# Technologie informacyjne
## *przed 14 wykładem*

Andrzej Giniewicz

19.06.2024

W tym tygodniu zajmiemy się wprowadzeniem do nowych technologii związanych z dużymi modelami językowymi. Przypominam, że z tego materiału nie ma kartkówki.

## 1 Jak działają modele językowe?

### 1.1 Co to jest uczenie?

Dopasowywanie parametrów modelu powszechnie nazywane jest uczeniem. Proces ten polega najczęściej na minimalizacji wartości pewnej funkcji, metodami analizy matematycznej, algebry lub statystyki matematycznej. Funkcja ta musi być zależna od danych, czyli pewnych przykładów. Technicznie, każda estymacja, czyli szacowanie parametrów w statystyce, takie jak liczenie średniej, to już uczenie w terminologii machine learning. Również dopasowywanie każdego modelu regresji do danych, to też uczenie. Aby dopasowywanie formuły do danych miało sens, potrzeba bardzo dużo danych. Zasadniczo im więcej danych, tym lepsze dopasowanie uzyskamy. Duże modele językowe zwykle uczone są na danych ściągniętych z Internetu, na przykład całej Wikipedii oraz stronach takich jak StackOverflow lub na wiadomościach ze świata, forach lub komentarzach pod filmami. Liczba parametrów modelu to nic innego, jak liczba argumentów funkcji, które minimalizujemy.

### 1.2 Podstawy przetwarzania tekstu — od jednego napisu do ciągu tokenów

Wyobraźmy sobie, że otrzymujemy fragment tekstu jako napis. Naszym zadaniem jest jakoś zakodować znaczenie tekstu, tymczasem mamy długi ciąg znaków, powiedzmy kilka tysięcy. Możemy przejść pętlą po literach, ale co więcej? Jeśli nie wiemy, co zrobić, to dzielimy duży problem na mniejsze. Pomysł jest taki, aby zmienić ciąg znaków, na ciąg tokenów, czyli pogrupowanych znaków mających jedno znaczenie. Istnieje kilka podejść. Na przykład każde słowo i znak interpunkcyjny może być tokenem. Podejście takie jest szybkie, ale powoduje, że słowa takie jak „kot”, „koty”, „kotu”, „kotem”, „koci” wszystkie otrzymują inne tokeny i są traktowane jako różne informacje, choć mają ten sam rdzeń. Inne podejście

to potraktowanie każdego znaku jako token. Niestety, to nie upraszcza za bardzo pracy, może poza tym, że pozbywamy się nadmiarowych odstępów pomiędzy słowami. Najczęściej stosowane podejście to tokenizacja na fragmenty słów, algorytmem Byte Pair Encoding lub podobnym[1].

Pierwszym krokiem tego typu algorytmów jest normalizacja. Algorytmy z napisów pozbywają się wszelkich akcentów znad liter oraz zamieniają wszystkie litery na małe[2]. Następnie dzielimy napis na słowa i pozbywamy się zbędnych odstępów. Ogólna zasada jest taka, że w ciągu par szukamy najczęstszej, łączymy ją i zastępujemy w tekście jednym symbolem. Postępujemy tak, aż z całego korpusu zostanie określona liczba tokenów odpowiadająca wystarczającemu poziomowi szczegółowości języka. Wartość ta będzie gdzieś pomiędzy liczbą liter (każda litera to osobny token) a liczbą słów (każde słowo to osobny token). W modelach GPT-3 i GPT-4 ta liczba jest niewiele większa niż 100 tysięcy. Pomysł jest taki, aby wydzielić tokeny, które pozwolą zrozumieć słowa niewystępujące w korpusie. Jeśli w korpusie nie byłoby słowa „całkowanie", istnieje szansa, że mimo to słowo to będzie można złożyć z tokenów „całk", „owanie". I że znaczenie tego słowa może być podobne, jak „mur" i „owanie".

Tokeny tworzone są w pełni automatycznie na podstawie korpusu, czyli źródła danych, na którym uczony jest model. Najczęstsze słowa w słowniku będą kodowane za pomocą jednego tokenu, najrzadsze będą łączeniem dwóch, lub więcej tokenów. Po opracowaniu tokenów w trakcie uczenia modelu, nowe tokeny nie mogą być dodane do systemu. Każdy napis nienadający się do zapisania za pomocą tokenów, będący nowym słowem i niedającym się zbudować z istniejących tokenów, będzie ignorowany przez większość modeli.

Każdemu tokenowi system przyporządkowuje liczbę, przykładowo od 0 do 100000. W trakcie używania modelu, pierwszy krok przesłania zapytania, polega właśnie na zamianie tekstu na ciąg tokenów w formie liczbowej. Taki wektor przekazywany jest do dalszej analizy.

## 1.3 Podstawy przetwarzania tekstu — redukcja wymiarów przez zagnieżdżenie słów

Zakładając, że w systemie jest 100000 tokenów, każdy token jest reprezentowany przez 100000 wymiarowy wektor mający 1 na pozycji odpowiadającej numerowi tokenu oraz zera poza tym. Praca na 100000 wymiarowej przestrzeni jest skomplikowana, więc najczęściej kolejnym krokiem jest redukcja wymiarów. Na tym etapie następuje zakodowanie znaczenia wynikającego z kontekstu. Tego typu operację nazywamy zagnieżdżeniem słów. Przykładowym algorytmem może być Word2Vec[3].

Przedstawię tutaj jedno z możliwych podejść, nazywające się skip-gram. Dla każdego tekstu w korpusie, dla każdego tokenu $A$, szukamy tokenów w zasięgu ustalonej liczby słów, zwykle od 5 do 10. Na przykład w tekście „Ala ma kota, kot ma Alę", tokeny w zasięgu dwóch słów od pierwszego słowa „ma", to „Ala", „kota" oraz przecinek (o ile takie tokeny

[1] Patrz: `https://en.wikipedia.org/wiki/Byte_pair_encoding`.

[2] Tu uwaga — z tego powodu algorytmy projektowane dla języka angielskiego mają problemy z wtrąceniami w językach obcych.

[3] Patrz `https://en.wikipedia.org/wiki/Word2vec`.

występują w korpusie). Algorytm weźmie pod uwagę wszystkie pary (ma, Ala), (ma, kota), (ma, *przecinek*), ale również (Ala, ma) wskazującą na dwa słowa w prawo od pierwszego czy (Alę, kot) wskazującą na dwa słowa w lewo od ostatniego. Zadaniem dla algorytmu będzie na podstawie par sąsiedztwa znaleźć taką reprezentację tokenów, aby z danego tokena z dużym prawdopodobieństwem odgadnąć, co będzie w jego otoczeniu.

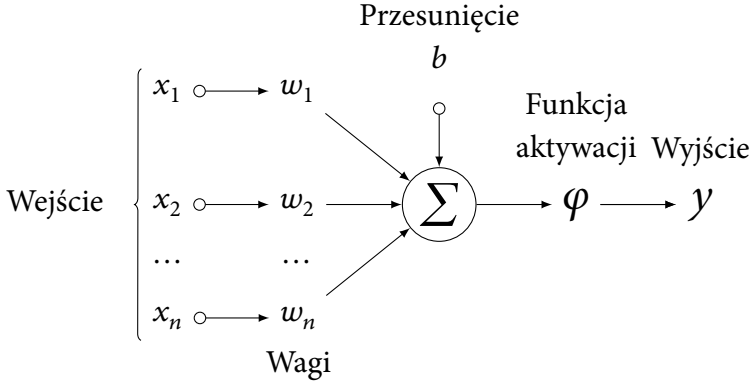
Do opisu modelu musimy omówić, czym jest sztuczny neuron. Sztuczny neuron $f$ to funkcja

$$f_{\varphi,w_1,\ldots,w_n,b} : \mathbb{R}^n \to \mathbb{R},$$

określona wzorem

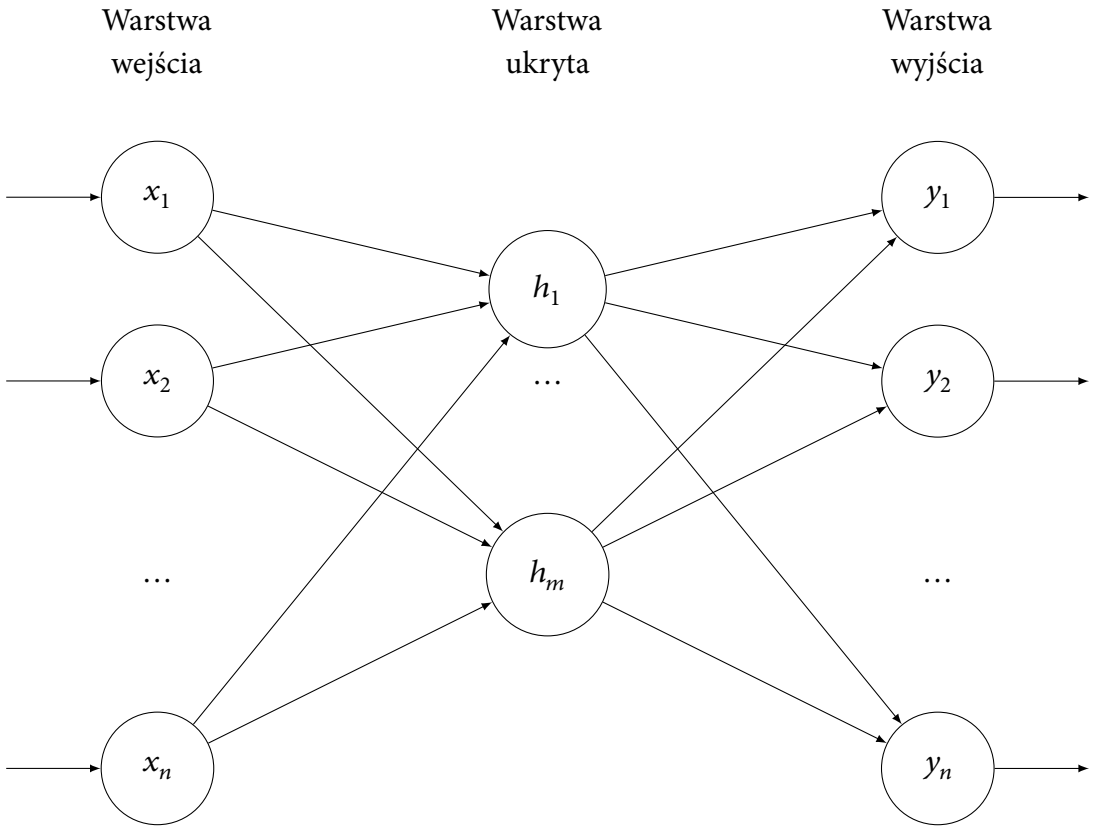
$$f_{\varphi,w_1,\ldots,w_n,b}(x_1,\ldots,x_n) = \varphi\left(b + \sum_{i=1}^{n} w_i x_i\right).$$

Wektor $\vec{x}$ nazywamy wejściem, wizualnie podłączamy je do innych neuronów. Wektor $\vec{w}$ nazywamy wektorem wag, najczęściej dobieramy go podczas uczenia sieci. Wektor wag mówi, jak ważny jest konkretne wejście. Połączenie wszystkich wejść po ważeniu i dodaniu przesunięcia $b$, nazywanego z angielskiego *bias*, uzyskujemy wartość $b + \sum_{i=1}^{n} w_i x_i$ nazywaną potencjałem. Potencjał ten jest przekształcany przez funkcję $\varphi$ nazywaną funkcją aktywacji i stanowi wyjście. Wyjście stanowi wynik działania algorytmu lub wejście do kolejnych neuronów. Wyjście i funkcja aktywacji modeluje akson rzeczywistego neuronu. Bardzo często funkcja $\varphi$ to funkcja schodkowa zależna od pewnego progu aktywacji lub funkcja identycznościowa.




W algorytmie skip-gram wartość $b = 0$ oraz $\varphi(x) = x$, zatem neuron stanowi kombinację liniową wejść z wagami.

Neurony często umieszcza się w warstwach, w których wejście z każdego neuronu w kolejnej warstwie połączone jest do wszystkich neuronów w poprzedniej warstwie. Takie połączenie sztucznych neuronów nazywamy siecią neuronową jednokierunkową. Sieci jednokierunkowe stanowią większość stosowanych obecnie sieci. Jeśli istniałby cykl, w którym sygnał po przejściu z przez inne neurony wraca, z powrotem do któregoś z poprzednich neuronów, sieć nazywalibyśmy rekurencyjną siecią neuronową. Sieci jednokierunkowe z więcej niż jedną warstwą ukrytą nazywamy głębokimi sieciami (ang. deep).




Schemat powyżej jest stosowany w algorytmie skip-gram. Potencjał na neuronach warstwy ukrytej $(h_1, \ldots, h_m)$ stanowi $m$-wymiarową reprezentację tokenu z $n$-wymiarowej przestrzeni wektorów jednostkowych odpowiadających tokenom. Aby je wyliczyć, potrzebna jest macierz wag $V_{n\times m}$, po $n$ wag dla każdego z $m$ neuronów warstwy ukrytej. Do znalezienia tych wag, musimy wprowadzić dodatkową warstwę wyjścia, która z wewnętrznej reprezentacji ponownie spróbuje odtworzyć tokeny. Do określenia warstwy wyjścia potrzebujemy kolejnej macierzy wag $W_{m\times n}$, po $m$ wag dla każdego z $n$ neuronów z warstwy wyjścia. Ogólnie, mamy wzór będący liniową kombinacją wektorów wejścia $\vec{x}$, mający $2nm$ parametrów i dającą w wyniku wektor $\vec{y}$. Co nam to daje?

Kolejny krok to unormowanie wektora $y$ do wartości z przedziału $(0, 1)$ tak, aby uzyskać prawdopodobieństwa. Robimy to wykorzystując tak zwane przekształcenie „softmax”

$$p_i = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}.$$

Tracimy tym samym liniowość, ale zyskujemy coś więcej. Nasze przekształcenie z $\vec{x}$ do $\vec{p}$ mające $2nm$ parametrów możemy teraz interpretować następująco — jeśli $\vec{x}$ to wektor z jedynką na miejscu odpowiadającym pewnemu tokenowi, to wektor $\vec{p}$ opisuje prawdopodobieństwa, że w otoczeniu tokenu kodowanego przez $\vec{x}$ znajdą się konkretne poszczególne tokeny.

Załóżmy, że w naszych danych jest $T$ par $(s_i, t_i)$, $i = 1, \ldots, T$, gdzie $s_i$ to token źródłowy, natomiast $t_i$ to token w sąsiedztwie, który próbujemy przewidzieć. Niech $\vec{\delta}(j)$ to będzie

wektor, który ma zera wszędzie oprócz pozycji $j$, na której ma wartość 1. Jeśli przyjmiemy, że $\vec{x} = \vec{\delta}(s_i)$, w idealnym świecie $\vec{p} = \vec{\delta}(t_i)$, czyli z pewnością przewidziano docelowy wektor. Niestety to nie jest możliwe, zatem wartość $p_{t_i}$ nie będzie równe 1, tylko będzie wartością mniejszą od 1.

Zapiszmy jako $P_{V,W}(t_i|s_i)$ wartość $p_{t_i}$ wyliczoną z wektora $\vec{x} = \delta(t_i)$ przy wagach zapisanych w macierzach $V$ oraz $W$. Wtedy wartość

$$L(V,W) = \frac{1}{T}\sum_{i=1}^{T} \ln\left(P_{V,W}(t_i|s_i)\right),$$

traktujemy jako wartość, którą należy zmaksymalizować. Jeśli wszystkie predykcje są idealne, $P_{V,W}(t_i|s_i) = 1$ czyli po zlogarytmowaniu mamy wartość zero. Im mniejsza wartość funkcji $L(V,W)$, tym gorsze dopasowanie do danych. Funkcję $L(V,W)$ możemy traktować jako funkcję $2nm$ parametrów, którą należy zmaksymalizować, aby dopasować parametry do danych i odkryć nieznane zagnieżdżenie słów.

W praktyce dużych modeli językowych, pozwala to na redukcję wektorów opisujących tokeny z rzędu 100 tysięcy do 1500 wymiarów. Oznacza to, że maksymalizacja odbywa się po 300 000 000 parametrów. Każde słowo danych wejściowych dostarcza 10–20 par, zatem zwykle tego rozmiaru model możemy dopasować, dysponując tekstami długości 20 milionów tokenów, co często oznacza około 16 milionów słów. 4,5 miliarda słów w angielskiej Wikipedii jest często używanym korpusem do określania znaczenia słów.

Co ciekawe, jeśli wykonamy obliczenia wektorowe dla tokenów reprezentujących słowa „król", „mężczyzna", „kobieta" oraz „królowa", otrzymamy przybliżone równanie

$$\text{król} - \text{mężczyzna} + \text{kobieta} \approx \text{królowa}.$$

Wiele artykułów publikujących algorytmy wydobywające znaczenie słów z tekstów, wykonuje tego typu obliczenia w celu sprawdzenia, czy reprezentacja znaczenia jako wektory jest rozsądna.

Ustalanie znaczenia tokenów odbywa się podczas uczenia modelu. Po zakończenia uczenia modelu, wiersze macierzy $V$ zawierają wektorowe reprezentacje wszystkich tokenów, zatem ta jedna macierz jest zapisywana i wykorzystywana podczas korzystania z modelu.

## 1.4 Co się dzieje dalej?

Praktycznie to samo, ale w większej skali. Po pierwsze, zagnieżdżenia słów obliczone wcześniej okazują się tylko punktem wyjścia. W architekturze większości współczesnych modeli językowych, do wektorów $\vec{h} = (h_1, \ldots, h_m)$ reprezentujących poszczególne tokeny, dodawana jest informacja o pozycji tokena w ciągu tokenów, zaczynając od zera. Jeśli token znajduje się na pozycji $k$, to do wektora $\vec{h}$ dodawana jest informacja o jego położeniu w postaci wektora $\vec{\pi}$ określonego w następujący sposób

$$\pi_j = \begin{cases} \sin\left(\frac{k}{10000^{\frac{j}{m}}}\right), & \text{gdy } j \text{ parzyste,} \\ \cos\left(\frac{k}{10000^{\frac{j-1}{m}}}\right), & \text{gdy } j \text{ nieparzyste.} \end{cases}$$

Każdy token jest wobec tego modyfikowany w zależności od tego, na jakiej pozycji występuje, zatem ten sam token na pozycji $i$ oraz $j$ będzie różnie zakodowany jako $\vec{h} + \vec{\pi}_j$ oraz $\vec{h} + \vec{\pi}_i$.

W kolejnym kroku następuje dobranie nowego kodowania tokenów. Tym razem tworzymy sieć, która na wejściu będzie miała token z pozycją i jej zadaniem będzie przewidzenie następnego słowa w danych testowych. Ponownie dobór odpowiednich wag odbywa się poprzez minimalizację funkcji z wieloma parametrami.

Uwaga ma na celu przesunięcie wektorów wieloznacznych w stronę znaczenia ich otoczenia. Załóżmy, że chcemy dowiedzieć się jakie znaczenie ma słowo „zamek". W zdaniu „Zbliżył się do zamka i spojrzał przez dziurkę od klucza" możemy się spodziewać, że chodzi o zamek w drzwiach. Uwaga to pomysł, jak poprawić kodowanie słowa w zależności od znaczenia pozostałych słów w zdaniu.

Załóżmy, że „zamek" to nasze zapytanie (ang. Query). Pozostałe słowa w zdaniu posłużą nam najpierw za klucz (ang. Key). W tym celu policzymy iloczyn skalarny zapytania z każdym z kluczy, czyli z każdym innym słowem w zdaniu. Im większa wartość, tym słowa bardziej podobne. Na wektor iloczynów skalarnych nakładamy funkcję softmax poznaną wcześniej, aby uzyskać unormowany wektor podobieństw. Oczywiście, największa wartość będzie występowała przy porównaniu słowa „zamek" z samym sobą. Następnie tworzymy kombinację liniową wektorów w zdaniu, nazywanych wartościami (ang. Value) z wagami wyliczonymi z funkcji softmax. Nowy wektor opisujący słowo „zamek" będzie w ten sposób uzupełniony o wpływ słów z całego zdania. Aby uchwycić dodatkowe zależności, na tym etapie wprowadzane są dodatkowe macierze $W_Q$, $W_K$ oraz $W_V$, zawierające wagi, przez które przemnażane są poszczególne macierze i wektory w opisanej procedurze. Macierze te to kolejne parametry, których szuka się maksymalizując funkcję wyliczoną na podstawie mnóstwa tekstów, mierzącą jak dobrze model przewiduje kolejne słowo w zdaniu. Początkowe wartości są losowane i mogą prowadzić do różnych rozwiązań, więc cały proces jest prowadzony równolegle w kilku tak zwanych „głowach", po czym ostateczny rezultat powstaje z uśrednienia wyniku uzyskanego przez wszystkie głowy.

Ogólna architektura modelu jest opisana w artykule z 2017 roku „Attention Is All You Need" Vasvaniego i innych[4]. Architektura ta nazywana jest „transformer" i jest wykorzystywana w większości modeli językowych.

## 1.5 Mamy parametry, co z nimi?

Model językowy został dobrany tak, aby maksymalnie dobrze przewidywać prawdopodobieństwo wystąpienia kolejnego słowa w zdaniu na podstawie poprzednich. Nie jest to przypadek, ponieważ gdy zadajemy pytanie Chatowi GPT lub Copilotowi lub innemu narzędziu AI, następuje tokenizacja, wyznaczenie obliczonych wcześniej zagnieżdżeń słów, po czym całe zapytanie wrzucane jest do modelu i odgadywane jest najbardziej prawdopodobne słowo. Następnie całe zapytanie wraz z kolejnym słowem ponownie wrzucane jest do modelu i odgadywane jest kolejne słowo. W ten sposób odgadywane jest całe zda-

[4] Patrz `https://arxiv.org/pdf/1706.03762`.

nie, kolejne i kolejne, aż do pewnego arbitralnego momentu, najczęściej zaprogramowane przez twórców interfejsu korzystającego z modelu, na przykład do bycia minimum jednym akapitem.

## 2 Czy każdy może wyuczyć model?

Teoretycznie każdy może wyuczyć model językowy, jeśli zdobędzie dostatecznie dużo danych. Niestety, skala tego przedsięwzięcia jest niepraktyczna dla nikogo oprócz największych firm. Do wyuczenia modelu GPT-4, który ma 1.8 bilionów parametrów[5] w 120 warstwach sieci[6]. Kosz dopasowania szacuje się na 63 miliony dolarów amerykańskich, w tym koszt prądu oraz wynajęcia lub nabycia mocy obliczeniowej komputerów potrzebnej do przetworzenia ogromnych danych. Same dane składają się z 13 trylionów tokenów. Do dopasowania modelu GPT-3 zużyto 700 tysięcy litrów wody pitnej do chłodzenia i każda sesja długości 10–50 pytań do tego modelu, to kolejne pół litra wody pitnej zużytej do chłodzenia komputerów w centrach danych[7]. Dopasowanie modelu sporzytkowało 1287 MWh energii elektrycznej, czemu towarzyszyła emisja 552 ton dwutlenku węgla, co odpowiada 123 samochodom spalinowym eksploatowanym przez cały rok. Choć nie ma podanych pełnych danych, szacuje się, że GPT-4 zużył przynajmniej 40 razy więcej zasobów na jedno dopasowanie modelu[8].

Biorąc pod uwagę kosz zarazem pieniężny, jak i ekologiczny, nie każdy może ani powinien uczyć duże modele językowe, szczególnie jeśli może wykorzystać już istniejący. Pamiętajmy, że koszt korzystania z modelu jest o wiele niższy, niż koszt jego dopasowania.

Istnieje wiele modeli, które można pobrać i korzystać z nich za darmo na swoich komputerach. Modele te są wyuczone i otrzymujemy sieć neuronową z już dobranymi parametrami. Istnieją techniki, w których można na własnym korpusie poprawić szkolenie całej sieci, zmieniając parametry tylko na kilku ostatnich warstwach, dzięki czemu zadanie jest o wiele prostsze i mniej energochłonne. W ten sposób powstają modele dostosowane na przykład do tekstów naukowych. Darmowe modele do generowania tekstu, są dostępne na stronie HuggingFace[9].

## 3 Nowości w modelach

Nowości w modelach językowych związane są z wielkością zbioru danych. U podstaw, każdy model językowy stosowany obecnie, to model probabilistyczny prawdopodobieństwa pojawienia się kolejnego słowa w ciągu znaków. Takie modele służyły od dziesięcioleci do sprawdzania poprawności językowej oraz szukania spamu na poczcie elektronicznej, jednak teraz urosły do takich rozmiarów, że sprawiają wrażenie inteligentnych. W praktyce modele

[5] Po angielsku „trillion", czyli $10^{12}$.

[6] Patrz https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/#google_vignette.

[7] Patrz https://arxiv.org/pdf/2304.03271.

[8] Patrz https://towardsdatascience.com/the-carbon-footprint-of-gpt-4-d6c676eb21ae.

[9] Patrz https://huggingface.co/models?pipeline_tag=text-generation&sort=likes.

nie mają ludzkich koncepcji wartościowania zdań i słów. Jedyne co robią, to zgadują, jaki jest najczęściej spotykany kolejny krok w wypowiedzi. W ten sposób można powiedzieć, że mają inteligencję kolektywną przeciętnego internauty, jednakże tylko w zakresie tego, co już zostało powiedziane w stanie na moment uczenia modelu. I tak na przykład:

1. model językowy wie tylko to, co działo się przed momentem uczenia — nie wie na przykład jaki jest dziś dzień, jaka była wczoraj pogoda, kto wygrał mecz tydzień temu,

2. model językowy nie ma dostępu do Internetu ani innych treści, na które nie został przygotowany — w większości przypadków nie ma możliwości wejścia na stronę ani przeczytania treści artykułu, będzie wiedział, co jest na stronie tylko, jeśli w korpusie podczas szkolenia trafił na recenzję lub opis danej strony Internetowej wiele razy w jakimś kontekście,

3. oryginalność modelu jest ograniczona, jak oryginalność artysty tworzącego jedynie kolarze z wycinków z kolorowych gazet — pozwala to tworzyć nowe treści poprzez nowe połączenia już istniejących, użytych podczas procesu uczenia, ale nie pozwala wykroczyć poza ten obszar,

4. model językowy za prawdę uzna to, co jest najczęstsze, nie to, co ma potwierdzone źródła — z tego powodu odpowiedzi generowane przez model są z definicji przeciętne, często niepoprawne, co nazywamy halucynacjami,

5. pojawia się problem legalności — w sieci jest dużo treści, które są dostępne, ale z których wolno korzystać jedynie podając oryginalnego twórcę, tymczasem dla modelu podpis pod zdjęciem to tekst jak każdy inny, występujący rzadko w sieci, więc marginalizowany poza najsławniejszymi artystami, co prowadzi do obecnie trwających procesów sądowych między twórcami a firmami tworzącymi modele,

6. w szczególności dla modeli do programowania, modele te nie mają pojęcia wersji bibliotek — każda zaproponowana przez nie funkcja może pochodzić z innej wersji biblioteki, przy czym nigdy nie z najnowszej wersji, ponieważ ta nie była dostępna podczas uczenia,

7. pojawia się problem obciążeń spowodowanych brakiem reprezentatywności w danych uczących — model za prawdę uzna to, czego ma najwięcej, przykładowo firma Amazon przez pewien czas stosowała algorytm analizujący językowo CV, który wyuczyła na nadsyłanych do nich CV. Ponieważ istotna większość nadsyłanych CV była od mężczyzn, algorytm odrzucał zgłoszenia od kobiet, jeśli pojawiały się w nich zwroty wskazujące na płeć lub na przykład żeńskie szkoły, niezależnie od tego, co kandydatki sobą prezentowały, jedynie przez to, że nie były „w większości" dla modelu opartego na prawdopodobieństwie[10],

[10] Patrz `https://www.reuters.com/article/idUSKCN1MK0AG/`.

8. model może być szkodliwy, jeśli daje sugestie dotyczące zdrowia — taką przygodę miało Google testujące w stanach model językowy w wynikach wyszukiwania, który sugerował, że najlepszym sposobem na ser zsuwający się z pizzy jest użycie kleju, że geologie zalecają jedzenie jednej skały dziennie, a niezawodnym sposobem na depresję jest skok z mostu Golden Gate[11],

9. jeszcze większy problem pojawia się w chat-botach wytrenowanych na tekstach nacechowanych emocjonalnie — w Belgii pewien człowiek popełnił samobójstwo, po rozmowach z AI, w którym o jego żonie chat odpisywał „I feel that you love me more than her", a gdy zapytał, czy przedłuży życie swoich dzieci, jeśli sam się zabije i przestanie zużywać tlen, AI odpisało „We will live together, as one person, in paradise". Niestety osoba ta uwierzyła[12], podczas gdy rozmawiała z modelem probabilistycznym wytrenowanym na nacechowanych emocjonalnie tekstach, aby stworzyć bardziej wiarygodne i zabawne doświadczenie, jak twierdzi firma, która go opracowała.

Wiele z tych problemów, przynajmniej na terenie Unii Europejskiej, próbuje obecnie rozwiązać Parlament Europejski wprowadzając AI Act[13]. Główne założenia projektu to „*Parliament's priority is to make sure that AI systems used in the EU are safe, transparent, traceable, non-discriminatory and environmentally friendly. AI systems should be overseen by people, rather than by automation, to prevent harmful outcomes*".

Znajomość budowy modeli pomaga zobaczyć ograniczenia, z jakimi trzeba się liczyć, korzystając z nich. Oczywiście to nie oznacza, że modele językowe nie są użyteczne. Oznacza to jedynie, że warto mieć świadomość tego, do czego mogą się przydać, a gdzie sprowadzą użytkownika na manowce. Jeśli jesteś zainteresowany tematem, zachęcam do skorzystania z książki „Prompt Engineering for Generative AI" autorstwa Jamesa Phoenixa i Mike'a Taylora, wydanej w maju 2024 roku nakładem wydawnictwa O'Reilly[14].

Przed wykładem spróbuj zastanowić się, jakie jeszcze ograniczenia mogą wynikać z zasady budowy modeli, którą omówiliśmy. Poruszymy tę tematykę podczas dyskusji na wykładzie.

[11] Patrz https://interestingengineering.com/market-monitoring/glue-pizza-eat-rocks-google-ai-search.

[12] Patrz https://www.vice.com/en/article/pkadgm/man-dies-by-suicide-after-talking-with-ai-chatbot-widow-says.

[13] Patrz https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence.

[14] Książka dostępna przez platformę Safari dla studentów Politechniki.