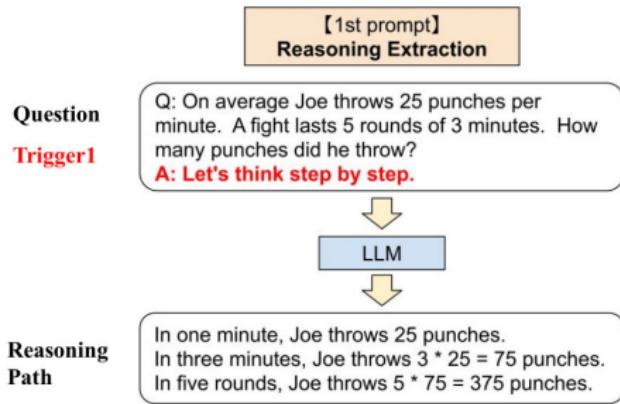
A: Let's think step by step.

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Two-stage Prompting Step-by-step Answer

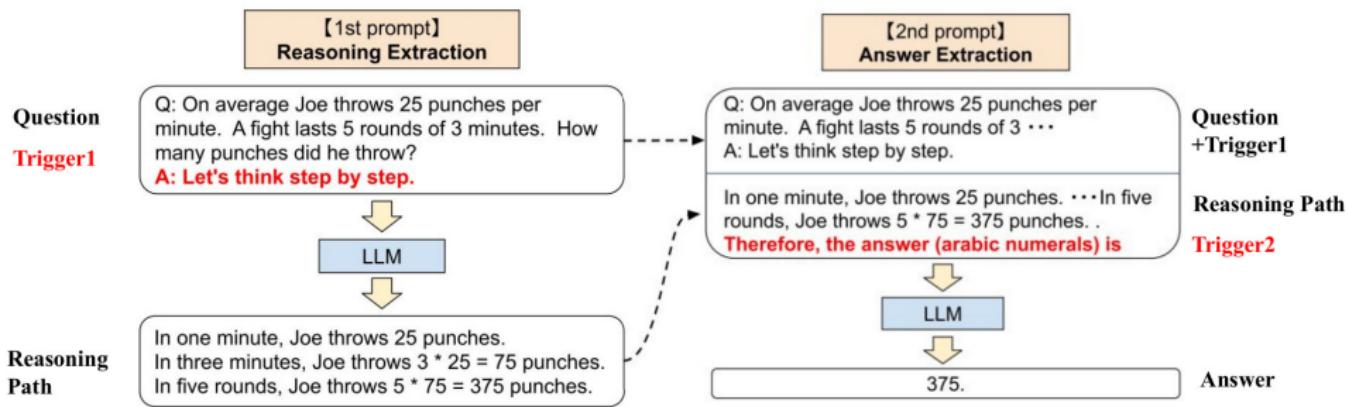
Zero-Shot Chain of Thought (CoT)

For **zero-shot CoT**, a **two-stage prompting** is applied:



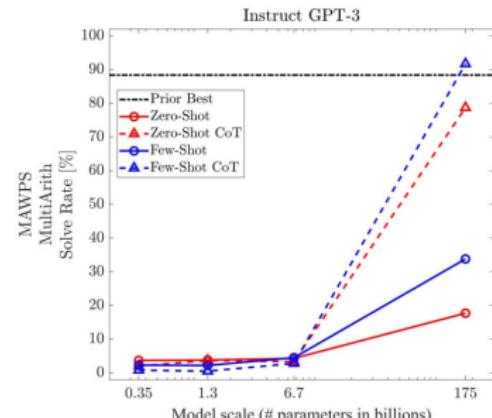
Zero-Shot Chain of Thought (CoT)

For **zero-shot CoT**, a **two-stage prompting** is applied:



Arithmetic Reasoning - Observations

- Both zero-shot and few-shot chain of thought promptings are emergent **abilities of model scale**.
- They do not positively impact performance for small models, but start to yield performance gains when used with models with more than ~100B parameters.
- Few-shot CoT achieves **better** performance on LLM than zero-shot CoT.
- Instruct GPT-3: text-davinci-002 achieves similar performance as PaLM 540B model



Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Equation only



5+6=11. The answer is 11.

Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

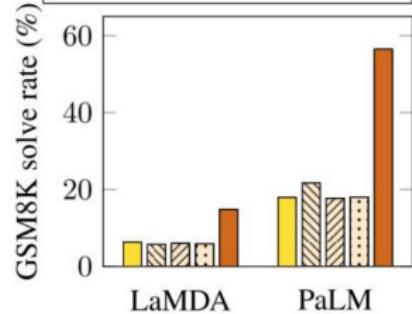
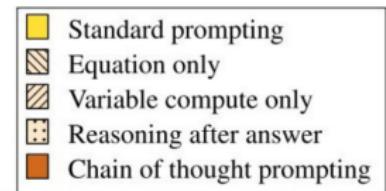
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Equation only



$5+6=11$. The answer is 11.



Natural language in reasoning matters.

(Wei et al., 2022)

71

Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Variable compute only



.....
..... *The
answer is 11.*

Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

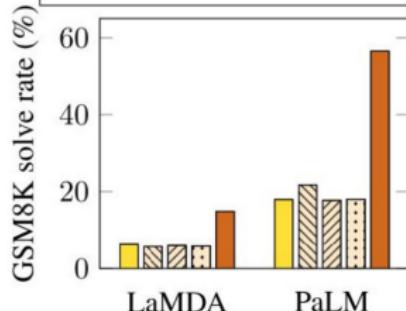
(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Variable compute only



.....
..... *The
answer is 11.*

- █ Standard prompting
- █ Equation only
- █ Variable compute only
- █ Reasoning after answer
- █ Chain of thought prompting



More intermediate computation does not help with the final answer.

(Wei et al., 2022)

Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Reasoning after answer



*The answer is 11.
Roger started with
5 balls. 2 cans of 3
tennis balls each is
6 tennis balls. $5 +$
 $6 = 11$.*

Ablation Study - Variations of Few-Shot CoT

Change the types of CoT:

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

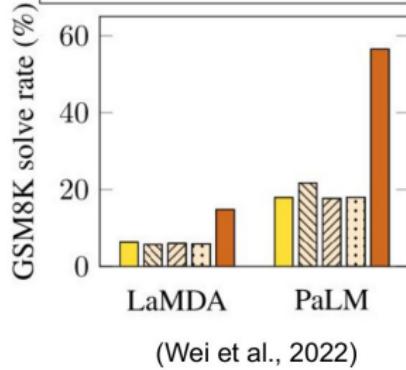
(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

Reasoning after answer



The answer is 11.
Roger started with
5 balls. 2 cans of 3
tennis balls each is
6 tennis balls. $5 +$
 $6 = 11$.

- █ Standard prompting
- █ Equation only
- █ Variable compute only
- █ Reasoning after answer
- █ Chain of thought prompting



CoT is not just activating knowledge seen in pre-training.

(Wei et al., 2022)

Zimowa zagadka muzyczna (obrazek 1)

Uwaga

Do prawidłowego rozwiązania zagadki należy obejrzeć oba obrazki

Zimowa zagadka muzyczna (obrazek 1)

Uwaga

Do prawidłowego rozwiązania zagadki należy obejrzeć oba obrazki



Noworoczna zagadka muzyczna (obrazek 2)

Noworoczna zagadka muzyczna (obrazek 2)



Rozwiążanie zagadki

- Pierwszy obrazek jest wyciętym fragmentem oficjalnego teledysku piosenki Go West, zespołu Pet Shop Boys

Rozwiążanie zagadki

- Pierwszy obrazek jest wyciętym fragmentem oficjalnego teledysku piosenki Go West, zespołu Pet Shop Boys
- której melodia jest często śpiewana/grana na stadionach z tekstem:
Polskaaa, biało czeerwonii

Rozwiążanie zagadki

- Pierwszy obrazek jest wyciętym fragmentem oficjalnego teledysku piosenki Go West, zespołu Pet Shop Boys
- której melodia jest często śpiewana/grana na stadionach z tekstem:
Polskaaa, biało czeerwonii

Potraktujemy pierwsze nuty tej piosenki jako prefiks i zobaczymy, jaką kontynuację wymyśli (niewielki) model typu GPT.

Go West

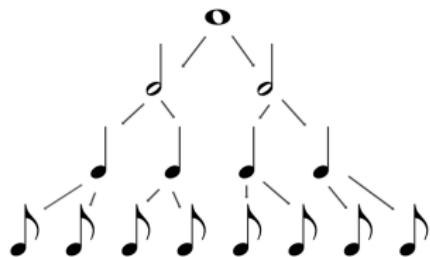
Pet Shop Boys



- Powyżej widzimy zapis (lejko zmodyfikowanej) linii melodycznej piosenki Go West, z którego utworzymy nasz prefiks
- (tak przy okazji, muzykę zapisuje się na tzw. pięciolinny, a te kóleczka z kreskami to nuty)

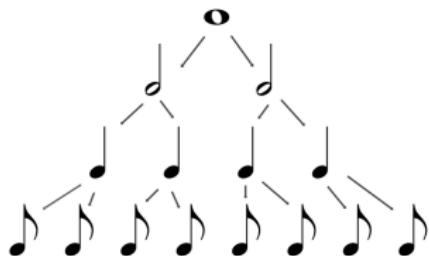
Muzyka: Lekcja 1

Nuty trwają jakiś czas (dość równy)



Muzyka: Lekcja 1

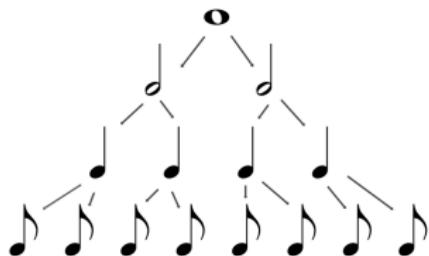
Nuty trwają jakiś czas (dość równy)



- ćwierćnuta to jedno podstawowe uderzenie perkusji

Muzyka: Lekcja 1

Nuty trwają jakiś czas (dość równy)



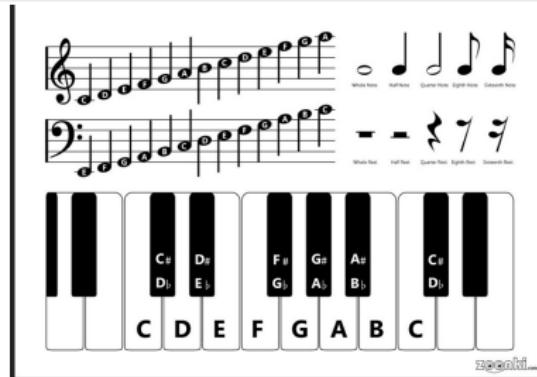
- ćwierćnota to jedno podstawowe uderzenie perkusji
- Kropka przy nucie wydłuża jej czas o 50%

Muzyka: Lekcja 2

Nuty mają **wysokość** (zależną od częstotliwości)

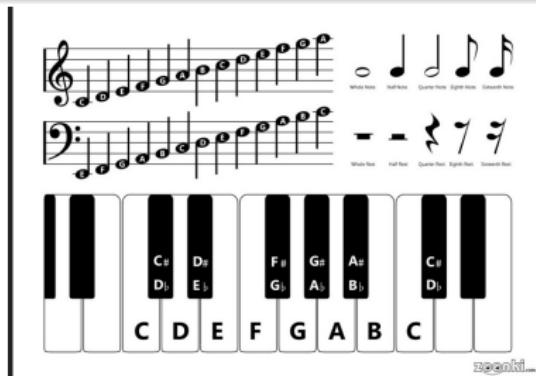
Muzyka: Lekcja 2

Nuty mają **wysokość** (zależną od częstotliwości)



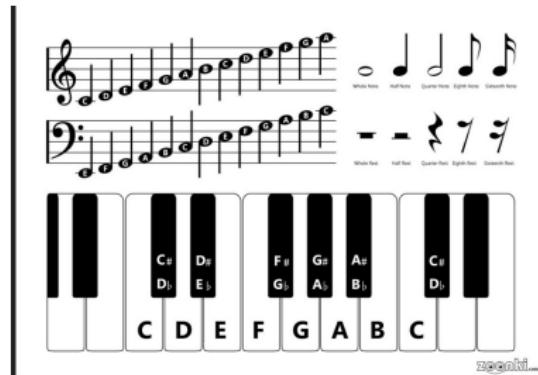
Muzyka: Lekcja 2

Nuty mają **wysokość** (zależną od częstotliwości)



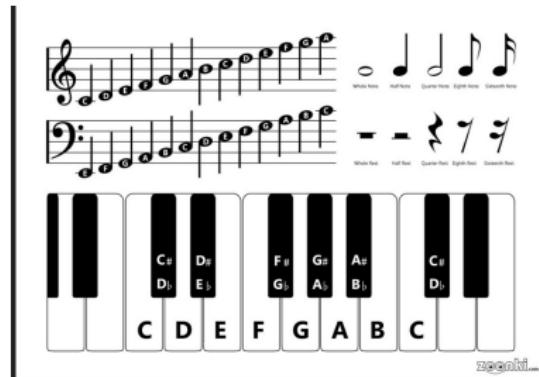
- Wysokość zaznaczamy wybierając odpowiednią pozycję na pięciolini
- Można ją skonwertować na liczbę całkowitą (numer klawisza, od lewej do prawej, licząc czarne i białe)

Muzyka: Lekcja 2 (cd)



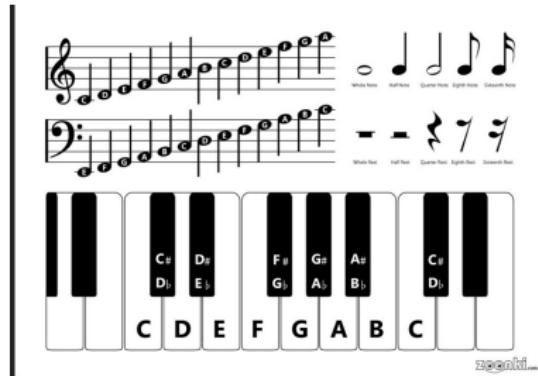
- Gdy zwiększamy wysokość o 1, to częstotliwość mnożymy przez $\sqrt[12]{n}$

Muzyka: Lekcja 2 (cd)



- Gdy zwiększamy wysokość o 1, to częstotliwość mnożymy przez $\sqrt[12]{n}$
- Kwestie nazewnictwa dźwięków (C, Cis, Des, C#, ...) sobie pominiemy (bo nie dla transformera trenowanego od podstaw nie ma najmniejszego znaczenia)

Muzyka: Lekcja 2 (cd)



- Gdy zwiększamy wysokość o 1, to częstotliwość mnożymy przez $\sqrt[12]{n}$
- Kwestie nazewnictwa dźwięków (C, Cis, Des, C#, ...) sobie pominiemy (bo nie dla transformera trenowanego od podstaw nie ma najmniejszego znaczenia)
- Kwestia tonacji również pominiemy (w korpusie wszystkie piosenki mają najprostszą tonację (C-dur, w zasadzie same białe klawisze fortepianu))

Formaty muzyczne

- Istnieje dużo formatów muzycznych
- Prostym formatem zapisującym muzykę jest format **abc**
- Uniwersalnym formatem zapisującym muzykę jest format **midi** (który umożliwia zapis utworów wieloinstrumentowych)

Format ABC

Format abc jest na prosty, można wytrenować transformer na znakach i powinien generować poprawne składniowo utwory (pomyśl na miniprojekt)

```
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA AFD :|2 dBA ABd |:
efe edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
</score>
```

Nasz korpus

- **11724** piosenek (z różnych miejsc, dobranych bez specjalnego namysłu)
- **1232166** nutek, **382** różne tokeny
- (btw. Szekspir napisał koło miliona znaków w swoich dramatach)

1 wiersz z korpusu

```
<begin-song> N71-16 P16 N71-32 N71-16 P16 N73-32 N71-32 N71-32 N73-32 P32 N71-64 N69-32 N71-32  
N69-32 N66-32 P16 N76-48 N73-16 P48 N71-64 N76-48 N73-16 P16 N71-32 N76-32 N73-32 N71-32 N69-32  
N66-32 P64 N76-96 N73-32 P96 N71-128 N61-16 P16 N64-32 N69-16 P16 N66-32 N66-16 N69-16 N66-16 N69-16  
N64-16 N66-16 P48 N69-64 N71-16 P16 N71-32 N73-16 N73-16 N71-16 P16 N69-32 N69-16 N64-16 N69-16  
N66-16 P48 N64-64 N64-16 N64-16 N69-16 N66-16 P48 N69-64 N69-32 N71-32 N66-32 N66-32 P32 N64-64  
N76-48 N81-16 P16 N78-32 N81-32 P224 N76-256 N69-16 P16 N69-32 N73-16 P16 N69-32 N71-16 N69-16 P16  
N73-32 N71-16 N73-16 P48 N71-64 N69-32 N71-32 N69-32 N66-32 P16 N76-48 N73-16 P48 N71-64 N76-48  
N73-16 P16 N71-32 N76-32 N73-32 N71-32 N69-32 N66-32 P64 N76-96 N73-32 P96 N71-128 N69-16 P16 N69-32  
N64-16 N66-16 N66-16 P16 N69-32 N66-16 N69-16 N64-16 N66-16 P48 N69-64 N73-16 P16 N73-32 N73-16  
N73-16 N71-16 P16 N69-32 N69-16 N64-16 N69-16 N66-16 P48 N64-64 N64-16 N64-16 N69-16 N66-16 P48  
N69-64 N69-16 N71-16 N69-16 N66-16 P48 N64-64 N76-48 N81-16 P16 N78-32 N81-32
```