Gniewosz **Leliwa**

SZTUCZNA INTELIGENCJA

O czym myśli, gdy nikt nie patrzy?

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/sztuit_ebook
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-289-2855-8

SPIS TREŚCI

SŁOWEM WSTĘPU

Sztuczna inteligencja jest obecna w naszym codziennym życiu od wielu, wielu lat, mimo że nie zawsze jesteśmy świadomi jej obecności. Często działa subtelnie i dyskretnie, z tylnego siedzenia, pozostając niewidzialną dla niewprawnego oka. Pomaga nam, gdy chcemy coś wyszukać lub przetłumaczyć. Wspiera, gdy robimy zakupy online lub korzystamy z nawigacji samochodowej. Filtruje naszą pocztę i chroni przed różnego rodzaju cyberzagrożeniami.

Ale miewa też swoje mroczne strony — zamyka nas w bańce i uzależnia, gdy korzystamy z mediów społecznościowych, oglądamy filmy na YouTubie lub „scrollujemy" Instagrama czy TikToka. Jej algorytmy próbują odczytać nasze preferencje — co lubimy, a czego nie, i podsyłać nam więcej tego samego, żebyśmy, broń Boże, nie przestawali. I jest w tym piekielnie skuteczna.

Ostatnimi czasy mówi się o niej znacznie więcej niż kiedyś. Już nie tylko w serwisach branżowych i popularnonaukowych, ale i w mediach głównego nurtu. Przede wszystkim za sprawą ChatGPT, OpenAI oraz wielkich modeli językowych. I trudno się temu dziwić, bo stoimy u progu kolejnej rewolucji, która tym razem — w przeciwieństwie do poprzednich — zwraca swoją uśmiechniętą twarz do nas, zwykłych śmiertelników, czyli użytkowników końcowych.

Oczywiście nie tylko do nas, ale to w zasadzie pierwszy raz w liczącej kilkadziesiąt lat historii AI, kiedy przeciętny zjadacz chleba ma szansę tej rewolucji dotknąć, posmakować. I to jeszcze jak! Dostajemy narzędzie, które posiada bez mała całą wiedzę ludzkości i które potrafi odpowiedzieć na dowolne pytanie zadane w języku naturalnym.

Ale tak jak wuj Ben przestrzegał Petera Parkera vel (spoiler) Spider-Mana, że „z wielką mocą wiąże się wielka odpowiedzialność", tak i my nie powinniśmy zapominać o zagrożeniach związanych z wykorzystaniem i dynamicznym rozwojem sztuczniej inteligencji.

Tylko jak to zrobić? Skąd wiedzieć, co sztuczna inteligencja potrafi, czego nie potrafi, a czego prawdopodobnie nigdy się nie nauczy? Jak odróżnić prawdziwe zagrożenie od kiepskiego scenariusza science fiction? Jak odnaleźć się wśród tysięcy „clickbaitowych" tytułów i nagłówków? No cóż, najlepiej otoczyć się solidną tarczą z własnej wiedzy. Sztuczna inteligencja to fascynująca i złożona dziedzina, która wbrew pozorom bazuje na dość prostych fundamentach. A moim celem jest o tych fundamentach w przystępny sposób opowiedzieć.

***

Zacznę od tego, czym ta książka nie jest. Z całą pewnością nie jest podręcznikiem do sztucznej inteligencji. Szanowny Czytelnik nie powinien oczekiwać, że po jej przeczytaniu usiądzie przy komputerze i zacznie trenować własne modele uczenia maszynowego. Istnieje mnóstwo bardzo porządnych kursów, także online, które przy odrobinie samozaparcia pozwolą szybko i sprawnie rozpocząć „karierę w AI".

Nie jest też bajką na dobranoc ani lekturą „do poduszki". Owszem, bywa nią, a ja staram się zręcznie mieszać treści łatwe, lekkie i przyjemne z tymi stanowiącymi nieco większe wyzwanie intelektualne. Pojawią się rozdziały, w których będę od Szanownego Czytelnika wymagał skupienia, gdyż opowiem dość szczegółowo o tym, jak działają sztuczne sieci neuronowe i w jaki sposób się uczą. O wyzwaniach i niezwykle pomysłowych sposobach radzenia sobie z nimi. Każdy taki „trudniejszy" rozdział będzie opatrzony stosownym ostrzeżeniem w jednym z pierwszych akapitów.

Nie będę od Szanownego Czytelnika wymagał wiedzy wykraczającej poza szkołę średnią. Żadnej skomplikowanej matematyki, żadnych całek czy pochodnych. Wzory, wykresy i schematy ograniczę do minimum. Chciałbym Szanownemu Czytelnikowi zaproponować spójną opowieść, interesującą dla inżyniera, ale i „zjadliwą" dla osób nietechnicznych, w której będzie przestrzeń na żart, anegdotę, zadumę, refleksję i radosne „aha!" od czasu do czasu.

Zaczniemy od absolutnych podstaw, ale dotrzemy też do najnowszych modeli generatywnych, zasilających ChatGPT. Nie tylko dowiemy się, jak działają, ale wspólnie prześledzimy niesamowitą drogę, która rozpoczęła się kilkadziesiąt lat temu, a która — poprzez kolejne fascynujące odkrycia — nas do nich doprowadziła.

Podzielę się z Szanownym Czytelnikiem przemyśleniami i anegdotami z mojego życia prywatnego i zawodowego. Opowiem, jak wygląda praca w start-upie zajmującym się sztuczną inteligencją. O projektach, z których jestem szczególnie dumny, bo pomagają ratować ludzkie życie. Ale też o wpadkach i trudnościach, bo błędów nie popełnia tylko ten, kto nic nie robi.

Muszę też Szanownego Czytelnika przed czymś przestrzec. Nie będę stronił od przekłamań i uproszczeń, czasem rażących i radykalnych. Dużo bardziej zależy mi na przekazaniu ogólnej koncepcji czy sposobu myślenia niż szczegółów o tym, jak działa ta czy inna architektura. Bo szczerze wierzę, że lepiej jest coś uprościć, ułatwiając zrozumienie, a dopiero potem odkłamać, niż na dzień dobry zarzucić odbiorcę wyższą matematyką i technicznym żargonem. Najpierw uczymy dzieci dodawać i odejmować, a znacznie później — i już nie każdego — uczymy o algebrach, grupach abelowych, strukturach i działaniach.

Tyle tytułem wstępu, więcej grzechów nie pamiętam. A teraz już zapraszam Szanownego Czytelnika do fascynującej, mam nadzieję, podróży w świat sztucznej inteligencji. Nazywam się Gniewosz Leliwa i będę Państwa przewodnikiem…

W KRAINIE SYMBOLISTÓW I KONEKSJONISTÓW

Dawno, dawno temu, za siedmioma górami, za siedmioma lasami żyły sobie obok siebie dwa plemiona — Symbolistów i Koneksjonistów. Ale tak normalnie, pokojowo, jak miłośnicy psów i kotów — nie chcemy przecież zaczynać książki od jakiejś awantury.

Oba plemiona od wielu, wielu lat specjalizowały się w budowie przeróżnych robotów — maszyn, które pomagały im w ich codziennej pracy, wykonując proste i powtarzalne czynności. Nie zawsze jednak tak było. U zarania dziejów, zanim zbudowano pierwszego robota, nie było żadnych podziałów, a wszystkich ludzi, zwanych Budowniczymi, łączyło wspólne marzenie o stworzeniu sztucznej istoty na swoje podobieństwo.

Zarzewiem wszelkich niesnasek okazały się różnice w podejściu do projektowania owej sztucznej istoty, a w szczególności jej inteligencji. Część Budowniczych chciała naśladować działanie ludzkiego mózgu. Przynajmniej na tyle, na ile sama je rozumiała. Tworzyć skomplikowane sieci, które uczyłyby się poprzez nawiązywanie, wzmacnianie i osłabianie połączeń między ich węzłami. To plemię nazwało się Koneksjonistami. Wierzyli oni, że odpowiednio duża i wyrafinowana sieć, nakarmiona wystarczająco dużą ilością danych, będzie mogła nauczyć się wszystkiego, co sami potrafią. A nawet więcej, bo przecież — jako sztucznemu tworowi — nieznane jej będą nasze ludzkie ograniczenia.

Część Budowniczych była sceptyczna wobec tego pomysłu. Swojego sukcesu doszukiwali się w naśladowaniu ludzkiego umysłu, nie mózgu. W reprezentowaniu wiedzy za pomocą symboli i konceptów, którymi można manipulować przy użyciu logicznych reguł — na wzór dedukcji lub abstrakcyjnego wnioskowania. To plemię nazwało się Symbolistami i wierzyło, że zdolność myślenia symbolicznego, polegająca na świadomym manipulowaniu symbolami, jest tym, co odróżnia ludzi od reszty świata zwierząt. Że jest kluczem do zrozumienia ludzkiej inteligencji.

Ja wiem, że na pierwszy rzut oka to wszystko może wydawać się dość skomplikowane, ale spójrzmy tylko, co robiły oba plemiona, chcąc przyuczyć swoje roboty do rozpoznawania kotów na pokazywanych im zdjęciach. Symboliści przekazywali robotowi to, co sami wiedzieli o kotach — że mają dwie pary łap, dwoje uszu, ogon i słodki pyszczek z wibrysami, a także w jakich pozycjach możemy przyłapać kota podczas robienia zdjęć. Nie można również zapomnieć o tym, czego w swoich przełomowych badaniach dowiódł Marc-Antoine Fardin — że koty są cieczą, ponieważ przyjmują kształt naczynia, w którym się znajdują. Dla odmiany Koneksjoniści zbierali najróżniejsze zdjęcia kotów i pokazywali je robotowi tak długo, aż ten nauczył się je rozpoznawać.

Jak łatwo się domyślić, każde z tych podejść miało swoje wady. Symboliści musieli dostarczyć robotowi całą potrzebną wiedzę, często wykraczającą poza ich własną specjalizację. Musieli znaleźć kocich ekspertów i przekazać ich wiedzę robotowi w sposób dla niego zrozumiały. Co gorsza, szybko okazało się, że jeśli kot siedział przodem do zdjęcia, to nie było widać ogona, a jeśli tyłem — słodkiego pyszczka z wibrysami. Jeśli takich przypadków nie było wiele, to stanowiły one łatwe do obsłużenia wyjątki. Niestety, w rzeczywistości takich „wyjątków" było bardzo dużo i wciąż pojawiały się nowe. Stopień skomplikowania kociego robota rósł w zastraszającym tempie, a jego utrzymanie i rozwijanie stało się bardzo kosztowne.

Podczas gdy robot Symbolistów odnosił pierwsze sukcesy, poprawnie rozpoznając koty w niektórych pozycjach, Koneksjoniści nadal zbierali zdjęcia, bo okazało się, że ich robot do nauki potrzebuje dużo więcej przykładów, niż potrzebują ludzie, a nawet bardzo małe dzieci. Potrzebowali większych sieci, większej ilości danych i większej mocy obliczeniowej. Nie poddawali się jednak, bo wiedzieli, że obóz Symbolistów również wpadł w tarapaty.

Ich upór się opłacił — w końcu udało im się wytrenować robota do rozpoznawania kotów na zdjęciach. Robot był drogi, a sam proces uczenia bardzo czasochłonny. Co więcej, osoby wyszukujące i przygotowujące zdjęcia kotów do nauki również musiały nielicho się narobić. Nie mogło być mowy o pomyłkach, bo robot był na nie bardzo wrażliwy i uparcie powielał wszelkie błędy. Ale gra okazała się warta świeczki — robot Koneksjonistów rozpoznawał koty o wszystkich możliwych kształtach, we wszystkich możliwych pozycjach.

Koneksjoniści, zachłyśnięci swoim sukcesem, zaprosili Symbolistów na prezentację nowego robota, a ci dwoili się i troili, żeby znaleźć choć jedno zdjęcie kota, którego robot nie da rady rozpoznać. Pokazywali mu zdjęcia w dzień, zdjęcia nocą, zdjęcia z bliska i z oddali. Koty grube, chude, długowłose i bez sierści. Nic nie pomagało — robot bez pudła rozpoznawał wszystkie koty. Koneksjoniści triumfowali.

I wtedy stało się coś zaskakującego. Młody Symbolista podszedł do robota ze zdjęciem futrzanej poduszki leżącej na fotelu. Bez kota. Robot popatrzył na zdjęcie i z całą stanowczością stwierdził:

— *To jest kot.*

Koneksjoniści pobledli. Tylko młody Symbolista dopytał:

— *Dlaczego uważasz, że to jest kot?*

Robot milczał. Symbolista zwrócił się do Koneksjonistów:

— *Dlaczego robot uważa, że to jest kot?*

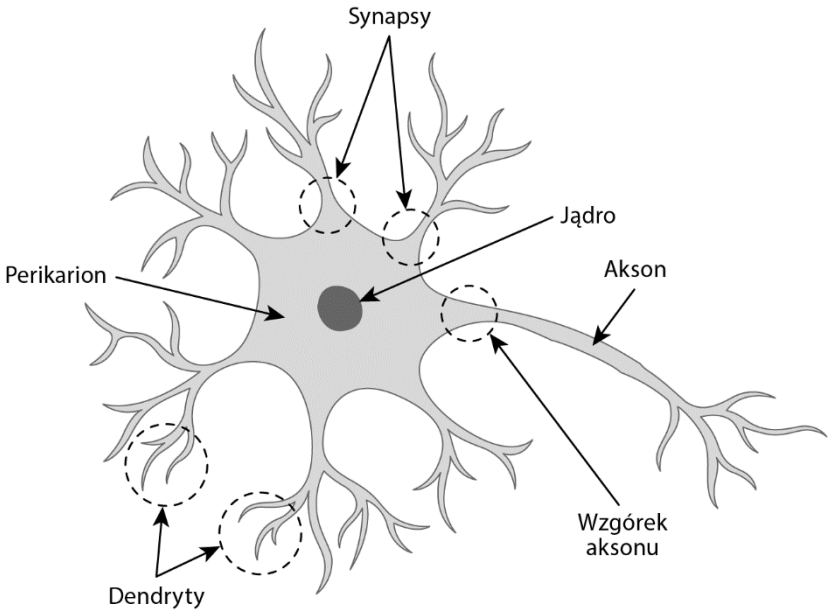
Zapadła niezręczna cisza, bo nikt na sali nie potrafił odpowiedzieć na to pytanie…

NEURONY BIOLOGICZNE I ICH SZTUCZNE ODPOWIEDNIKI

Zanim zrozumiemy, dlaczego pytania młodego Symbolisty nie doczekały się odpowiedzi, musimy dowiedzieć się, w jaki sposób działają sztuczne sieci neuronowe. Ustaliliśmy już, że mają naśladować mózg człowieka. Ale co to właściwie znaczy? Koneksjoniści, na swoje szczęście, nie musieli wymyślać koła na nowo. Mieli za wzór to, co stworzyła natura, a później ukształtowały miliony lat ewolucji. Pełnymi garściami czerpali z badań neurologów i neuropsychologów, odkrywających sekrety ludzkiego mózgu.

O neuronach biologicznych, czyli o komórkach nerwowych występujących między innymi w ludzkim mózgu, musimy wiedzieć tyle, że służą do przewodzenia i przetwarzania informacji w postaci sygnału elektrycznego, oraz że łączą się ze sobą poprzez synapsy, tworząc bardziej złożone struktury, zwane sieciami neuronowymi. Na razie w zupełności nam to wystarczy.

Na poniższym rysunku przedstawiono budowę neuronu biologicznego:

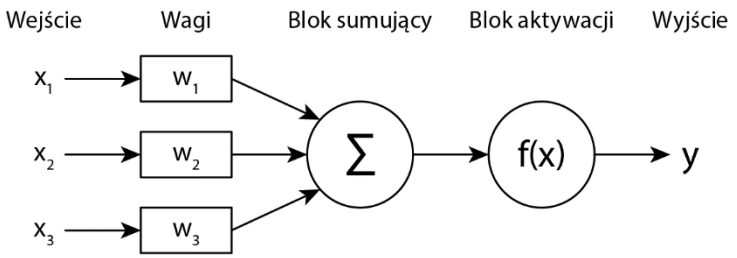


Rysunek 1. Schemat budowy neuronu biologicznego

Każdy neuron odbiera informację z zewnątrz poprzez synapsy położone na dendrytach. Pojedynczy neuron może posiadać wiele takich wejść, a w samych synapsach sygnał wejściowy może być wstępnie wzmacniany lub osłabiany. Ciało komórki nerwowej, czyli perikarion, syntetyzuje wiele różnych cząsteczek biologicznych, a jego błona komórkowa przyspiesza lub hamuje impulsy nerwowe przepływające przez neuron.

W tym miejscu sygnały wejściowe są sumowane i przekazywane dalej, do wzgórka aksonu. Jeśli łączny sygnał jest wystarczająco silny, to informacja — w myśl zasady „wszystko albo nic" — opuszcza neuron poprzez synapsy położone na aksonie. Z reguły jeden neuron posiada tylko jedno wyjście. A teraz, kiedy na skutek naszych barbarzyńskich uproszczeń zagwarantowaliśmy już biologom, a w szczególności neurobiologom, palpitację serca, możemy przejść do neuronów sztucznych.

Poniższy schemat przedstawia budowę neuronu sztucznego:



Rysunek 2. Schemat budowy sztucznego neuronu

Neurony sztuczne, podobnie jak ich biologiczni krewni, składają się z kilku analogicznych elementów. Aha, drobna uwaga: od tego momentu, gdy będę pisał o sztucznych neuronach i sztucznych sieciach neuronowych, najczęściej będę pomijał przymiotnik „sztuczny". Oczywiście, że z lenistwa. A oto te elementy:

1. Wejścia (x_1, x_2, x_3) to odpowiednik dendrytów. Przyjmują sygnał wejściowy z zewnątrz sieci lub z innych neuronów. W każdym neuronie znajduje się jedno lub kilka wejść.
2. Wagi (w_1, w_2, w_3) odpowiadają wzmacnianiu lub osłabianiu sygnału wejściowego w synapsach. Każde wejście ma swoją wagę, przez którą wymnażana jest wartość sygnału wejściowego.
3. Blok sumujący (Σ) jest odpowiednikiem perikarionu. Odpowiada za sumowanie wartości sygnałów wejściowych wymnożonych przez odpowiednie wagi.
4. Blok aktywacji (f) jest odpowiednikiem wzgórka aksonu. Wyznacza wartość przyjętej funkcji aktywacji dla sumarycznej wartości sygnału.
5. Wyjście (y) to odpowiednik aksonu. Przekazuje wartość funkcji aktywacji do kolejnego neuronu lub jako sygnał wyjściowy sieci. W każdym neuronie znajduje się tylko jedno wyjście.

Zgodnie z oznaczeniami z naszego schematu blok sumujący wykonuje następujące działanie:

$$x = x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3$$

Możemy to uogólnić na dowolną liczbę wejść i zgrabnie zapisać w postaci:

$$x = \sum_{i=1}^{liczba\ wejść} x_i \times w_i$$

Znośnie? Mam nadzieję. Matematycy, fizycy i informatycy należą do najbardziej leniwych ludzi na świecie, więc każda okazja do skrócenia zapisu jest przez nich skrzętnie wykorzystywana.

Funkcja aktywacji to po prostu funkcja jeden zmiennej. Podstawiamy *x*, odczytujemy wartość *f*(*x*) i przekazujemy ją na wyjście neuronu. Stosuje się bardzo wiele różnych funkcji aktywacji — możemy śmiało wybierać spośród skokowych, liniowych i nieliniowych. Z reguły funkcję aktywacji dobiera się w zależności od rozwiązywanego problemu, natomiast jej podstawową — nomen omen — funkcją jest wzmocnienie informacji istotnej i wytłumienie nieistotnej. Znowu — brzmi dość skomplikowanie, ale istnieją dobre praktyki i doświadczony inżynier wie, jaka funkcja będzie najlepsza dla danego problemu.

Czas na przykład. Kasia organizuje imprezę. A że z kasą ostatnio u niej krucho, to zdecydowała, że będzie to impreza składkowa. Zaprosiła swoje dwie najlepsze przyjaciółki, Emily i Jean-Marie, żeby opowiedzieć im przy herbatce i ciasteczkach — bo ustawa o wychowaniu w trzeźwości — o sieciach neuronowych.

A zatem mamy 3 uczestniczki imprezy, czyli 3 wejścia do neuronu. Niestety, przez wszechobecną inflację i drożyznę potrzeba co najmniej 200 złotych, żeby impreza w ogóle się mogła się odbyć. I ten warunek musi sprawdzać nasza funkcja aktywacji. A skoro tak, to powinna to być prosta funkcja skokowa:

$$f(x) = \begin{cases} 0\ dla\ x < 200 \\ 1\ dla\ x \geqslant 200 \end{cases}$$

Impreza będzie mogła się odbyć, jeśli na wyjściu z neuronu dostaniemy wartość 1.

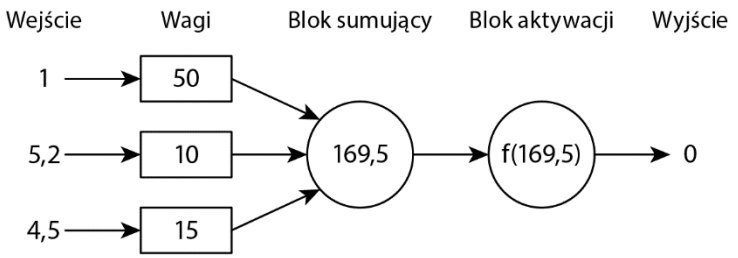
Na tym jednak nie kończą się nasze zgryzoty. Kasia pochodzi z Polski i płaci w złotówkach (PLN), Emily z Wielkiej Brytanii i płaci w funtach (GBP), a pochodząca z Francji Jean-Marie płaci w euro (EUR). Na całe szczęście możemy łatwo sprawdzić kurs funta i euro w internecie. Dla uproszczenia przyjmijmy, że 1 GBP kosztuje

5,20 PLN, a 1 EUR kosztuje 4,50 PLN. To będą nasze… wartości wejściowe ($x_1 = 1$, $x_2 = 5{,}2$, $x_3 = 4{,}5$). Zdaję sobie sprawę, że intuicja może nam płatać figla i podpowiadać, żeby zrobić z nich wagi, ale te wartości są nam znane, a to właśnie wagi mamy dobierać w procesie uczenia.

No to poszukajmy tych wag. Po ile powinny zrzucać się dziewczyny, żeby impreza mogła dojść do skutku? Z braku lepszych pomysłów zgadujemy: 50 złotych, 10 funtów i 15 euro ($w_1 = 50$, $w_2 = 10$, $w_3 = 15$). Wartości wejściowe i wagi trafiają do bloku sumującego:

$$x = \sum_{i=1}^{3} x_i \times w_i = 50 \times 1 + 10 \times 5{,}2 + 15 \times 4{,}5 = 50 + 52 + 67{,}5 = 169{,}5$$

Po chwili widzimy, że udało im się zebrać 169 złotych i 50 groszy. Podstawiamy tę wartość do naszej funkcji aktywacji i dostajemy wartość 0.

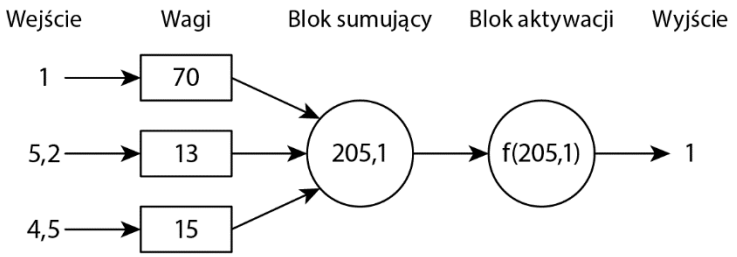


Rysunek 3. Przykład działania sztucznego neuronu dla określonych wartości (0 na wyjściu)

Niestety, wygląda na to, że Kasia i Emily przyniosły ciut za mało. Prawdziwym metodom dobierania wag poświęcimy jeszcze cały osobny rozdział. Słowo. Na razie proszę o cierpliwość i po prostu zgadujmy dalej. Niech będzie: 70 złotych, 13 funtów i 15 euro ($w_1 = 70$, $w_2 = 13$, $w_3 = 15$). Liczymy:

$$x = \sum_{i=1}^{3} x_i \times w_i = 70 \times 1 + 13 \times 5{,}2 + 15 \times 4{,}5 = 70 + 67{,}6 + 67{,}5 = 205{,}1$$

Tym razem dziewczyny zebrały 205 złotych i 10 groszy, co po podstawieniu do funkcji aktywacji daje nam wartość 1.



Rysunek 4. Przykład działania sztucznego neuronu dla zmodyfikowanych wartości (1 na wyjściu)

Sukces. Impreza się odbędzie, a Emily i Jean-Marie dowiedzą się kilku interesujących ciekawostek o sieciach neuronowych.

No dobra, to jak to jest z tymi wagami? Są wstępnie dobierane, a następnie wielokrotnie modyfikowane. Na tym właśnie polega proces uczenia — na poszukiwaniu takich wag, które dla różnych parametrów wejściowych pozwolą poprawnie przewidywać wartość parametru wyjściowego. Nie żeby to było jakieś szczególnie ważne, ale warto wspomnieć, że przyjmują wartości rzeczywiste, zazwyczaj od –1 do 1.

Jak już się rzekło — poszukiwaniem wag zajmiemy się później. A co ze wstępnym dobieraniem? Może same zera? To zdecydowanie najgorszy pomysł, bo taka sieć byłaby niezdolna do nauki. Niewiele lepiej byłoby w przypadku samych jedynek. Okazuje się, że najlepiej jest dobierać wagi losowo. A w ramach ciekawostki dodam, że losowo nie znaczy byle jak, bo istnieją lepsze i gorsze rozkłady zmiennych losowych. Ale to już materiał na zupełnie inną książkę…

***

Zmienne losowe, rachunek prawdopodobieństwa i statystyka są nierozerwalnie związane ze sztuczną inteligencją, a w szczególności z uczeniem maszynowym. I od razu dodajmy — bywają bardzo nieintuicyjne, a niektóre ich aspekty zdają się wręcz przeczyć zdrowemu rozsądkowi. Warto o tym pamiętać. A jeśli ktokolwiek twierdzi inaczej, to polecam zapoznać się z jednym z paradoksów opartych na rachunku prawdopodobieństwa — z paradoksem Monty'ego Halla.

Młodsi Czytelnicy mogą tego nie pamiętać, ale w latach 1997 – 2001 telewizja Polsat emitowała dość popularny teleturniej *Idź na całość*, którego centralnym elementem był właśnie ów paradoks. Poszczególne odcinki były dość mocno zniuansowane, ale rozgrywka sprowadzała się przede wszystkim do decydowania, czy zachować już zdobytą nagrodę, czy też próbować wymienić ją na inną, licząc się ze stratą.

Ikonicznym i powtarzalnym schematem było stawianie gracza przed wyborem jednej z trzech zasłoniętych „bramek". W jednej z nich była nagroda główna, np. samochód, a w dwóch pozostałych — nagrody pocieszenia — garnki, meble, wycieczka albo… futro z szopów pakistańskich. Nie pytajcie, lata 90. były naprawdę szalone. No i był jeszcze on — niesławny Zonk — maskotka przedstawiająca kota w worku, najgorsza spośród nagród pocieszenia. Jej nazwa na stałe zagościła w naszym języku jako synonim zdziwienia lub zaskakującej porażki.

Gracz deklarował, którą bramkę wybiera, a prowadzący — znając ich zawartość — odsłaniał jedną z dwóch pozostałych. Zawsze tę, w której nie było nagrody głównej. W tym momencie prowadzący pytał gracza, czy ten chciałby zmienić swoją decyzję i wybrać tę drugą bramkę, która wciąż pozostawała zasłonięta. Część graczy ulegała namowom, a część obstawała przy swoim. A jak postąpiłby Szanowny Czytelnik? Intuicja podpowiada nam, że nie ma to żadnego znaczenia — obu możliwym sytuacjom przypisujemy równe szanse. Prawda jest jednak zgoła inna. Statystycznie bowiem zawsze opłaca się zmienić swój pierwotny wybór. Dlaczego? Już tłumaczę…

Początkowa sytuacja jest oczywista i bezdyskusyjna. Każda bramka to dokładnie ⅓ szansy na główną nagrodę. Gracz wybiera jedną z nich, a zatem jego szansa na wygraną wynosi ⅓. A jaka jest szansa, że nagroda znajduje się w jednej z dwóch pozostałych bramek? ⅓ plus ⅓, czyli ⅔, prawda? Więc dlaczego wydaje nam się, że jeśli odsłonimy jedną z nich i okaże się, że nie ma w niej nagrody głównej, to nagle szanse obu wciąż nieodsłoniętych bramek w jakiś magiczny sposób się wyrównają?

Nasz mózg lekceważy niezwykle istotną informację — wskazanie pustej bramki — i skupia się wyłącznie na fakcie, że teraz wybieramy jedną z dwóch — a nie trzech — bramek. A ponieważ (piłka jest jedna, a) bramki są dwie, to „oczywiście" szanse wynoszą pół na pół. Prawda jest jednak taka, że teraz szansa na nagrodę

główną w bramce pierwszego wyboru to ⅓, w bramce odsłoniętej to 0, a w drugiej zasłoniętej bramce to nadal ⅔. A zatem gdy zmieniamy bramkę, nasza szansa na wygraną rośnie dwukrotnie. Postępując racjonalnie, nikt nigdy nie powinien pozostać przy swoim pierwotnym wyborze.

Jeśli Szanowny Czytelnik nadal nie dowierza, to gorąco zachęcam do rozpisania na kartce wszystkich możliwych przypadków. Jest ich dokładnie dwanaście. Można wtedy dokładnie policzyć, ile razy wygrywa strategia zmiany decyzji, a ile razy strategia obstawania przy swoim pierwotnym wyborze. Pomocne może okazać się również przeprowadzenie analogicznego rozumowania dla większej liczby bramek. Co wydarzy się, jeśli zamiast 3 będziemy mieli 100 bramek? Wybieramy jedną z nich, a prowadzący odsłania 98 „pustych" bramek. Czy nadal będziemy upierać się, że nie ma znaczenia czy zmienimy, czy też nie?

Statystyka może być potężnym sojusznikiem lub bezlitosnym wrogiem. Kasyna zarabiają mnóstwo pieniędzy nie dlatego, że krupierzy mają więcej szczęścia niż gracze. Wszystkie gry kasynowe, takie jak blackjack, ruletka czy jednoręki bandyta, są skonstruowane w taki sposób, aby zapewnić przewagę kasyna nad graczem. Zazwyczaj jest to zaledwie kilka procent, ale to w zupełności wystarczy.

Liczenie kart w blackjacku jest rodzajem strategii, która zakłada, że w pewnych specyficznych okolicznościach, gdy do rozdania pozostaje więcej kart wysokich niż niskich, matematyczna przewaga przechyla się chwilowo na stronę gracza. Dlatego też, mimo że liczenie kart nie jest nielegalne, właściciele kasyn wypraszają od stołu osoby liczące, a w skrajnych przypadkach zakazują im wstępu do swoich kasyn. Warto mieć to na uwadze, angażując się w różnego rodzaju gry losowe czy zakłady bukmacherskie.

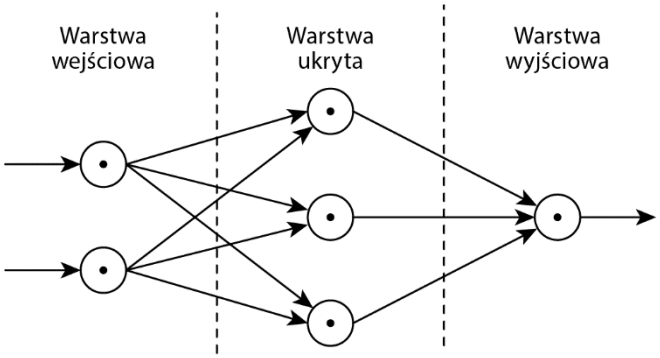
SIECI NEURONOWE SĄ JAK CEBULA

Pojedynczy neuron niewiele może. A zatem czas zbudować sieć, czyli strukturę złożoną z wielu połączonych ze sobą neuronów. Oczywiście nasz pojedynczy neuron może być przypadkiem granicznym, tworząc najprostszą możliwą sieć, ale taka sieć nie będzie ani praktyczna, ani ciekawa. Żeby rozwiązywać bardziej skomplikowane problemy, potrzebujemy znacznie więcej neuronów. Tylko jak je ułożyć? W jaki sposób połączyć?

Sieci neuronowe są jak Shrek i cebula — mają warstwy. Wyróżniamy co najmniej trzy:

- warstwę wejściową, która przyjmuje dane z zewnątrz i przesyła je dalej w głąb sieci,
- warstwę ukrytą, gdzie zachodzi proces uczenia i szukania zależności,
- warstwę wyjściową, która oblicza wartości wyjściowe sieci i zwraca wynik na zewnątrz.

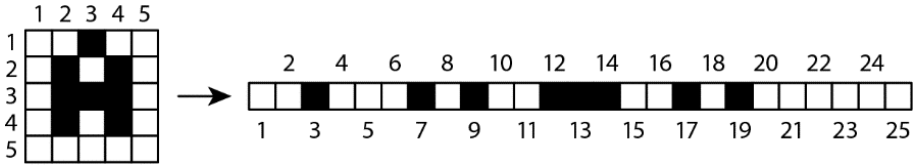
W każdej warstwie może być wiele neuronów. Ich liczba w warstwie wejściowej z reguły odpowiada liczbie cech (zmiennych) w danych wejściowych. Jeśli na wejście do sieci chcielibyśmy podawać zdjęcia, to mogą to być wartości kolejnych pikseli. Jeśli mamy tekst, to zazwyczaj będą to wyrazy lub jakieś bardziej fikuśne cechy, takie jak litery czy pary wyrazów.



Rysunek 5. Schemat jednokierunkowej sztucznej sieci neuronowej
z pojedynczą warstwą ukrytą

Tu pojawia się pierwszy haczyk. Liczba neuronów w warstwie jest stała, a to oznacza, że dane wejściowe muszą mieć zawsze ten sam rozmiar. Jeśli danymi wejściowymi są zdjęcia, to musimy ustalić jakąś akceptowalną szerokość i wysokość, np. 400 na 200 pikseli, a wszystkie niepasujące obrazy musimy odpowiednio dopasować — obciąć, przeskalować lub wypełnić braki. Musimy również przekształcić nasz dwuwymiarowy obraz w jednowymiarowy wektor, czyli w uporządkowany zbiór liczb, a w naszym przypadku — wartości pikseli.

Jeśli zdecydujemy się przyjmować obrazy o rozmiarze 64 na 64 piksele, to nasz wektor wejściowy będzie zawierał 4096 liczb (64 razy 64). Sytuację dla czarno-białego obrazu o rozmiarze 5 na 5 pikseli przedstawia następujący diagram — po lewej obraz, po prawej wektor:



Rysunek 6. Przykład mapowania dwuwymiarowego obrazu
do jednowymiarowego wektora

Czytelnik interesujący się filmem lub fotografią z pewnością zauważy, że 64 na 64 piksele to bardzo niewiele. Zdjęcia w rozdzielczości 4K składają się z 3840 na 2160 pikseli, a odpowiadający im wektor musiałby zawierać ponad 8 milionów liczb. A mówimy tu przecież tylko o warstwie wejściowej. Nie omawialiśmy jeszcze kolejnych warstw ani połączeń między nimi, ale już teraz możemy się domyślać, że taka sieć musiałaby być ogromna.

Co gorsza, redukcja wymiarów — z dwóch do jednego — powodowałaby utratę informacji o kształtach obiektów na obrazie. W reprezentacji dwuwymiarowej piksele poszczególnych obiektów sąsiadują ze sobą w obu wymiarach. W reprezentacji jednowymiarowej sytuacja się komplikuje, bo obraz zostaje pocięty na paski o szerokości jednego piksela, ułożone jeden za drugim. Sąsiedzi z lewej i z prawej pozostają na miejscu, ale sąsiada z góry i z dołu nagle dzielą dwie szerokości obrazu.

Niech to będzie drobny kamyczek, który zmąci wodę w naszym stawie. Obiecuję, że wrócimy jeszcze do tych zagadnień w kolejnych rozdziałach, bo — jak Szanowny Czytelnik może się słusznie domyślać — nie z takimi problemami poradzili sobie prawdziwi superbohaterowie — naukowcy, wynalazcy i inżynierowie.

Przejdźmy zatem do drugiego popularnego źródła danych wejściowych — tekstu. Tu sytuacja wygląda bardzo podobnie. Musimy założyć jakąś stałą liczbę cech, a dla uproszczenia niech to będą wyrazy. I tu pojawia się drugi haczyk — sieć neuronowa potrzebuje wartości liczbowych, nie wyrazów. No co za niefart. Cóż można z tym począć?

Możemy użyć słownika, który przyporządkuje wyrazom jakieś wartości liczbowe, które wypełnią nasz wektor. Jeśli mamy, to możemy skorzystać z gotowca. Jeśli nie, możemy zbudować swój własny słownik, w oparciu o dostępny zbiór danych tekstowych. Takie słowniki zazwyczaj trochę się ogranicza, np. biorąc pod uwagę tylko wyrazy, które w danym korpusie tekstowym wystąpiły co najmniej 10 razy. Słowo „korpus" ma oczywiście wiele znaczeń, ale w tej książce będzie zawsze oznaczało jedno i to samo — reprezentatywny zbiór danych.

Ale jak to ogranicza? A co, jeśli potem mimo wszystko pojawi się taki rzadki wyraz? No cóż, dokładnie to samo, co w przypadku wyrazów, które w ogóle nie wystąpiły w naszym korpusie. Zwykle zostawia się furtkę w postaci specjalnego wyrazu —

oznaczmy go jako „<nieznany>", do którego wrzuca się zbiorczo wszystkie wyrazy spoza słownika. I zwykle nie stanowi to problemu, bo przecież i nam, ludziom, zdarza się nie znać jakiegoś słowa. Bardzo często możemy domyślić się jego znaczenia z kontekstu.

To, jakich cech zdecydujemy się użyć na wejściu, ma fundamentalne znaczenie dla całego procesu uczenia sieci. W przypadku bardziej skomplikowanych architektur możemy pokusić się na przykład o użycie liter zamiast wyrazów. Pomysł na pierwszy rzut oka może wydawać się kontrowersyjny, ale ma swoje zalety. Okazuje się bowiem, że takie podejście znacząco uodparnia model na różnego rodzaju literówki. Łatwo to zresztą wyjaśnić — wyraz zapisany z błędem raczej nie występuje w korpusie na tyle często, żeby trafić do słownika.

***

Zanim przejdziemy do warstwy ukrytej, skierujemy jeszcze na moment nasze dociekliwe spojrzenia na warstwę wyjściową. Ile tam powinno być neuronów? Tu odpowiedź jest w miarę prosta — powinno być tyle neuronów, ilu wartości liczbowych potrzebujemy, żeby w wystarczający sposób reprezentować wynik działania sieci, czyli predykcję naszego modelu.

Jeśli rozwiązujemy problem klasyfikacyjny, czyli chcemy, żeby nasza sieć przyporządkowywała dane wejściowe do jednej z kilku możliwych klas (kategorii), to chcemy mieć dokładnie tyle neuronów, ile mamy kategorii. A ponieważ liczba neuronów w warstwie ukrytej, podobnie jak w każdej innej warstwie, jest stała, to musimy dokładnie wiedzieć, czego szukamy.

Pora na przykład. Problemem klasyfikacyjnym może być rozpoznawanie zwierząt na zdjęciach. Jeśli chcemy, żeby nasz model oceniał, czy na zdjęciu jest żubr, bóbr, łoś, lis, wilk, kuna, koń, wydra, ryjówka, zając, to każde z tych zwierząt stanowić będzie odrębną klasę, reprezentowaną przez pojedynczy neuron w warstwie wyjściowej. A jeśli Szanowny Czytelnik, czytając powyższą listę, dodał w myślach co nieco przed łosiem, to gratuluję znajomości polskiej kinematografii.

A zatem mamy 10 zwierząt i 10 neuronów w warstwie wyjściowej. W rzeczywistości, budując taki klasyfikator, najprawdopodobniej dodalibyśmy jeszcze jedną lub

dwie klasy, reprezentujące zbiorczo inne zwierzęta i/lub brak jakichkolwiek zwierząt na fotografii. Niech ostatecznie będzie 12 klas i 12 neuronów, czyli dodajemy „inne" oraz „brak".

Jakie wartości mogą przyjmować te neurony? Jakie tylko chcemy. Jeśli sięgniemy pamięcią do poprzedniego rozdziału i połączymy kropki, to szybko dojdziemy do wniosku, że zależy to wyłącznie od wybranej funkcji aktywacji. Z reguły wygodnie jest stosować wartości z zakresu od 0 do 1, bo można je łatwo interpretować jako prawdopodobieństwo przynależności do danej klasy. Jeśli na wyjściu dostaniemy 0,9 dla łosia, 0,05 dla konia i mniej niż 0,01 dla pozostałych klas, to wiemy, że na zdjęciu najprawdopodobniej jest łoś.

Inny przykład. Dla odmiany coś z przetwarzania języka naturalnego, czyli klasyfikacja tekstu. Być może Szanownemu Czytelnikowi obił się o uszy termin „analiza sentymentu". Jeśli nie, to nic nie szkodzi — już tłumaczę. Analiza sentymentu polega na ocenie wydźwięku emocjonalnego wypowiedzi, np. recenzji filmu czy książki. Chcemy wiedzieć, czy dana opinia jest pozytywna, negatywna, czy neutralna. Można to oczywiście dalej udziwniać i komplikować, ale w najprostszym możliwym scenariuszu mamy trzy klasy, czyli trzy neurony w warstwie wyjściowej.

Na wejście sieci podajemy treść analizowanej recenzji, a na wyjściu odczytujemy tzw. sentyment, np. 0,7 — pozytywny, 0,2 — neutralny i 0,1 — negatywny. W takim przypadku wiemy, że dana opinia najprawdopodobniej jest pozytywna. Analiza sentymentu jest szeroko stosowana w bardzo różnych obszarach — od marketingu i badań rynku, przez monitorowanie opinii klientów i reputacji własnej marki, aż po systemy rekomendacyjne, podpowiadające nam, gdzie zjeść, co kupić i jaki film obejrzeć wieczorem.

Mamy wejście, mamy wyjście, to teraz w końcu czas na… dygresję. Tak jest! To po prostu zbyt dobre miejsce, żeby nie wtrącić kilku słów o zbiorach danych, na których trenowane są sieci neuronowe. Procesowi uczenia poświęcimy cały osobny rozdział, ale przygotujmy sobie pod to odpowiedni grunt. Jak zatem wygląda zbiór uczący dla analizy sentymentu? To nic innego jak zbiór wielu recenzji produktów, gdzie treści każdej recenzji przypisana jest — najczęściej przez człowieka — etykieta odpowiadająca jej wydźwiękowi: pozytywnemu, negatywnemu lub neutralnemu.

W największym skrócie chodzi o to, aby nasza sieć — na podstawie tych oznaczonych przykładów — nauczyła się, jakie cechy treści na wejściu sprawiają, że dana recenzja jest oceniana jako pozytywna, negatywna bądź neutralna, i żeby następnie można było użyć tej wyuczonej już sieci do przypisywania etykiet zupełnie nowym, niewidzianym wcześniej recenzjom.

Proces przypisywania etykiet danym, wykorzystywanym później między innymi do trenowania i testowania sieci neuronowych, nazywa się anotacją danych. Najczęściej zajmują się tym odpowiednio przeszkolone osoby, które nazywa się anotatorami. Temu procesowi również poświęcimy cały osobny rozdział.

A czy istnieją jakieś inne problemy, poza klasyfikacją? Istnieją, a jakże! Nie zawsze musi przecież chodzić o włożenie czegoś do jednego z kilku koszyków. Czasem chcielibyśmy — tak zwyczajnie, po ludzku — zaszaleć i uzyskać jakąś wartość numeryczną. Mówimy wtedy o regresji, czyli o estymacji wartości jednej zmiennej, tzw. objaśnianej, w oparciu o zadane wartości innych zmiennych, tzw. objaśniających.

Absolutnie sztandarowym przykładem, prezentowanym w wielu kursach data science, jest przewidywanie ceny nieruchomości w oparciu o takie zmienne objaśniające jak metraż, data budowy czy odległość od centrum. Na wejściu do sieci mamy tyle neuronów, ile jest zmiennych objaśniających, a na wyjściu — tylko jeden neuron, reprezentujący wartość numeryczną zmiennej objaśnianej. W naszym przypadku jest to cena nieruchomości.

***

Autor *Małego Księcia*, Antoine de Saint-Exupéry, mógł wiedzieć co nieco na temat sieci neuronowych, bo napisał, że „najważniejsze jest niewidoczne dla oczu". Być może w pełnym cytacie było też coś o „widzeniu sercem" i chodziło mu o coś zupełnie innego niż warstwy ukryte, ale dlaczego nie mielibyśmy zabawić się we współczesnego dziennikarza i wyciąć z kontekstu tylko to, co nam pasuje? Przechodzimy zatem do tego, co niewidoczne dla oczu.

W przeciwieństwie do warstwy wejściowej i wyjściowej warstw ukrytych może być więcej niż jedna. Przyjęło się mówić, że jeśli sieć ma co najmniej dwie warstwy ukryte, to jest to sieć głęboka i mamy wtedy do czynienia z głębokim uczeniem

maszynowym. Im więcej warstw ukrytych, tym bardziej skomplikowane problemy sieć jest w stanie rozwiązywać.

O tym, ile warstw ukrytych zastosować, decyduje twórca sieci. Decyzja jest arbitralna, ale wiadomo skądinąd, jakiego typu problemy jest w stanie rozwiązywać sieć z jedną warstwą, jakie z dwiema, a do jakich problemów potrzeba znacznie głębszych sieci. Można by się tu nawet pokusić o jakiś przepis czy receptę, ale wymagałoby to użycia przynajmniej kilku brzydkich wyrażeń, takich jak „przestrzeń sygnałów wejściowych" czy „obszar decyzyjny". Oszczędzę tego Szanownemu Czytelnikowi, a w zamian zadamy inne ważne pytanie: czy zawsze więcej znaczy lepiej?

Otóż nie, nie zawsze. W przypadku zbyt dużej liczby warstw wejściowych możemy doprowadzić do tzw. przeuczenia sieci, czyli do sytuacji, w której model doskonale wręcz dopasowuje się do danych uczących, ale zanika jego zdolność uogólniania, przez co kompletnie nie radzi sobie z nowymi przypadkami. Nie spełnia zatem swojej podstawowej funkcji.

Taki model zapamiętuje wszystko i zaczyna dopasowywać się do przypadkowych błędów w danych uczących, bo — upraszczając — jest zbyt duży i zbyt chłonny w stosunku do rozmiarów i zawartości informacyjnej danych w zbiorze uczącym. Odwrotnością przeuczenia jest niedouczenie i — jak łatwo się domyślić — może być związane ze zbyt małą liczbą warstw ukrytych lub ze zbyt małą liczbą neuronów w tych warstwach. I tu dochodzimy do kolejnego ważnego pytania: ile powinno być neuronów w warstwie ukrytej?

To kolejny dość skomplikowany problem. Nie chcemy przecież, żeby model się nam przeuczył, ale nie chcemy też, żeby się nie douczył. Na szczęście są trzy banalnie proste zasady, których wystarczy przestrzegać, żeby nie wpaść w tarapaty. Po pierwsze: neuronów w warstwie ukrytej ma być tyle, żeby było dobrze. Po drugie: ani za dużo, ani za mało. I po trzecie: w sam raz.

Żarty żartami, ale nie ma niestety gotowego wzoru na liczbę neuronów w warstwie ukrytej. Są za to różnego rodzaju „praktyczne" rady, mówiące o tym, od czego warto zacząć poszukiwanie właściwej liczby. Jedne mówią, że to gdzieś pomiędzy liczbą neuronów w warstwie wejściowej i wyjściowej. Inne, że ich liczba nie powinna

być większa niż podwojona liczba neuronów w warstwie wejściowej. A jeszcze inne podają bardzo precyzyjny przepis „na start", w postaci $^2/_3$ liczby neuronów w warstwie wejściowej plus liczba neuronów w warstwie wyjściowej.

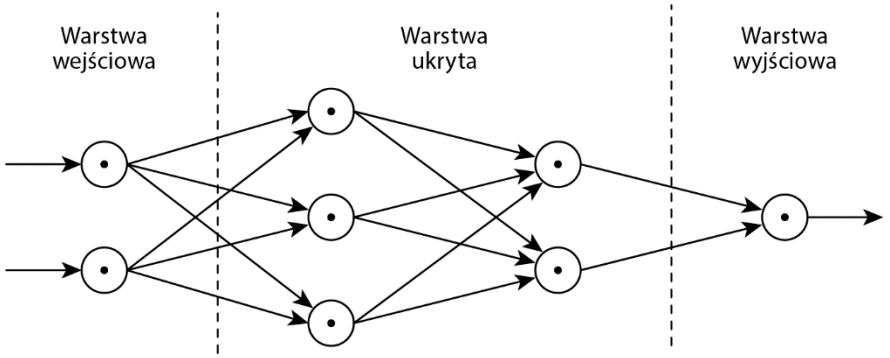
Nie wygląda to zbyt zachęcająco, prawda? No cóż, jeśli na wejściu mamy 600 neuronów, a na wyjściu 100, to wiemy przynajmniej, poza jaki zakres raczej nie warto wychodzić. A z braku lepszych pomysłów nie zaszkodziłoby po prostu zacząć od tych 500 neuronów, wskazywanych przez nasz precyzyjny przepis „na start". W praktyce wygląda to tak, że „właściwą" liczbę neuronów i warstw ukrytych określa się metodą prób i błędów, korzystając z tego typu zasad, ale przede wszystkim z własnego doświadczenia.

Gdy już znajdziemy „właściwą" liczbę neuronów, czyli taką, dla której nasza sieć dobrze się uczy, możemy optymalizować jej wielkość, stosując tzw. „pruning", co można przetłumaczyć na język polski jako „przycinanie". W największym skrócie chodzi o to, żeby w procesie trenowania sieci znajdować i usuwać neurony, których wpływ na jej działanie jest minimalny lub zerowy. Istnieje wiele różnych algorytmów przycinających, ale to już szczegóły, które nie będą nam do szczęścia potrzebne.

***

To, czego z całą pewnością potrzebujemy do szczęścia, to połączenia między poszczególnymi neuronami w sieci. Ale tu, dla odmiany, będzie dość łatwo — każdy neuron z warstwy N łączy się ze wszystkimi neuronami z warstwy N+1. A to oznacza, że neurony nie łączą się między sobą w obrębie tej samej warstwy ani neuron z warstwy pierwszej nie łączy się bezpośrednio z neuronami z warstwy trzeciej. I to w zasadzie wszystko, jeśli chodzi o ten aspekt.

W pełni połączoną sieć neuronową z czterema warstwami, zawierającą 2 neurony w warstwie wejściowej, 3 neurony w pierwszej i 2 neurony w drugiej warstwie ukrytej oraz 1 neuron w warstwie wyjściowej, przedstawia następujący schemat:



Rysunek 7. Schemat jednokierunkowej sztucznej sieci neuronowej
z podwójną warstwą ukrytą

Łatwo policzyć, że taka prosta 4-warstwowa sieć zawiera 14 połączeń. Rozważana wcześniej sieć, zawierająca 600 neuronów w warstwie wejściowej, 500 w ukrytej i 100 w wyjściowej, miałaby 350 tysięcy połączeń. A gdybyśmy chcieli do niej dodać kolejną warstwę ukrytą, również zawierającą 500 neuronów, to liczba połączeń urosłaby do 600 tysięcy. Z kolei 4-warstwowa sieć, zawierająca odpowiednio 4000, 2000, 2000 i 1000 neuronów, miałaby już 14 milionów połączeń. Pięknie rośnie, prawda? Jak liczba domorosłych wirusologów w początkach pandemii COVID-19.

Myślę, że wyrobiliśmy już sobie pierwsze intuicje, co do liczby połączeń w sieciach neuronowych. A z poprzedniego rozdziału wiemy, że na wejściach poszczególnych neuronów mamy wagi, które ustala się w procesie trenowania sieci. Drobny wyjątek stanowią tu neurony z warstwy wejściowej. Są one pasywne w tym sensie, że nie zmieniają wprowadzanej informacji, a jedynie powielają ją i przekazują dalej, do pierwszej warstwy ukrytej. A to z kolei oznacza, że nie korzystają z wag — co wchodzi, to wychodzi. Z perspektywy sieci mamy zatem tyle samo wag, co połączeń. I jest to bardzo ważna liczba, często używana do opisu złożoności sieci, nazywana również liczbą parametrów modelu uczenia maszynowego.

No to podrzućmy coś spektakularnego, żeby poczuć skalę — ChatGPT. Szanowny Czytelnik z pewnością słyszał o tym najpopularniejszym na świecie chatbocie, stworzonym przez firmę OpenAI. Ba, być może to właśnie on stał się asumptem do sięgnięcia po niniejszą lekturę.

Gdy piszę te słowa, modelem dostępnym publicznie w ramach subskrypcji ChatGPT Plus jest wersja GPT-4. Będziemy wracać do niego wielokrotnie na łamach tej książki, ale na razie skupmy się wyłącznie na liczbie warstw i parametrów, o których dowiedzieliśmy się wskutek (nie)fortunnego wycieku danych do internetu. Nie przeciągając ani chwili dłużej — model GPT-4 zawiera ponad 1,7 biliona parametrów na 120 warstwach sieci. I mam tu na myśli bilion rozumiany „po polsku", czyli milion milionów — jedynka i dwanaście zer. Robi wrażenie, prawda?

Na koniec jeszcze jedno małe sprostowanie. Istnieje bardzo wiele rodzajów i architektur sieci neuronowych. Na razie skupiamy się wyłącznie na najprostszych z nich — tzw. perceptronach wielowarstwowych (MLP, ang. *Multilayer Perceptron*), nazywanych również sieciami typu *feed-forward* (FNN, ang. *Feed-forward Neural Network*) ze względu na kierunek przepływu informacji.

Mimo swej relatywnej prostoty nadal są bardzo chętnie stosowane i pozostają jednym z najpopularniejszych typów sztucznych sieci neuronowych. I o ile dwa kolejne rozdziały też będą przede wszystkim o nich, to później wypłyniemy na naprawdę szerokie wody. Obiecuję. Zależy mi jednak na porządnym zrozumieniu tych podstaw, bo stanowią one solidny fundament dla tych nadchodzących rozważań.

W dalszych rozdziałach porozmawiamy także o innych rodzajach i architekturach sieci neuronowych — o CNN-ach, RNN-ach i transformerach. Ale nie po to, by terroryzować kogokolwiek żmudną matematyką, tylko po to, by opowiedzieć fascynującą historię o ludzkiej kreatywności w starciu z własnymi i technologicznymi ograniczeniami.

JAKIM CUDEM TO W OGÓLE MA DZIAŁAĆ?

Wiemy już, jak wygląda prosta sieć neuronowa. Ale w jaki sposób taka dziwaczna struktura miałaby się czegokolwiek nauczyć, a nauczona — rozwiązywać rzeczywiste problemy? Uchyliliśmy już rąbka tajemnicy, że chodzi o ustalanie wag na wejściach do neuronów. Zanim przejdziemy dalej, uściślijmy jeszcze kwestię przepływu informacji w sieci.

Perceptrony wielowarstwowe są sieciami jednokierunkowymi, z angielskiego *feed-forward*, co można niezgrabnie przetłumaczyć jako „sprzężenie w przód". Oznacza to tylko tyle, że informacja płynie wyłącznie w jedną stronę — od neuronów w warstwie wejściowej, przez neurony w kolejnych warstwach ukrytych, aż do warstwy wyjściowej. Bez żadnych zawrotek, cyklów czy pętli. Takie podejście może wydawać się oczywiste, ale istnieją sieci, które umożliwiają dwukierunkowy przepływ informacji, i warto o tym pamiętać.

No to teraz zapnijcie pasy, bo spróbujemy wyjaśnić, jak to wszystko w ogóle może działać. Na razie jeszcze bez uczenia. Zapomnijcie o uczeniu. Załóżmy, że mamy już nauczoną sieć.

Czy Szanowny Czytelnik jest sobie w stanie wyobrazić, że taka sztuczna sieć neuronowa:

- z odpowiednim wejściem i wyjściem,
- z odpowiednią liczbą parametrów,
- z odpowiednio ustalonymi wagami,

- z odpowiednio dobranymi funkcjami aktywacji,
- która przekazuje i przetwarza informację od wejścia, przez kolejne warstwy ukryte, aż do wyjścia, zgodnie z tym, co omówiliśmy w poprzednich rozdziałach,

jest w stanie rozwiązywać relatywnie skomplikowane problemy, takie jak analiza sentymentu?

To jest kluczowy moment, bo bez przekonania, że jest to możliwe, wszystko inne będzie budowane na zwątpieniu, które jest dość lichym fundamentem. Ale też powiedzmy sobie jasno: nie ma absolutnie nic złego w niedowierzaniu i wątpieniu. Wręcz przeciwnie — to podstawa rozwoju nauki.

Dlatego proponuję wziąć teraz na warsztat bardzo konkretny przykład i „tłuc" go tak długo, aż wszystko stanie się jasne. Niech to będzie analiza sentymentu. Po pierwsze wiemy już, co to jest. Po drugie większość z nas miała styczność z pozytywnymi i negatywnymi recenzjami lub komentarzami w mediach społecznościowych. A po trzecie przygotowanie wejścia dla danych tekstowych może być bardzo ciekawym i pouczającym doświadczeniem. Zaczynamy…

Z poprzedniego rozdziału wiemy, że gdy na wejście sieci chcemy podać tekst, to najpierw musimy przekształcić go do reprezentacji wektorowej, czyli do uporządkowanego zbioru liczb. A jak to robimy? Bierzemy korpus, budujemy z niego słownik, odcinamy rzadko występujące wyrazy. I mamy. Proste? No nie, w ogóle nie proste. Przecież to były same ogólniki. Jeśli Szanowny Czytelnik bezwiednie przytakiwał mi w myślach, to gorąco zachęcam do większej uważności, by nie ulegać takim tanim chwytom erystycznym.

Zacznijmy od wag — jaką funkcję pełnią w sieci? Mają wzmacniać lub osłabiać poszczególne sygnały w taki sposób, żeby cechy istotne miały duży wpływ na wyjście, a cechy mało istotne lub nieistotne — wręcz przeciwnie. Co w takim razie mogłoby znaleźć się w tym naszym wektorze wejściowym, żebyśmy mogli to skutecznie wzmacniać lub osłabiać?

No to może coś, co reprezentowałoby kolejne słowa z treści recenzji? Tylko jak wtedy miałaby taka sieć działać, skoro zdanie „Film był niesamowity" zmienione

w koszmar polonistki z podstawówki, czyli w zdanie „A więc film był niesamowity", miałoby pozytywnie nacechowane słowo „niesamowity" raz na trzeciej, a raz na piątej pozycji? Możemy się domyślać, że takie słowa będą bardzo ważne z perspektywy analizy sentymentu. Jak mielibyśmy wzmocnić odpowiadający im sygnał, skoro raz są w jednym miejscu wektora wejściowego, a zaraz potem kompletnie gdzie indziej?

No dobrze, czyli raczej wolelibyśmy, żeby słowo „niesamowity" było zawsze w tym samym miejscu, bo wtedy łatwo możemy wyobrazić sobie wzmacnianie ważnych sygnałów i wygaszanie nieistotnych. To już coś. Moglibyśmy na przykład zbudować taki wektor w oparciu o nasz słownik. Tylko jak to zrobić, skoro w poszczególnych recenzjach i komentarzach występuje tylko garstka słów z całego słownika?

Wektor to zbiór uporządkowanych liczb. I właśnie to uporządkowanie odegra tu kluczową rolę. Dzięki niemu każde słowo ze słownika może mieć swoją ustaloną pozycję, a my wciąż będziemy mieli pełną dowolność, jeśli chodzi o wartość odpowiadającą danej pozycji.

Możemy na przykład zliczyć wystąpienia poszczególnych słów w treści recenzji. Wtedy wszystkie pozycje w wektorze odpowiadające wyrazom, których akurat nie ma w danym tekście, będą wypełnione zerami. Będziemy też wiedzieć, które słowa wystąpiły kilkukrotnie, co może okazać się przydatne, szczególnie w przypadku słów wartościujących.

Załóżmy, że zbudowaliśmy sobie niewielki słownik, który wygląda tak:

[„film", „serial", „książka", „dobry", „niesamowity", „fajny", „zły", „słaby", „fatalny"]

Brakuje w nim bardzo wielu wyrazów, ale nic nie szkodzi. Słowa mają ustaloną pozycję, więc dla naszej krótkiej, choć treściwej recenzji

„Film był niesamowity. To kolejny dobry film tego reżysera"

możemy zliczyć wystąpienia poszczególnych słów i przedstawić wektor wejściowy w następującej postaci:

[2, 0, 0, 1, 1, 0, 0, 0, 0]

Widzimy, że rozmiar wektora wejściowego jest taki sam jak rozmiar słownika — zawiera tyle samo elementów. W rzeczywistych zastosowaniach słowniki i wektory wejściowe są bardzo duże. Korpusy tekstowe używane w przetwarzaniu języka naturalnego nierzadko zawierają kilkadziesiąt lub nawet kilkaset tysięcy unikalnych słów. Skąd aż tyle?

Słowo zawierające literówkę to nowe unikalne słowo. Słowo z przyklejonym na końcu znakiem interpunkcyjnym również. Słowo zapisane z myślnikiem i bez to dwa różne słowa. Między innymi dlatego tak ważnym etapem budowy modelu uczenia maszynowego jest odpowiednie przygotowanie danych, czyli tzw. preprocessing. Osoby zawodowo zajmujące się tym tematem spędzają od 50% do 80% swojego czasu pracy na zbieraniu i przygotowywaniu danych. To pokazuje, jak istotny — i żmudny — jest to proces.

Standardowym podejściem jest usuwanie znaków interpunkcyjnych oraz innych elementów, które nie wnoszą wartości z perspektywy rozwiązywanego problemu. Gdy budujemy model do analizy sentymentu, nie potrzebujemy tych wszystkich kropek i przecinków. Ale gdybyśmy chcieli zbudować model do korekty interpunkcji, sytuacja wyglądałaby zgoła inaczej.

Najczęściej do takich niechcianych elementów zalicza się tzw. stop words, czyli wyrazy nieinformatywne, nieniosące żadnych istotnych treści, takie jak spójniki, zaimki, przyimki czy liczebniki. W języku polskim z reguły jest to lista kilkuset słów. Zazwyczaj są to najpopularniejsze słowa w danym języku, ale to raczej nie powinno nikogo szczególnie dziwić.

Innym zabiegiem znacząco redukującym wielkość słownika jest sprowadzenie wielu odmian danego słowa do jednej ustalonej formy, np. słowa „problemami", „problemów" czy „problemy" sprowadza się do formy podstawowej — „problem". Ten zabieg nazywa się stemmingiem lub lematyzacją, a różnica między nimi jest taka, że lematyzacja sprowadza wyraz do formy podstawowej, zaś stemming do tzw. rdzenia, który nie musi być poprawnym słowem.

Na koniec, jak wspominaliśmy w poprzednim rozdziale, redukujemy wielkość słownika, odcinając wszystkie słowa, które w naszym korpusie nie wystąpiły wystarczająco często. Mamy zatem wektor, którego rozmiar odpowiada liczbie słów w słowniku. Pozycja składowej wektora odpowiada pozycji słowa w słowniku,

a jej wartość — liczbie wystąpień danego słowa w aktualnie analizowanym tekście. Teraz nietrudno wyobrazić sobie, że nasza odpowiednio wyuczona sieć „wie", które słowa wpływają na to, że dana recenzja jest pozytywna, negatywna lub neutralna.

***

Spostrzegawczy Czytelnik szybko jednak zauważy, że budując taki wektor wejściowy, tracimy całą informację o kolejności słów w analizowanych treściach. Jeśli recenzent napisze „Film nie był zbyt ciekawy", to istnieje spora szansa, że nasza sieć błędnie sklasyfikuje tę recenzję jako pozytywną. To niestety cena, którą płacimy za przyjęcie tak uproszczonej reprezentacji danych wejściowych. To nie znaczy, że nasz model będzie bezużyteczny. Jeśli tego typu sytuacje są rzadkie, to statystycznie nasz model może osiągać całkiem dobre rezultaty.

Ale nas to oczywiście nie satysfakcjonuje. Jesteśmy zbyt ambitni, żeby zadowolić się takim rozwiązaniem. Czy jest zatem coś, co możemy z tym fantem zrobić, bez składania ofiary całopalnej z naszej prostej struktury sieci? Jest, a jakże!

Nasz pierwszy model reprezentacji wektorowej nosi nazwę *bag-of-words*, co dość zgrabnie można przetłumaczyć jako „worek słów". Nazwa jest o tyle adekwatna, że słowa rzeczywiście wrzucamy do wora, kompletnie nie przejmując się ich kolejnością. Alternatywą, uwzględniającą kolejność słów, jest model N-gramowy, gdzie zamiast pojedynczego słowa bierzemy N słów kolejno występujących po sobie w danym korpusie. W przypadku 2-gramów są to dwa słowa, a w przypadku 3-gramów — pięć. Żartowałem — trzy. Sprawdzam czujność.

Pora na przykład. Poszukajmy czegoś dobrego, bo z całą pewnością będziemy potrzebować porządnego przykładu od czasu do czasu. Proponuję mój ulubiony cytat z Alberta Einsteina:

Zdrowy rozsądek to zbiór uprzedzeń nabytych do osiemnastego roku życia.

A teraz rozpiszmy je na wszystkie możliwe 3-gramy, bez żadnego wymyślnego preprocessingu, usuwając wyłącznie interpunkcję:

1: (Zdrowy, rozsądek, to)

2: (rozsądek, to, zbiór)

3: (to, zbiór, uprzedzeń)

4: (zbiór, uprzedzeń, nabytych)

5: (uprzedzeń, nabytych, do)

6: (nabytych, do, osiemnastego)

7: (do, osiemnastego, roku)

8: (osiemnastego, roku, życia)

Mamy osiem 3-gramów. Każdy taki *N*-gram jest teraz pojedynczym elementem naszego nowego słownika i wektora wejściowego. Zastanówmy się na spokojnie, jakie to będzie miało konsekwencje. W modelu *bag-of-words* jedno słowo to jeden element słownika. W modelu 3-gramowym dla każdego takiego słowa dostaniemy tyle elementów, ile różnych trójek z tym słowem uda się znaleźć w korpusie.

Mam nadzieję, że tym zgrabnym porównaniem udało mi się pokazać, że nasz wektor wejściowy spuchnie do absurdalnych rozmiarów. A im większe *N*, tym większy wektor. Proza życia — „nie ma darmowych obiadów". Ale zyskujemy coś bardzo cennego — wiedzę o zależnościach między słowami.

Jeśli recenzent napisałby „Film nie był dobry", to już model 3-gramowy będzie w stanie wychwycić negację i nie da się na to nabrać. W przeciwieństwie do modelu 2-gramowego i *bag-of-words*, który zresztą można postrzegać jako najprostszy możliwy wariant modelu *N*-gramowego, gdzie *N* to po prostu 1.

Warto nadmienić, że istnieją inne, znacznie ciekawsze i bardziej wysublimowane metody reprezentowania danych. A także inne, znacznie skuteczniejsze metody uwzględniania nie tylko kontekstu, ale i znaczenia. Nie są one jednak na tyle proste, żebyśmy już teraz byli w stanie je omówić. Niech to będzie dla Szanownego Czytelnika zachęta do dalszej lektury.

Uczenie maszynowe to niekończące się eksperymentowanie. To rozwiązywanie pozornie nierozwiązywalnych problemów i przełamywanie coraz to nowych barier. Mam nadzieję, że po lekturze tego rozdziału Szanowny Czytelnik lepiej rozumie, w jaki sposób porządnie wytrenowana sieć neuronowa jest w stanie rozwiązywać niebanalne problemy. Mam nadzieję, bo właśnie wskakujemy do króliczej nory, gdzie w końcu zajmiemy się samym procesem uczenia…

TRENING CZYNI MISTRZA

Sieci neuronowe uczy się z przykładów. Każdy przykład uczący składa się z sygnałów zadawanych na wejściu i przyporządkowanych im oczekiwanych sygnałów wyjściowych. Czyli zadając pytanie, od razu podajemy również prawidłową odpowiedź, żeby z czasem — w ramach treningu — sieć nauczyła się, w jaki sposób może sama znajdować analogiczne odpowiedzi dla nowych, niewidzianych wcześniej pytań. Nazywamy to uczeniem nadzorowanym. Istnieją również nienadzorowane metody uczenia, ale w tym momencie nie będziemy zaprzątać sobie nimi głowy.

W przypadku omawianej wcześniej analizy sentymentu sygnały wejściowe reprezentują treść recenzji, a sygnały wyjściowe — jej sentyment (pozytywny, negatywny lub neutralny). Wielkość zbioru uczącego, potrzebnego do prawidłowego wytrenowania sieci, zależy między innymi od stopnia złożoności rozwiązywanego problemu oraz od tego, jak licznie poszczególne klasy są w tym zbiorze reprezentowane. Analiza sentymentu nie należy do problemów przesadnie skomplikowanych, więc już kilkaset przykładów (recenzji i ich sentymentu) dla każdej z klas powinno dawać przyzwoite rezultaty.

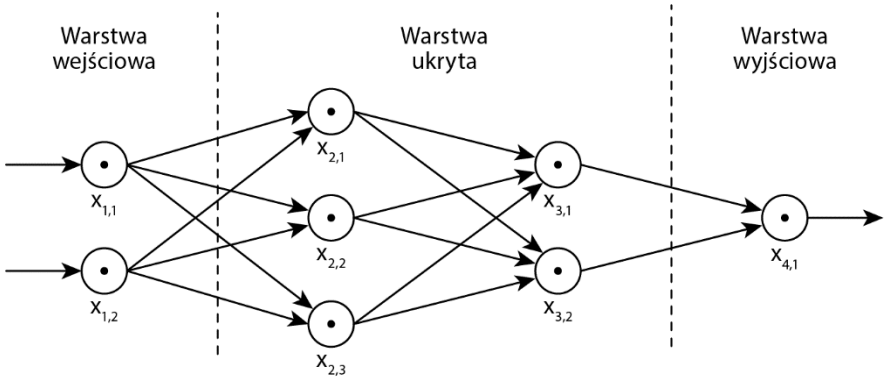
***

A teraz uwaga — to, co wydarzy się za moment, będzie od Szanownego Czytelnika wymagało nieco więcej uwagi i skupienia niż dotychczasowe treści. Nie polecam czytać tego rozdziału „do poduszki". Omówimy bowiem krok po kroku proces uczenia sieci typu *feed-forward*. Wprowadzimy trochę nowych oznaczeń, zapiszemy kilka wzorów i rozrysujemy odpowiadające im schematy. Nic szczególnie trudnego, co wymagałoby wiedzy spoza szkoły średniej. Po prostu świeży i wypoczęty umysł znacznie lepiej poradzi sobie z przyswajaniem tego typu treści.

Zacznijmy od krótkiego podsumowania tego, czego dowiedzieliśmy się do tej pory:

1. Perceptron wielowarstwowy, nazywany również siecią typu *feed-forward*, jest siecią jednokierunkową. Oznacza to, że gdy sieć realizuje powierzone jej zadanie, informacja płynie wyłącznie w jedną stronę — od warstwy wejściowej, przez kolejne warstwy ukryte, aż do warstwy wyjściowej.
2. Neurony z warstwy *N* łączą się ze neuronami z warstwy *N*+1 — każdy z każdym. Neurony z tej samej warstwy nie łączą się ze sobą. Neurony nie łączą się również z neuronami z innych warstw, niebędących ich bezpośrednimi sąsiadami.
3. Początkowe wagi neuronów dobierane są losowo, a następnie podlegają wielokrotnym modyfikacjom w procesie trenowania sieci.
4. Każdy neuron sieci zawiera blok sumujący, który dodaje do siebie ważone wartości sygnałów wejściowych danego neuronu.
5. Każdy neuron sieci zawiera blok aktywacji, który zwraca wartość przyjętej funkcji aktywacji dla sumarycznej wartości sygnału z bloku sumującego.

Załóżmy, że mamy 4-warstwową sieć neuronową. Niech kolejne warstwy sieci zawierają odpowiednio: 2 neurony (warstwa wejściowa), 3 neurony (pierwsza warstwa ukryta), 2 neurony (druga warstwa ukryta) i 1 neuron (warstwa wyjściowa). Taką właśnie sieć przedstawia następujący schemat:



Rysunek 8. Schemat jednokierunkowej sztucznej sieci neuronowej z podwójną warstwą ukrytą (oznaczone neurony)

W przeciwieństwie do poprzednich schematów tym razem nazwiemy wartości wyjściowe, opuszczające poszczególne neurony sieci, korzystając z następującej notacji:

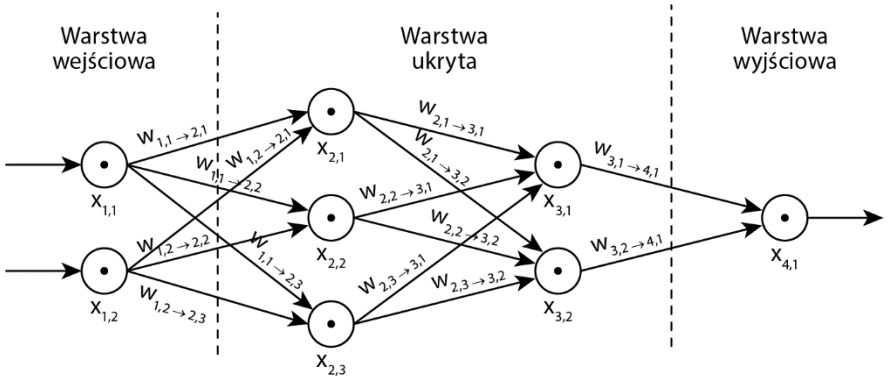
- $X_{N,A}$ — wartość wyjściowa neuronu *A* z warstwy *N*.

Oznacza to, że $X_{2,3}$ odpowiada wartości wyjściowej trzeciego neuronu (licząc od góry) z drugiej warstwy (czyli z pierwszej warstwy ukrytej).

Wagi, które ustalać będziemy w procesie uczenia sieci, występują zawsze na połączeniach między neuronami. Skorzystamy z tego wygodnego faktu, oznaczając je symbolami obu neuronów, które łączą, uwzględniając również kierunek tego połączenia (lub zwrot, jak powiedziałby fizyk):

- $w_{N,A \to M,B}$ — waga połączenia pomiędzy neuronem *A* z warstwy *N* i neuronem *B* z warstwy *M*, przy czym dozwolone są wyłącznie połączenia z warstwą sąsiadującą, więc *M* zawsze będzie równe *N*+1.

Oznacza to, że $w_{2,3 \to 3,1}$ odpowiada wadze pomiędzy neuronem o wartości wyjściowej $X_{2,3}$ (trzeci neuron od góry z drugiej warstwy) a neuronem o wartości wyjściowej $X_{3,1}$ (pierwszy neuron od góry z trzeciej warstwy). Nasz poprzedni schemat uzupełniony o nowe oznaczenia wag prezentuje się następująco:



Rysunek 9. Schemat jednokierunkowej sztucznej sieci neuronowej z podwójną warstwą ukrytą (oznaczone neurony oraz wagi na połączeniach)

Oznaczmy jeszcze funkcję aktywacji danego neuronu w analogiczny sposób:

- $f_{N,A()}$ — funkcja aktywacji neuronu *A* z warstwy *N*, gdzie argumentem funkcji (tym, co jest w nawiasie) jest wynik działania bloku sumującego w tym właśnie neuronie, co z kolei oznacza, że wynik działania funkcji $f_{N,A()}$ to po prostu $X_{N,A}$.

Nie będę Szanownego Czytelnika straszył ogólnym wzorem na wartość wyjściową dowolnego neuronu *A* z warstwy *N*, choć bylibyśmy już w stanie go zapisać. Niech to będzie zadanie dla chętnych. Zamiast tego zapiszmy wzór na wartość wyjściową jakiegoś konkretnego neuronu, powiedzmy $X_{2,3}$:

$$X_{2,3} = f_{2,3}\left(X_{1,1} \times w_{1,1\to 2,3} + X_{1,2} \times w_{1,2\to 2,3}\right)$$

***

Jesteśmy już gotowi, żeby podać do naszej sieci pierwszy przykład uczący:

1. Na wejście sieci, czyli na pierwszą warstwę (wejściową), podajemy sygnały wejściowe z przykładu uczącego — $X_{1,1}$ oraz $X_{1,2}$.
2. Wyznaczamy wszystkie wartości wyjściowe kolejnych neuronów z drugiej warstwy sieci (pierwszej warstwy ukrytej), czyli $X_{2,1}$, $X_{2,2}$ oraz $X_{2,3}$.
3. Wyznaczamy wartości wyjściowe neuronów z trzeciej warstwy sieci (drugiej warstwy ukrytej), czyli $X_{3,1}$ oraz $X_{3,2}$.
4. Wyznaczamy i odczytujemy wartość wyjściową czwartej, ostatniej warstwy sieci (wyjściowej) — $X_{4,1}$.

Wartość $X_{4,1}$ jest jednocześnie wartością wyjściową całej sieci, a to oznacza, że możemy porównać ją z wartością oczekiwaną, wyznaczoną uprzednio przez człowieka dla danego przykładu uczącego. Oznaczmy sobie tę wartość jako $X_{oczekiwana}$. Trafienie w wartość oczekiwaną przy losowych wagach jest bardzo mało prawdopodobne, ale nie jest niemożliwe. W Lotto też zdarzają się przecież główne wygrane, i to przy zawrotnej szansie 1 do 13 983 816. Gdybyśmy jednak trafili taką „szóstkę", to uśmiechamy się pod nosem i idziemy dalej — do kolejnego przykładu uczącego.

Prawdopodobnie jednak nie trafiliśmy. W takim przypadku wyznaczamy różnicę wartości wyjściowej sieci i wartości oczekiwanej z przykładu uczącego. Nazwijmy ją sygnałem błędu neuronu i oznaczmy jako:

- $B_{N,A}$ — sygnał błędu neuronu *A* z warstwy *N*.

Sygnał błędu neuronu warstwy wyjściowej, zgodnie z nową notacją, oznaczymy jako $B_{4,1}$. Wyznaczamy go, korzystając z następującego wzoru:

$$B_{4,1} = \frac{1}{2}\left(X_{4,1} - X_{oczekiwana}\right)^2$$

Nie wygląda to strasznie, prawda? I dalej poszłoby już z górki, gdyby nie jeden drobny szkopuł — nie znamy żadnych innych wartości oczekiwanych. Z przykładu uczącego wiemy jedynie, co chcielibyśmy dostać na wyjściu z całej sieci. Nie mamy bladego pojęcia, co powinno być na trzecim neuronie z pierwszej warstwy ukrytej ani na pierwszym neuronie z drugiej warstwy ukrytej.

Nie wszystko jednak stracone. Mamy punkt wyjścia, a to już coś. Z pomocą przychodzi nam algorytm wstecznej propagacji błędów, z angielskiego *backpropagation*. Niech Szanowny Czytelnik zapamięta tę nazwę. To bardzo szczwany i przełomowy pomysł, który jeszcze w latach 80. ubiegłego wieku skierował całe to głębokie uczenie maszynowe na właściwe tory, po których — z niemałymi zresztą sukcesami — jedzie aż do dziś. Cóż nam bowiem po najpiękniejszej nawet architekturze sieci, jeśli nie da się jej niczego nauczyć?

Algorytm wstecznej propagacji błędów polega na tym, że wyznaczony sygnał błędu neuronu warstwy wyjściowej, czyli nasz $B_{4,1}$, propaguje się wstecz na kolejne neurony sieci. Propagacja następuje w kierunku przeciwnym do przepływu informacji w sieci, czyli od prawej do lewej — od ostatniej warstwy ukrytej aż do pierwszej. Co więcej, algorytm korzysta dokładnie z tych samych wag, których używaliśmy wcześniej do mnożenia sygnałów wejściowych.

No dobrze, to lecimy z tym koksem. Mamy wyznaczony sygnał błędu $B_{4,1}$ z warstwy czwartej i chcemy propagować go wstecz, do warstwy trzeciej, gdzie już czekają na nas dwa kolejne neurony — $X_{3,1}$ oraz $X_{3,2}$. Dla przypomnienia: wartość wyjściowa naszego jedynego neuronu z warstwy wyjściowej sieci byłaby wyznaczana (gdybyśmy naprawdę to robili) przy użyciu tego wzoru:

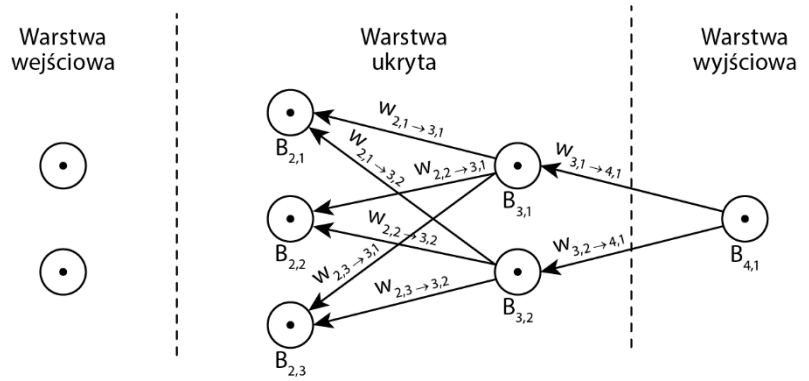
$$X_{4,1} = f_{4,1}\left(X_{3,1} \times w_{3,1\to 4,1} + X_{3,2} \times w_{3,2\to 4,1}\right)$$

Używając dokładnie tych samych wag, możemy teraz wyznaczyć sygnały błędów neuronów z warstwy trzeciej. Skorzystamy w tym celu z następujących wzorów:

$$B_{3,1} = B_{4,1} \times w_{3,1\to 4,1}$$

$$B_{3,2} = B_{4,1} \times w_{3,2\to 4,1}$$

Widzimy już chyba, co tu się tak naprawdę dzieje. Robimy dokładnie to samo, co robiliśmy wcześniej, wyznaczając wartości wyjściowe neuronów przy normalnym działaniu sieci. Są dwie różnice — tym razem idziemy od prawej do lewej zamiast od lewej do prawej, oraz pomijamy działanie funkcji aktywacji. Schemat naszej sieci przedstawia teraz kierunek wstecznej propagacji, z zaznaczonymi sygnałami błędów oraz wagami wykorzystywanymi do ich wyznaczenia:



Rysunek 10. Schemat działania algorytmu wstecznej propagacji błędów dla jednokierunkowej sztucznej sieci neuronowej z podwójną warstwą ukrytą

W kolejnym kroku będzie tylko odrobinę trudniej. Przechodząc z warstwy trzeciej do drugiej, musimy propagować sygnały błędów z dwóch neuronów naraz, a nie jak dotąd z jednego. Ale przecież wcześniej, idąc zgodnie z kierunkiem działania sieci, robiliśmy coś bardzo podobnego — wyznaczając wartości wyjściowe neuronów z warstwy trzeciej, sumowaliśmy ważone wartości wyjściowe wszystkich neuronów z warstwy drugiej. A zatem dla neuronów z warstwy drugiej (pierwszej ukrytej)

możemy w analogiczny sposób wyznaczyć poszczególne sygnały błędów, sumując ważone sygnały błędów wszystkich neuronów z warstwy trzeciej (drugiej ukrytej):

$$B_{2,1} = B_{3,1} \times w_{2,1\to3,1} + B_{3,2} \times w_{2,1\to3,2}$$
$$B_{2,2} = B_{3,1} \times w_{2,2\to3,1} + B_{3,2} \times w_{2,2\to3,2}$$
$$B_{2,3} = B_{3,1} \times w_{2,3\to3,1} + B_{3,2} \times w_{2,3\to3,2}$$

I to już wszystkie sygnały błędów, które musieliśmy wyznaczyć dla pojedynczego przykładu uczącego. Wypadałoby jeszcze wytłumaczyć, dlaczego nie liczymy ich dla warstwy wejściowej, prawda? Ano dlatego, że sygnały błędów służą nam do modyfikacji wag. Przypominam, że właśnie o to toczy się gra. Wagi zawsze „łączą" dwa neurony z sąsiednich warstw, ale sposób ich modyfikacji zależy wyłącznie od sygnału błędu neuronu docelowego. A zatem chcąc zmodyfikować wagę $w_{1,1\to2,1}$, wystarczy znać $B_{2,1}$.

Na koniec powinniśmy jeszcze zapisać jakiś elegancki wzór na modyfikację tych wag, ale tego już Szanownemu Czytelnikowi oszczędzę. Nie dlatego, że jest to nie wiadomo jak skomplikowane, ale dlatego, że nie da się tego zrobić bez liczenia pochodnych. A ludzie nie lubią i nie chcą kupować książek z pochodnymi, więc robię to dla swojego własnego dobra.

Coś jednak napisać wypada, prawda? Nowa waga to stara waga powiększona lub pomniejszona o pewną wartość, będącą iloczynem pochodnej funkcji aktywacji, sygnału błędu neuronu docelowego oraz wartości wyjściowej neuronu, z którego aktualnie modyfikowana waga „wychodzi". Bardzo proszę. Napisałem to raz i już nigdy więcej nie musimy o tym rozmawiać. Jak to mówią: co dzieje się w sieci neuronowej, zostaje w sieci neuronowej.

Gdy już zmodyfikujemy wszystkie wagi, bierzemy na warsztat kolejny przykład uczący. I kolejny. I kolejny. Uczenie perceptronu wielowarstwowego polega więc na iteracyjnym wykonywaniu powyższego algorytmu wstecznej propagacji błędów dla kolejnych przykładów ze zbioru uczącego. Możemy to podsumować następującym „przepisem kulinarnym":

1. Weź (kolejny) przykład uczący ze zbioru treningowego.

2. Podaj na wejście sieci sygnały wejściowe z danego przykładu uczącego.

3. Wylicz wartości sygnałów wyjściowych dla neuronów w kolejnych warstwach sieci, idąc od pierwszej warstwy ukrytej, aż do warstwy wyjściowej.
4. Wyznacz sygnały błędów dla neuronów z warstwy wyjściowej, porównując wartości sygnałów wyjściowych sieci z odpowiadającymi im wartościami oczekiwanymi z danego przykładu uczącego.
5. Wyznacz sygnały błędów dla neuronów w kolejnych warstwach sieci, idąc od ostatniej warstwy ukrytej, aż do pierwszej.
6. Zmodyfikuj wagi połączeń sieci w oparciu o wyznaczone sygnały błędów.
7. Powtórz.

Oczywiście nikt tego nie liczy samodzielnie. Spokojnie. Wszystkie te żmudne, ale powtarzalne czynności, które da się zapisać w postaci algorytmu, czyli właśnie takiego „przepisu kulinarnego", są przez mądrych ludzi implementowane, optymalizowane i dostarczane innym mądrym ludziom w postaci łatwych do użycia bibliotek. I chwała im za to. Nie ma nic lepszego niż dobrze ukierunkowane lenistwo.

Dzięki temu inżynier uczenia maszynowego jest w stanie tworzyć przydatne i skuteczne rozwiązania bez szczegółowej znajomości wszystkich niuansów związanych z architekturą sieci czy algorytmem jej uczenia. Podobnie jak programista dziś już w zasadzie nie musi nawet wiedzieć, jak działa komputer. Można dyskutować czy to dobrze, czy źle. Osobiście jestem fanem wszechstronnej wiedzy, bo pozwala wyjść poza utarte schematy i spojrzeć na dany problem z kilku różnych perspektyw. Ale też trudno odmówić słuszności stwierdzeniu, że „codzienna" praca idzie dużo sprawniej, gdy jedna osoba wytwarza młotek, druga gwoździe, a jeszcze inna korzysta z nich, zbijając ze sobą deski.

Jeśli Szanowny Czytelnik dotarł do tego miejsca, to należą się szczere wyrazy uznania. Serio — zdaję sobie sprawę, że nie są to treści łatwe i przystępne dla osób spoza branży. A jeśli komuś się to podobało… no cóż, zalecam konsultację ze specjalistą. Żartuję, bardzo się cieszę. Moją intencją było odarcie sieci neuronowych z magii. Ale nie po to, by je zdyskredytować lub zohydzić, tylko po to, żebyśmy mogli spojrzeć na nie takie, jakimi są naprawdę, bez upiększania, pudrowania i malowania trawy na zielono.

KLĄTWA CZARNEJ SKRZYNKI

Dokonało się — oto zerwaliśmy owoc z drzewa poznania sieci neuronowych. Posiedliśmy wiedzę, która pozwoli nam wreszcie odpowiedzieć na pytanie z początku książki. To, które zadał młody Symbolista, a o którym większość Czytelników zdążyła już zapewne zapomnieć: dlaczego nikt nie potrafił wyjaśnić działań robota Koneksjonistów, który twierdził, że na zdjęciu jest kot, pomimo że go tam nie było?

Teraz już wiemy, że robot Koneksjonistów, czyli wyuczona sieć neuronowa, to nic innego jak zbiór bardzo wielu połączonych ze sobą neuronów, umieszczonych na kolejnych warstwach sieci, którym dobrano odpowiednie wagi w procesie jej uczenia. Ale czy umielibyśmy powiedzieć, dlaczego na wyjściu dostajemy taką, a nie inną wartość?

Moglibyśmy spróbować prześledzić wszystkie połączenia między neuronami i ich wagi, żeby sprawdzić, które sygnały wejściowe są wzmacniane, a które wygaszane. I być może dla bardzo prostych przypadków w drodze dedukcji udałoby się ustalić, które z nich miały największy wpływ na wyjście sieci. Ale im głębsza sieć, tym bardziej rozmywają się te wszelkie zależności i tym trudniej o jakiekolwiek sensowne wnioski. Spróbuję to wyjaśnić na przykładzie.

Wyobraźmy sobie, że działanie sieci neuronowej jest jak układanie puzzli:

1. Warstwa wejściowa reprezentuje puzzle wysypane z pudełka na podłogę.
2. W pierwszej warstwie ukrytej odwracamy wszystkie elementy obrazkiem do góry.
3. W drugiej szukamy brzegów i układamy „ramkę".
4. W trzeciej układamy kształty na zasadzie kontrastu, np. dachy na tle nieba.

5. W czwartej łączymy te kształty w obiekty, np. ściany, okna i dachy w budynki.
6. W piątej łączymy ze sobą całe obiekty, np. budynki w nadmorskie miasteczko.
7. W szóstej wypełniamy to, co pozostało niewypełnione — niebo, wodę czy trawę.
8. W warstwie wyjściowej mamy już kompletny obraz i możemy w końcu powiedzieć, co się na nim znajduje.

Nie, wcześniej nie mogliśmy, bo pudełko z obrazkiem zjadł nam pies. Proszę się nie czepiać. I tak — zdaję sobie sprawę, że istnieją inne strategie układania puzzli. Wiem też, że wcale nie musieliśmy układać całości, żeby móc powiedzieć, co znajduje się na obrazku. Nie psujmy sobie jednak tak pięknej analogii jakimiś błahostkami.

Każda kolejna warstwa sieci to kolejny poziom abstrakcji — pojedyncze elementy, kształty, obiekty, grupy obiektów. Problem polega na tym, że kolejne warstwy ukryte rzeczywiście mogą reprezentować te nasze wyobrażenia, ale wcale nie muszą. Mogą reprezentować coś zupełnie innego. Coś, o czym nigdy byśmy nie pomyśleli jako przedstawiciele *homo sapiens* — gatunku, którego mózg, wbrew pozorom, wcale nie służy do myślenia. Tylko czy to aby na pewno jest problem?

Nikt przy zdrowych zmysłach nie powinien twierdzić, że nasz sposób postrzegania rzeczywistości jest jedynym możliwym albo obiektywnie lepszym od innych. Jeśli nie będziemy się upierać, że krzesło tym się różni od stołu, że krzesło ma oparcie, a stół nie, to nigdy nie natrafimy na problem w postaci taboretu. I bardzo dobrze. Wystarczy, że na co dzień bohatersko walczymy z całą masą problemów, które sami stworzyliśmy.

Dlatego fakt, że nie musimy tłumaczyć maszynie, jak wygląda kot ani czym różni się od psa, uważam za ogromną zaletę. Sieć sama nauczy się tego z pokazywanych jej przykładów. Nie potrzebujemy dyplomu z anatomii zwierząt domowych. Nie potrzebujemy głębokiej wiedzy domenowej. Ba, w wielu dziedzinach nawet jej nie mamy. Albo, co gorsza, nasze przyzwyczajenia lub wyobrażenia na dany temat są wręcz szkodliwe.

W latach 2004 – 2012 amerykańska stacja telewizyjna Fox emitowała bardzo popularny, również w Polsce, serial o lekarzach — *Dr House*. Większość odcinków do znudzenia eksploatowała następujący schemat — pacjent choruje na pewną chorobę A, ale symptomy i pierwsze wyniki badań wskazują na inną, bardziej prawdopodobną — przynajmniej zdaniem większości lekarzy — chorobę B. Pacjent zaczyna być leczony zgodnie z tą drugą diagnozą i wkrótce jego stan się pogarsza.

I tu wchodzi on — tytułowy doktor House. Cały na biało, mimo że akurat jako jedyny lekarz w serialu nie nosi fartucha. Łączy pozornie nieistotne fakty — bardzo rzadki objaw choroby lub skutek uboczny działania leku ze specyficzną wiedzą o pacjencie, zazwyczaj pozyskaną z… włamania do jego domu — i stawia właściwą diagnozę. Pacjent zaczyna być leczony na A, dochodzi do siebie i odjeżdża w stronę zachodzącego słońca. Napisy końcowe.

Ale po co ja o tym wszystkim piszę? Bo wszyscy kłamią? Nie, nie tym razem. Otóż nasza sieć neuronowa, właśnie dlatego, że nie obarczamy jej naszymi przypuszczeniami czy uprzedzeniami, może być takim doktorem House'em na sterydach. Może dostrzec rzadkie zależności, których młody lekarz nie zauważy, a lekarz z wieloletnim doświadczeniem zignoruje jako mało prawdopodobne. Albo analizując setki tysięcy przypadków, znajdzie zależności, których żaden lekarz nigdy wcześniej zwyczajnie nie mógł dostrzec ze względu na skalę. Do najciekawszych zastosowań sztucznej inteligencji jeszcze wrócimy, ale teraz porozmawiajmy w końcu o klątwie czarnej skrzynki, bo niestety każdy kij ma dwa końce.

***

Czarna skrzynka to termin określający system lub urządzenie o znanej funkcjonalności, ale nieznanym mechanizmie działania. Znamy wejście, znamy wyjście, ale nie umiemy powiedzieć, co dokładnie dzieje się w środku. Wyuczona sieć neuronowa jest właśnie taką czarną skrzynką.

Zgoda — mamy warstwy i neurony, wiemy, ile ich jest, możemy nawet podejrzeć wagi na poszczególnych połączeniach. Ale nie mamy pojęcia, co reprezentują kolejne warstwy ukryte sieci. Nie umiemy powiedzieć, dlaczego robot Konekcjonistów stwierdził, że na zdjęciu jest kot, pomimo że go tam nie było. Możemy

zgadywać, że model pomylił kształt kota z kształtem poduszki, ale tak po prawdzie nie wiemy nawet, czy on w ogóle szukał jakichkolwiek kształtów.

To dość paradoksalne, bo wiemy, że sztuczne neurony i tworzone z nich sieci są wzorowane na ich biologicznych odpowiednikach. Być może stąd nasza przemożna chęć do antropomorfizacji, do doszukiwania się w nich naszych ludzkich zachowań i ludzkiego sposobu postrzegania rzeczywistości. To błąd, który może nas wpuścić w maliny. Lepiej jest wiedzieć, że czegoś nie wiemy, i zaakceptować ten fakt, niż wyciągać daleko idące wnioski na podstawie naszych wątpliwych przypuszczeń.

Zastanówmy się zatem — co to dla nas oznacza? Czy to rzeczywiście problem, że nie wiemy, co dzieje się w środku czarnej skrzynki? Przy tak postawionym pytaniu na usta ciśnie się najbardziej naukowo-inżynierska odpowiedź, jaką na przestrzeni wieków podawały najtęższe umysły tego świata, czyli: to zależy.

A od czego, jeśli można wiedzieć? Przede wszystkim od zastosowania. Nikt nie będzie rwał sobie włosów z głowy, jeśli nie dowiemy się, dlaczego model pomylił kota z poduszką. Albo dlaczego błędnie ocenił negatywną recenzję jako pozytywną. Zwłaszcza że w przypadku takich zastosowań jak analiza sentymentu istotna jest statystka, a nie pojedyncze wpadki modelu. Zatrzymajmy się tu na moment i porozmawiajmy o tym, w jaki sposób testuje się modele uczenia maszynowego.

***

Z poprzednich rozdziałów wiemy, że sieci neuronowe uczy się, „pokazując" im kolejne przykłady ze zbioru uczącego. Testuje się bardzo podobnie — „pokazując" analogiczne przykłady ze zbioru testowego. Oba zbiory powinny być rozłączne, co oznacza, że model testuje się wyłącznie na przykładach, które nie były wcześniej wykorzystywane do uczenia sieci.

Zazwyczaj przygotowuje się jeden duży zbiór danych, który następnie dzieli się na zbiór uczący i testowy w ustalonych proporcjach, np. 70% przypadków trafia do zbioru uczącego, a 30% do testowego. Na podstawie przypadków testowych sprawdza się poprawność predykcji modelu, czyli zgodność wartości wyjściowej z wartością oczekiwaną.

W przypadku problemów klasyfikacyjnych wyróżniamy 4 typy wyników, które możemy otrzymać, oceniając pojedynczy przypadek testowy. Znowu muszę poprosić o chwilę uwagi i skupienia. Za moment wprowadzimy kilka nowych oznaczeń, wzorów i dziwacznych określeń, ale Szanowny Czytelnik przekona się, że to wszystko ma sens.

Dlaczego aż 4 typy wyników? Dlatego, że mamy ocenę modelu, którą sprawdzamy, i ocenę człowieka, którą traktujemy jako poprawną. Każdy przypadek może należeć (wynik dodatni) lub nie należeć (wynik ujemy) do danej kategorii. I każdą z tych dwóch możliwości przyrównujemy do oceny człowieka, będącej jedynym źródłem prawdy. Jeśli ocena modelu zgadza się z oceną człowieka, to mamy wynik prawdziwy, a jeśli się nie zgadza — fałszywy.

Stąd oceniając poprawność klasyfikacji, możemy otrzymać 1 z 4 wyników:

- wynik prawdziwie dodatni (*TP* od angielskiego *true positive*), kiedy przypadek testowy należy do danej kategorii i model prawidłowo go do niej zaklasyfikował;
- wynik prawdziwie ujemy (*TN* od angielskiego *true negative*), kiedy przypadek testowy NIE należy do danej kategorii i model prawidłowo go do niej NIE zaklasyfikował;
- wynik fałszywie pozytywny (*FP* od angielskiego *false positive*), nazywany również błędem I typu, kiedy przypadek testowy NIE należy do danej kategorii, ale model go do niej błędnie zaklasyfikował;
- wynik fałszywie negatywny (*FN* od angielskiego *false negative*), nazywany również błędem II typu, kiedy przypadek testowy należy do danej kategorii, ale model błędnie go do niej NIE zaklasyfikował.

Poniższa tabelka przedstawia wszystkie możliwe wyniki dla prostego modelu oceniającego, czy na zdjęciu jest kot, czy też go tam nie ma.

Tabela 1. Wszystkie możliwe do uzyskania wyniki oceny modelu rozpoznającego na zdjęciu kota lub jego brak

OCENA CZŁOWIEKA	OCENA MODELU	WYNIK
kot	*kot*	*TP — prawdziwie dodatni*
nie kot	*nie kot*	*TN — prawdziwie ujemny*
nie kot	*kot*	*FP — fałszywie dodatni*
kot	*nie kot*	*FN — fałszywie ujemny*

Jeśli teraz dla całego zbioru testowego podliczymy poszczególne typy otrzymywanych wyników, to będziemy mogli policzyć dwie nowe i piekielnie użyteczne metryki — precyzję (od angielskiego *precision*) oraz kompletność (od angielskiego *recall*). Ze względu na ich charakter lubimy przedstawiać je procentowo, ale może po kolei…

W prostych żołnierskich słowach: jeśli mamy koszyk i chcemy nazbierać do niego jabłek, to kompletność określa, ile spośród wszystkich jabłek udało nam się zebrać, a precyzja pozwala określić, ile gruszek i śliwek przez przypadek trafiło do naszego koszyka z jabłkami.

Kompletność mówi nam, ile ze wszystkich przypadków, które powinny zostać zaklasyfikowane do danej kategorii, rzeczywiście do niej trafiło. Jeśli do zebrania było 10 jabłek, a my znaleźliśmy tylko 8 z nich, to kompletność wynosić będzie 80%.

Precyzja natomiast określa, ile przypadków, które zaklasyfikowaliśmy do danej kategorii, słusznie do niej trafiło. Jeśli do koszyka trafiło 10 owoców, a tylko 8 z nich to jabłka, to precyzja także wynosić będzie 80%.

Do wyznaczenia precyzji i kompletności wystarczy nam suma poszczególnych typów wyników w zbiorze testowym. Niech *TP* oznacza liczbę wyników prawdziwie dodatnich, *FP* — liczbę wyników fałszywie dodatnich, a *FN* — liczbę wyników fałszywie ujemnych. Wtedy możemy zapisać następujące wzory:

- $precyzja = \frac{TP}{TP+FP}$
- $kompletność = \frac{TP}{TP+FN}$

A skoro tak dobrze nam idzie, to zapiszmy od razu wzór na średnią harmoniczną F_1, łączącą obie te metryki:

- $F_1 = 2 \times \frac{precyzja \times kompletność}{precyzja + kompletność}$

Po co nam ona? — zapytacie. I słusznie, bo naprawdę dobry inżynier zamiast dodawać kolejne byty, powinien odejmować je tak długo, aż nie pozostanie już nic więcej do odjęcia. Ta średnia odegra jednak dość ważną rolę, bo posiada tę interesującą cechę, że przyjmuje najwyższe wartości wtedy, kiedy obie metryki są sobie równe, a w przypadku rosnących dysproporcji gwałtownie spada.

Policzmy:

- 50% precyzji i 50% kompletności daje $F_1 = 50\%$,
- 70% precyzji i 30% kompletności daje $F_1 = 42\%$,
- 90% precyzji i 10% kompletności daje $F_1 = 18\%$.

I z tej przyczyny to właśnie pod średnią harmoniczną precyzji i kompletności najczęściej optymalizuje się modele uczenia maszynowego. Stosuje się do tego trzeci zbiór danych, tzw. zbiór walidacyjny, który znów trzeba „wyszarpać" ze zbioru uczącego i testowego, choćby w jednej z popularnych proporcji — 80/10/10, 70/15/15 lub 60/20/20, gdzie oczywiście dominującym składnikiem jest zbiór uczący.

***

Nie wystarczy zatem, że nauczymy sieć, jak rozwiązywać dany problem. Musimy ją jeszcze porządnie przetestować, żeby wiedzieć, na ile wiarygodne są jej predykcje. Dopiero wtedy będziemy w stanie ocenić, czy i w jaki sposób możemy wykorzystać nasze rozwiązanie, godząc się przy tym, że nie będziemy wiedzieli, co dzieje się w jego wnętrzu i w jaki dokładnie sposób podejmowana jest decyzja.

Jeśli do analizy sentymentu zastosujemy model uzyskujący w testach 90% precyzji i 70% kompletności, to możemy założyć, że trend prezentowany przez ten model będzie wiarygodny, i możemy kierować się nim przy podejmowaniu naszych decyzji biznesowych. Jeśli stosunek recenzji pozytywnych do negatywnych rośnie w czasie, to nie ma powodu do zmartwień. Jeśli spada — warto zainteresować się tematem.

A jeśli nagle zalewa nas fala negatywnych recenzji, to trzeba czym prędzej bić na alarm, bo najprawdopodobniej coś poszło nie tak i wymagana jest natychmiastowa reakcja.

Jak można się domyślać, istnieją również zastosowania, dla których wyjaśnialność modelu jest istotna lub wręcz kluczowa. Jeśli działamy w oparciu o trend lub statystkę, to — jak w powyższym przypadku — możemy po prostu zamknąć oczy i zaufać modelowi. Jeśli jednak nasze działania zależą od wyniku pojedynczej predykcji, to wolelibyśmy rozumieć, skąd taka, a nie inna decyzja, a nie polegać wyłącznie na ogólnej, choćby nie wiem jak wysokiej i dokładnie zmierzonej skuteczności modelu. A im większa waga tej decyzji i podejmowanego w oparciu o nią działania, tym większe zapotrzebowanie na wyjaśnialność modelu. W skrajnych przypadkach może przecież chodzić nawet o ludzkie życie.

W naturalny sposób przychodzą tu na myśl zastosowania sztucznej inteligencji w medycynie. Jeśli nasz model zdiagnozował u pacjenta nowotwór, to chcemy zrozumieć, skąd taka diagnoza, zanim zaczniemy stosować agresywne metody leczenia lub interwencję chirurgiczną. Oczywiście w przypadku modeli diagnostycznych mamy lekarza, który może poszukać wyjaśnienia poza modelem. A co, jeśli nie ma na to czasu i decyzję trzeba podjąć natychmiast, wyłącznie w oparciu o predykcję modelu?

Że niby trudno wyobrazić sobie taką sytuację? A pojazdy autonomiczne? Samochody samosterujące? Prace nad nimi trwają już od wielu lat, a od co najmniej kilku są z powodzeniem testowane w Stanach Zjednoczonych, w Kalifornii i Arizonie, oraz w Chinach, na ulicach liczącego ponad 13 milionów mieszkańców miasta Shenzhen. Ich decyzje muszą być podejmowane autonomicznie, w czasie rzeczywistym, w oparciu o sygnały spływające z różnego rodzaju czujników. Taki pojazd musi w ułamku sekundy ocenić czy to, co nagle pojawiło się na jezdni, to pieszy, czy może targany wiatrem worek na śmieci. Ocenić i zareagować, nie stwarzając przy tym niebezpieczeństwa dla pozostałych uczestników ruchu drogowego.

Tyle że to znowu przypadek skrajny, bo — ze względu na wymagany czas reakcji — ewentualna wyjaśnialność modelu raczej nie zapobiegłaby nieszczęściu, a jedynie mogłaby pomóc w ustaleniu, dlaczego do nieszczęścia w ogóle doszło. No więc kiedy cała ta wyjaśnialność ma największe znaczenie? Wtedy, kiedy okienko

czasowe na podjęcie decyzji jest na tyle duże, że pozwala na uwzględnienie wyjaśnień predykcji, ale na tyle małe, że nie pozwala na przeprowadzenie niezależnej analizy poza modelem. Fani filmów akcji i dreszczowców bez pudła wskażą tu przeróżne zastosowania militarne, gdzie przesadna zwłoka w podejmowaniu decyzji może prowadzić do bardzo poważnych konsekwencji.

Wyjaśnialność modelu ma również znaczenie wtedy, gdy decyzję musimy podjąć natychmiast, ale później musimy się z niej jeszcze wytłumaczyć. Tu z przyjemnością podzielę się przykładami z mojej pracy zawodowej, z firmy Samurai Labs, której jestem dumnym współzałożycielem i w której — jako dyrektor ds. technologii — odpowiadam między innymi za tematy związane ze sztuczną inteligencją. Jednym z rozwiązań oferowanych przez Samurai Labs jest proaktywna ochrona społeczności internetowych przed całą masą niebezpiecznych zjawisk i zachowań, szeroko nazywanych cyberprzemocą.

Chyba każdy z nas korzystał kiedyś z jakiegoś komunikatora internetowego, czatu, forum albo wymieniał opinie w komentarzach czy to pod filmem na YouTubie, czy też w mediach społecznościowych. Gdy pojawia się gagatek, który w rozmowie obraża lub nęka innych i nie szanuje zasad panujących w danej społeczności, to szybka reakcja pozwala uchronić potencjalne ofiary i zapobiec eskalacji przemocy. Dlatego zadaniem naszej sztucznej inteligencji, pełniącej wtedy rolę autonomicznego moderatora, jest reagować natychmiast, proaktywnie, zanim komukolwiek stanie się krzywda.

Ale ponieważ funkcjonujemy również w otwartych społecznościach internetowych, gdzie wolność słowa jest cenioną wartością, to ów gagatek ma prawo wiedzieć, za co dokładnie został czasowo wyciszony lub zbanowany albo dlaczego jego wypowiedź została zablokowana lub usunięta. Jeśli jesteśmy w stanie podjąć natychmiastową decyzję i skutecznie wytłumaczyć nasze przesłanki, to takie podejście ma dodatkowe walory edukacyjne, deeskalacyjne i pozwala na utrzymanie zdrowych standardów komunikacji — bez cyberprzemocy, ale i bez cenzury.

Innym bardzo ważnym i niezwykle trudnym zagadnieniem, którym zajmujemy się w Samurai Labs, jest pomoc osobom w kryzysie presuicydalnym i suicydalnym, czyli osobom rozważającym lub planującym odebranie sobie życia. Tylko w 2023 roku nasza sztuczna inteligencja pozwoliła na udzielenie wsparcia ponad 25 tysiącom

takich osób na całym świecie. Wystarczy, że osoba w kryzysie da temu wyraz, opisując, co czuje, myśli lub przeżywa, na platformie internetowej, którą monitorujemy. Możemy wtedy wykryć zagrożenie i zareagować, wysyłając specjalną interwencję, wybrane przez ekspertów materiały samopomocowe oraz namiary na sprawdzone organizacje, które oferują profesjonalną, bezpłatną i anonimową pomoc.

I tu również, oprócz szybkiej reakcji, znaczenie ma wyjaśnialność predykcji, która pozwala lepiej ocenić stan osoby w kryzysie, a przez to skuteczniej dobrać metody dotarcia do niej i formy oferowanej pomocy. Istotnym predyktorem, czyli czynnikiem objaśniającym, jest ocena, czy mamy do czynienia z ideacją, zamiarem, czy planem. Osoba w kryzysie może dzielić się swoimi myślami samobójczymi, może pisać o zamiarze, gotowości i podjętej decyzji, a może przedstawiać szczegółowy opis uwzględniający czas, miejsce i sposób przeprowadzenia planowanej próby samobójczej. Rozróżnienie tych sytuacji jest niezwykle ważne. Do innych predyktorów możemy zaliczyć opis traumatycznych przeżyć, poprzednich prób samobójczych czy też akt pożegnania się z rodziną, przyjaciółmi lub innymi członkami danej społeczności.

Chyba wypada mi w tym momencie przeprosić Szanownego Czytelnika za tę niespodziewaną bombę. Zdaję sobie sprawę, że atmosfera zrobiła się już tak ciężka i gęsta, że można by ją kroić nożem, ale zależało mi na tym, żeby pokazać, że sztuczna inteligencja to nie tylko lepiej dobrane reklamy, ale także ratowanie ludzkiego życia. Tak na serio, czasem w ostatniej chwili. I tu nie tylko szybkość reakcji, ale również wyjaśnialność procesu podejmowania decyzji może mieć bardzo istotne znaczenie.

***

Chcemy mieć wyjaśnialne modele, bez dwóch zdań. Ale sieci neuronowe nie bardzo chcą z nami w tym aspekcie współpracować. Mimo to próbujemy, a tzw. wyjaśnialna sztuczna inteligencja (XAI, od angielskiego *eXplainable AI*) jest istotnym kierunkiem prac badawczych wielu firm i jednostek naukowych. Dlatego też sytuacja jest zła, ale nie beznadziejna.

Z pomocą przychodzą nam narzędzia, które wspierają nie tyle wyjaśnialność, co raczej interpretowalność modeli uczenia maszynowego. Jedne pozwalają przypisywać poszczególnym predykcjom tzw. stopień pewności (ang. *confidence score*), wskazujący, jak bardzo konkretne cechy wpływają na ostateczny wynik. Inne pomagają wykrywać negatywne cechy modeli, takie jak stronniczość (ang. *bias*) czy dryf danych (ang. *data drift*).

Stronniczość polega na tym, że wyuczony model wykazuje pewne systematyczne uprzedzenia w stosunku do określonej części analizowanych przypadków. Najczęściej dzieje się tak w przypadku źle dobranych lub źle zbalansowanych zbiorów uczących. Przykład: od wielu lat w amerykańskich szpitalach funkcjonuje system sztucznej inteligencji przewidujący, którzy pacjenci mogą wymagać dodatkowej opieki medycznej.

W 2019 roku wykryto, że algorytm wyraźnie dyskryminował pacjentów ciemnoskórych. Działo się tak dlatego, że model bazował na założeniu, że zapotrzebowanie na opiekę medyczną jest bezpośrednio skorelowane z indywidualnymi wydatkami na ochronę zdrowia. W związku z tym do uczenia modelu wykorzystano wcześniejsze wydatki pacjentów, a te okazały się wyraźnie niższe w przypadku osób ciemnoskórych. Podobną stronniczością wykazywały się algorytmy wspomagające rekrutację w dużych przedsiębiorstwach (dyskryminujące kobiety) lub oceniające zdolność kredytową potencjalnych klientów banków (dyskryminujące osoby ciemnoskóre).

Dryf danych jest związany z naturalną zmiennością świata, w którym przyszło nam żyć. To, że jakiś model wyuczony na danych z 2023 roku działa poprawnie dla danych z 2023 roku, nie oznacza, że będzie działał poprawnie dla analogicznych danych z 2024 lub 2025 roku. Świat nieustannie się zmienia i nasz model może osiągać gorsze wyniki dla nowych danych. Przykład: modele do rozpoznawania twarzy działały bardzo dobrze do 2020 roku, ale w związku z pandemią COVID-19 i obowiązkiem noszenia maseczek nagle przestały. Dlaczego? Bo w danych uczących nie było zdjęć twarzy w maseczkach, które od 2020 roku stanowiły absolutną większość danych wejściowych rejestrowanych w miejscach publicznych. Oczywiście taka „degradacja" modeli wcale nie musi być gwałtowna i skokowa. Może być powolna i stopniowa, np. wskutek długotrwałych zmian preferencji zakupowych konsumentów.

Warto dodać, że bardzo często metody stosowane w celu poprawy wyjaśnialności lub interpretowalności modeli uczenia maszynowego jednocześnie obniżają skuteczność samych modeli. W skrócie: im lepsza wyjaśnialność, tym niższa skuteczność. To trochę jak z zasadą nieoznaczoności Heisenberga, mówiącą, że istnieją pary wielkości, których nie da się zmierzyć z dowolną dokładnością. Im dokładniej znamy położenie cząstki, tym mniej wiemy o jej pędzie. A dla tych, którzy reagują alergicznie na wszelkie odniesienia do mechaniki kwantowej, mamy najlepsze możliwe podsumowanie tego zjawiska, zaserwowane nam przez brytyjskiego pisarza Alana Alexandra Milne'a w *Chatce Puchatka*: „Im bardziej Puchatek zaglądał do środka, tym bardziej Prosiaczka tam nie było"…

JAK HAKOWAĆ MODELE UCZENIA MASZYNOWEGO?

Najciekawsza konsekwencja „klątwy czarnej skrzynki" zasługuje moim skromnym zdaniem na osobny rozdział. Ale zanim zaczniemy, chciałbym wytłumaczyć się z tego nieco „clickbaitowego" tytułu. Moją intencją nie jest namawiać Szanownego Czytelnika do jakichkolwiek nieetycznych lub nielegalnych działań, a jedynie opowiedzieć o pewnej niezwykle ciekawej słabości sieci neuronowych, która bywa celem bardzo wyszukanych ataków, przypominających ataki hakerskie. Niestety, ze względu na brak dobrego tłumaczenia, będę posługiwał się ich angielską nazwą — *adversarial attacks*.

Zacznijmy od czegoś prostego. Co Szanowny Czytelnik widzi na poniższej fotografii?

Rysunek 11. Zdjęcie oryginalne[1]

[1] Źródło: Aleksander Mądry, Ludwig Schmidt, *A Brief Introduction to Adversarial Examples*, *https://gradientscience.org/intro_adversarial/* [dostęp 11.2024].

Spokojnie, to nie jest podchwytliwe pytanie. Na zdjęciu jest świnka. Jeśli Szanowny Czytelnik jej nie widzi, to… sam nie wiem. Umówmy się, że tam po prostu jest. I że nowoczesna sieć neuronowa, wytrenowana do rozpoznawania zdjęć, również ocenia, że na 91% jest to świnka.

A skoro tak dobrze nam poszło, to idziemy za ciosem. Co Szanowny Czytelnik widzi na tej fotografii?

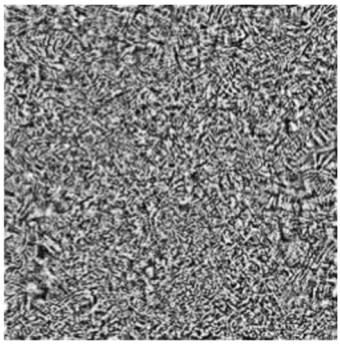
Rysunek 12. Zdjęcie zmodyfikowane[2]

Tu zapewne pojawia się konsternacja. Może jakaś pomyłka w druku? Nie? Szybki rzut oka na poprzedni obrazek — to przecież to samo zdjęcie, ta sama świnka. I być może dla Szanownego Czytelnika nadal jest to świnka, ale dla naszej nowoczesnej sieci neuronowej to na 99% jest samolot pasażerski. Tylko jakim cudem?

Ludzkie oko nie jest w stanie dostrzec bardzo subtelnej różnicy między tymi obrazami, niezależnie od ich jakości czy rozdzielczości. Pierwsze zdjęcie to oryginał. Drugie jest efektem zaburzenia każdego piksela oryginalnego zdjęcia o pewną bardzo niewielką, ale precyzyjnie dobraną wartość. To tak, jakbyśmy na nasze oryginalne zdjęcie nałożyli specjalnie przygotowany filtr, który zmienia wartości poszczególnych pikseli, ale w taki sposób, żeby dla nas — ludzi — oba zdjęcia wyglądały identycznie.

Dla ciekawskich: ten filtr wygląda tak jak na rysunku 13.

[2] Źródło: Aleksander Mądry, Ludwig Schmidt, *A Brief Introduction to Adversarial Examples*, *https://gradientscience.org/intro_adversarial/* [dostęp 11.2024].



Rysunek 13. Filtr wykorzystany do modyfikacji oryginalnego zdjęcia[3]

Hola, hola, przecież zobaczylibyśmy gołym okiem, gdyby ktoś majstrował z takim filtrem przy naszym zdjęciu. Zgadza się. Dlatego przed nałożeniem na obraz oryginalny był on dodatkowo wymnożony przez wartość 0,005 w celu sprowadzenia zaburzeń do jak najmniejszych wartości. Minimalnych, ale wystarczająco dużych, żeby wpłynęły na działanie klasyfikatora i zmiany jego predykcji ze „świnka" (91%) na „samolot pasażerski" (99%).

Wyróżniamy wiele typów *adversarial attacks*. Jednym z ważniejszych kryteriów ich podziału jest sposób przygotowania ataku w oparciu o jakąś niedoskonałość modelu lub procesu jego uczenia. Najbardziej oczywistym wydaje się tzw. zatruwanie danych (ang. *data poisoning*), polegające — jak łatwo się domyślić — na zmanipulowaniu zbioru uczącego, np. poprzez „podłożenie" błędnie oznaczonych przypadków treningowych.

Dotychczas mówiliśmy przede wszystkim o samodzielnym przygotowywaniu zbiorów uczących, więc taki scenariusz może wydawać się bardzo mało prawdopodobny. Wcale nie musi tak być. Dość popularną praktyką jest wykorzystywanie dużych zbiorów treści generowanych przez użytkowników końcowych (UGC, ang. *User Generated Content*), np. w systemach rekomendacyjnych albo modelach językowych. Wrócimy do tego tematu w rozdziale dotyczącym anotacji danych.

[3] Źródło: Aleksander Mądry, Ludwig Schmidt, *A Brief Introduction to Adversarial Examples*, *https://gradientscience.org/intro_adversarial/* [dostęp 11.2024].

Inne rodzaje ataków, jak ten ze świnką i samolotem pasażerskim, wykorzystują słabości już wyuczonych modeli. Nawet gdy nie wiemy nic na temat atakowanego modelu, możemy poszukać jego potencjalnych „dziur" metodą prób i błędów. W tym celu stosuje się całą masę częstych i bardzo drobnych manipulacji wejściem, żeby odkryć, jak wpływa to na wyjście. Zmieniamy wartości kilku pikseli i sprawdzamy. Zmieniamy wartości kilku innych i sprawdzamy ponownie. W rezultacie atakujący jest w stanie precyzyjnie ustalić, co i jak należy zmienić, żeby uniknąć detekcji lub kompletnie zmienić jej wynik.

Bardzo ciekawym rodzajem ataku jest tzw. ekstrakcja modelu (ang. *model extraction*), dotycząca przede wszystkim sieci, które były trenowane na wrażliwych bądź niejawnych danych. Metoda polega na specjalnym próbkowaniu wejścia, ale nie jak poprzednio, żeby odkryć „dziury" w modelu, tylko po to, żeby odtworzyć dane, na których model był uczony. Atakujący w taki sposób dobiera wejście, aby z tego, co model zwraca na wyjściu, dowiedzieć się, jak działa, a w rezultacie — na jakich danych został wytrenowany. Uzyskanych w ten sposób danych, wrażliwych bądź niejawnych, można użyć bezpośrednio w swojej nieetycznej i niemoralnej działalności, a można pokusić się o zreplikowanie oryginalnego modelu. Czasem używa się tu również szerszego terminu inżynieria wsteczna (ang. *reverse engineering*).

W 2018 roku w pracy zatytułowanej *Adversarial Patch* grupa naukowców z Google'a przedstawiła bardzo kreatywne rozwinięcie ataku ze świnką i samolotem pasażerskim. Opracowali naklejkę. Taką normalną, „fizyczną" naklejkę, która po umieszczeniu w pobliżu analizowanego obiektu kompletnie zmieniała wynik klasyfikacji modelu do rozpoznawania obrazów. I to nie byle jakiego modelu, bo jednego z najlepszych na tamte czasy — VGG16. Ale za to byle jakiego obiektu, bo w zasadzie dowolnego.

Na potrzeby prezentacji wybrano banana. Najpierw na stole leżał sam banan i tu nie było zaskoczeń — model przypisał obrazowi klasę „banan" z pewnością 97%. Gdy na scenę wjechała naklejka z normalnym zdjęciem tostera, to — zgodnie z oczekiwaniami — niewiele się zmieniło. Tak dobry model oczywiście nie dał się

na to nabrać. Ale gdy zamiast zwykłej na scenie pojawiła się naklejka przedstawiająca… hm, psychodeliczny toster po LSD albo efekt romansu tostera z tęczą, model zwariował i zaczął wskazywać klasę „toster" z pewnością 99%.

Ta druga naklejka została wygenerowana przy pomocy jednej z metod *adversarial attacks* na podstawie zdjęcia tostera. Czy przypominała toster? Może trochę. Zachęcam Szanownego Czytelnika do zerknięcia do pracy i samodzielnej oceny. Wystarczy wygooglować „adversarial patch". W artykule jest nawet link do filmu na YouTubie, w którym „na żywo" dokładane są naklejki i pokazywany jest aktualny odczyt z modelu. Naukowcy zwrócili uwagę, że taka naklejka może być dowolnie dystrybuowana przez internet, drukowana i używana do oszukiwania algorytmów rozpoznawania obrazów. Niezależnie od tego, co akurat znajduje się na wejściu. Przerażające, prawda?

Zatrzymajmy się na moment i pomyślmy o potencjalnych konsekwencjach. Wiemy już, że znając konkretny model rozpoznawania obrazów, możemy przygotować naklejkę, która ten model oszuka. Autonomiczne pojazdy korzystają z sieci neuronowych m.in. do rozpoznawania znaków drogowych. A co, jeśli ktoś przyklei podobną naklejkę na taki znak? Brzmi absurdalnie? No cóż, specjaliści z firmy McAfee[4] wcale nie musieli aż tak się wysilać, żeby sprowokować samochód marki Tesla do niebezpiecznego zachowania. Nie potrzebowali do tego specjalnie przygotowanych naklejek, a jedynie 5-centymetrowego kawałka czarnej taśmy. W 2019 roku przeprowadzili test polegający na doklejeniu takiej taśmy do trójki w znaku ograniczenia prędkości do 35 mil na godzinę.

Po lewej mamy właściwy wygląd znaku, a po prawej — ten sam znak z przyklejoną taśmą.

[4] Steve Povolny, Model Hacking, *ADAS to Pave Safer Roads for Autonomous Vehicles*, *https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/* [dostęp 11.2024].

Rysunek 14. Po lewej: prawidłowy znak ograniczenia prędkości. Po prawej: ten sam znak z doklejonym kawałkiem czarnej taśmy[5]

I to wystarczyło, żeby Tesla przyspieszyła z 35 mil na godzinę (ok. 56 km/h) do 85 mil na godzinę (ok. 136 km/h). Nie muszę chyba dodawać, że miało to miejsce na drodze przystosowanej do tego pierwszego limitu. Szanowny Czytelnik mógłby w tym momencie stwierdzić, że przecież człowiek też mógłby dać się nabrać na taki trik. I być może osoba niedowidząca rzeczywiście mogłaby pomylić trójkę z ósemką, ale to nie znaczy, że bezrozumnie docisnęłaby gaz do dechy. Na szczęście z tyłu nosa nosimy galaretowany komputer, który — w przeciwieństwie do współczesnych sieci neuronowych — jest przystosowany do wykonywania bardzo wielu różnych zadań, w tym przede wszystkim do utrzymywania nas przy życiu.

***

Żeby było jasne — moim celem nie jest straszenie Szanownego Czytelnika sztuczną inteligencją, która nic, tylko przebiera wirtualnymi nóżkami, żeby nas zgnoić, zabić albo zniewolić. Nic z tych rzeczy. Przynajmniej jeszcze nie teraz. Warto zaznaczyć, że eksperymenty, które tu przytaczam, mają już ładnych kilka lat i — jak można się spodziewać — problemy, których dotyczyły, zostały już w znacznym stopniu rozwiązane. Za moment zresztą opowiem o kilku metodach radzenia sobie z *adversarial attacks*.

[5] Steve Povolny, Model Hacking, *ADAS to Pave Safer Roads for Autonomous Vehicles*, *https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/* [dostęp 11.2024].

Ale najpierw, wracając do intencji, moja przestroga. Rozwiązania oparte na uczeniu maszynowym miewają istotne słabości i luki w zabezpieczeniach lub sposobie funkcjonowania. Te słabe punkty mogą być wykorzystywane do ich obejścia bądź oszukania, a w efekcie do zaszkodzenia nam — ich użytkownikom. I w tym sensie nie różnią się od szeroko rozumianego oprogramowania, którego luki w zabezpieczeniach mogą być celem ataku hakerów.

Różnica jest taka, że kod zwykłego programu możemy w całości przeanalizować, przetestować i te luki znaleźć. W przypadku sieci neuronowych nadal mamy do czynienia z czarną skrzynką. Dlatego zalecam zdroworozsądkowy, umiarkowany optymizm. Albo jak mawiał mój nauczyciel matematyki z liceum: „ufaj, ale sprawdź". W zasadzie mawiał też „jak się nie ma, co się lubi, to się kocha żonę", więc nie jestem pewien, czy był najlepszym autorytetem od powiedzonek…

No dobrze, to jak się bronić przed *adversarial attacks*? Oczywiście staropolską metodą walki z kacem, czyli „klin klinem", albo — jak kto woli — w myśl zasady „czym się strułeś, tym się lecz". Najpopularniejszą metodą jest uwzględnienie w samym procesie uczenia modelu przypadków, które potencjalnie mogą służyć do ataku. Uczenia lub… douczenia, jeśli mamy już dobry model, ale chcemy go dodatkowo uodpornić na ataki. Taki proces nazywa się *adversarial training* i jest zazwyczaj bardzo powolny i bardzo kosztowny. Dlaczego? Dlatego, że sami musimy znaleźć potencjalne słabości modelu, a następnie wygenerować wiele kombinacji przypadków uczących dla każdej z tych słabości.

Załóżmy, że chcemy wykrywać, kiedy jeden z uczestników czatu zaczyna obrażać innych. Może to robić przy użyciu różnego rodzaju inwektyw, takich jak „o, motyla noga, ty niesforny huncwocie". I tak — zdaję sobie sprawę, że nawet dzieci w przedszkolu tak nie mówią, ale nie wiadomo, kto to będzie czytał, więc skupmy się na problemie, a nie konkretnym przypadku.

Wiemy, że *adversarial attack* to próba manipulacji wejściem, która ma na celu oszukanie modelu. Ale gdy zamierzamy kogoś obrazić, to z reguły chcemy, żeby osoba obrażana wiedziała, że jest obrażana. Możemy w tym celu zamienić niektóre z liter na podobnie wyglądające symbole, np. zapisując naszą inwektywę jako: „o, motyla noga, ty nie$forny huncw0cie". Widzimy, że po prostu zamieniłem „s"

na symbol dolara, a „o" na zero. Człowiek bez problemu odczyta taką wiadomość, a nieprzystosowany do tego model będzie miał nie lada problem.

Nie potrzeba nawet szczególnej kreatywności, żeby wymyślić całą masę podobnych podmianek, a to tylko jedna z bardzo wielu opcji. Możemy powtórzyć jakąś literę („huncwoocie"), pominąć ją („huncwcie") albo zamienić na inną („huncwucie"). Możemy zapisać wyraz z błędem ortograficznym („hóncwocie"). Możemy powstawiać w środek wyrazu spacje („hu nc wo cie"), myślniki („hun-cwo-cie") albo dowolne inne znaki („h_u_n_c_w_o_c_i_e"). Ogranicza nas tylko nasza wyobraźnia, a do tego możemy stosować dowolne kombinacje tych zabiegów. Chyba Szanowny Czytelnik już widzi, dokąd to zmierza…

No właśnie — dlatego *adversarial training* jest tak powolnym i kosztownym procesem. Najpierw musimy wygenerować pierdylion przeróżnych kombinacji takich ataków, a następnie douczać nasz model tak długo, aż nabędzie odpowiedniej odporności. Ale cóż począć? Zaciska się zęby i robi to, bo w wielu zastosowaniach, takich jak obrazowanie medyczne, biometryczna identyfikacja tożsamości czy wspomniane już wcześniej pojazdy autonomiczne, koszt ewentualnego błędu jest nieporównywalnie wyższy. Na szczęście istnieją dedykowane metody i narzędzia, które znacznie nam w tym pomagają.

Na koniec wspomnę o jeszcze jednej technice defensywnej, która pokazuje, że nie tylko hakerzy wykazują się ponadprzeciętną kreatywnością, ale również strona broniąca się bywa niesamowicie pomysłowa, kontrując działania napastników. Tzw. destylacja obronna (ang. *defensive distillation*), bo o niej mowa, to technika polegająca na zbudowaniu nie jednego, a co najmniej dwóch modeli.

Pierwszy model jest trenowany do rozwiązywania konkretnego zadania na rzeczywistych i oryginalnych danych uczących. Na razie nic specjalnego, prawda? Może poza faktem, że tego modelu nigdy nikomu nie udostępniamy. Nazywamy go „nauczycielem" i używamy tego, co zwraca na wyjściu, do wytrenowania drugiego modelu — „ucznia". I dopiero ten model jest udostępniany. A zatem „uczeń" nie ma kontaktu z oryginalnym zbiorem uczącym, przez co jest w naturalny sposób odporny na ekstrakcję modelu. Co więcej, może być znacznie mniejszy, szybszy i tańszy niż „nauczyciel". Sprytne, prawda?

MODEL JEST CO NAJWYŻEJ TAK DOBRY, JAK DANE UŻYTE DO JEGO WYTRENOWANIA

To zdanie powinno być mantrą wszelkiej maści data scientistów. Mam nadzieję, że Szanowny Czytelnik wybaczy mi kolejną angielską nazwę, ale polskie odpowiedniki w stylu „naukowiec danych" czy jeszcze lepiej — „mistrz danych" absolutnie mi nie leżą. A skoro o danych mowa, to czas porozmawiać o procesie ich anotacji. To niezwykle ważki temat w kontekście uczenia maszynowego, a przy tym jeden z najbardziej marginalizowanych i lekceważonych, również przez osoby profesjonalnie zajmujące się sztuczną inteligencją, nad czym wielce ubolewam.

Idealnie wpisuje się tu angielski zwrot „garbage in, garbage out", w skrócie GIGO, czyli „śmieci na wejściu, śmieci na wyjściu". Nie da się zbudować dobrego modelu bez wysokiej jakości danych uczących. Co więcej, jest to praktycznie jedyny obszar uczenia maszynowego, gdzie możemy wpuścić nieco wiedzy eksperckiej z dziedziny, do której należy rozwiązywany problem. Jak dla mnie to aż nadto, żeby podejść do tematu na poważnie. A postaram się, żeby było też ciekawie…

Wiemy już, do czego są nam potrzebne anotowane zbiory danych — do uczenia, walidacji i testowania modeli uczenia maszynowego. Sam termin „anotacja danych" to taka fikuśna nazwa na oznaczanie dokumentów lub ich fragmentów etykietami odpowiadającymi kategoriom, do których będziemy te dokumenty lub ich fragmenty klasyfikować. Oczywiście, pisząc tu o dokumentach, mam na myśli nie tylko dokumenty tekstowe, ale dowolne dane, podawane na wejście modelu.

Mogą to być obrazy, nagrania audio i wideo, ustrukturyzowane dane, takie jak wyniki morfologii krwi, a nawet pozycje figur na szachownicy.

Anotując dane tekstowe, możemy oznaczać całe dokumenty, tak jak robiliśmy to do tej pory przy analizie sentymentu, gdzie całej recenzji przypisywaliśmy jedną z trzech etykiet — „pozytywna", „negatywna" lub „neutralna". Możemy jednak pójść krok dalej i oznaczać konkretne fragmenty tekstu, dotyczące poszczególnych aspektów ocenianego produktu lub usługi i przypisanego im wydźwięku emocjonalnego. W przypadku recenzji smartfonów możemy wyróżniać sentyment w stosunku do ekranu, szybkości działania czy długości życia baterii, a w przypadku restauracji — jakości jedzenia, obsługi czy atmosfery lokalu.

To, co i w jaki sposób oznaczamy, z reguły definiuje tzw. instrukcja anotacji danych — dokument, który opracowujemy na potrzeby rozwiązywania konkretnego problemu przy użyciu sztucznej inteligencji. To takie kompendium wiedzy, opisujące w szczegółach wszystkie aspekty tego konkretnego procesu — od precyzyjnej definicji problemu, przez wskazanie wybranego narzędzia i metod oznaczania danych, aż po zestaw problematycznych przykładów, sygnalizujących potencjalne trudności i podpowiadających, jak sobie z nimi radzić. Nowy problem to zazwyczaj nowa instrukcja anotacji danych.

Tworząc taką instrukcję, musimy odpowiedzieć sobie na szereg pytań dotyczących tego, jak szczegółowo chcemy oznaczać nasze dane. Czy zostajemy na poziomie całego dokumentu, czy oznaczamy konkretne fragmenty? Jeśli fragmenty, to jakie — całe zdania czy może frazy? Jeśli zdania, to co w przypadku kiepskiej interpunkcji? Jakimi kategoriami będziemy oznaczać nasze dane? Czy wystarczy klasyfikacja binarna — „tak/nie", czy może wprowadzamy wiele różnych klas? Czy pojedynczy dokument lub fragment może należeć do wielu różnych klas jednocześnie? Co w przypadku, kiedy poszczególne klasy będą się ze sobą zazębiać?

Z im trudniejszym i bardziej złożonym zjawiskiem mamy do czynienia, tym ważniejsza jest precyzyjna definicja problemu. Jeśli skomplikowane zagadnienie, takie jak mowa nienawiści, potraktujemy hasłowo i powierzchownie, to oznaczenia dwóch anotatorów mimo najszczerszych chęci mogą skrajnie się od siebie różnić. W takim przypadku powinniśmy zwrócić się do ekspertów w danej dziedzinie i wraz z nimi przygotować porządne definicje klas, których zamierzamy szukać.

A i to często nie wystarczy w konfrontacji z rzeczywistością. Gdy tylko uporamy się z teorią i zerkniemy w prawdziwe dane, szybko okaże się, że nadal istnieje spory wycinek problemu, z którym nie do końca wiadomo, co należy robić. A to przypadek, który jest częścią zjawiska, ale nie pasuje do żadnej ze zdefiniowanych kategorii. A to przypadek „na granicy" modelu, ciut za słaby, żeby się do niego załapać. A to przypadek „na granicy" dwóch klas, z których trzeba wybrać tylko jedną. Co począć? Zbieramy te wszystkie przypadki, omawiamy, opisujemy i dorzucamy do instrukcji, żeby nasi anotatorzy w razie wątpliwości mogli sprawdzić, jak sobie z nimi radzić.

Niestety, to jeszcze nie koniec. Niezależnie od tego, jak dobrze przygotujemy instrukcję anotacji danych, nie unikniemy całkowicie rozbieżności w oznaczeniach. Zawsze pojawi się jakiś nowy, dyskusyjny przypadek albo górę wezmą osobiste przekonania oceniającego. Czasem przyczyny są jeszcze bardziej prozaiczne, a błędy i przeoczenia wynikają po prostu ze zmęczenia dość nużącą i monotonną pracą. Dlatego popularną praktyką jest niezależne oznaczanie każdego dokumentu przez co najmniej dwóch, a czasem nawet trzech i więcej anotatorów.

Hola, hola, ale przecież sieć neuronowa potrzebuje do nauki jednoznacznie oznaczonych przykładów, prawda? Zgadza się. Jeśli mamy porządną instrukcję, to większość przypadków będzie oznaczona identycznie przez wszystkich anotatorów. I te przypadki możemy uznać za poprawnie oznaczone. Ale co z pozostałymi?

Tu cały na biało wchodzi superanotator, który mocą swojego autorytetu rozstrzyga i ujednoznacznia wszelkie dyskusyjne przypadki i rozbieżności w myśl zasady, że „moja racja jest najmojsza". Żartuję… ale nie do końca. Superanotator to z reguły ekspert z szeroką wiedzą domenową i bogatym doświadczeniem w anotacji danych. Opiekuje się całym procesem, regularnie rozmawia z anotatorami, waży ich argumenty, a instrukcję zna praktycznie na pamięć. I dlatego jest w stanie podejmować najlepsze możliwe decyzje. Dlatego jest super…

***

To w zasadzie wyczerpywałoby podręcznikowe podejście do tematu. Ale co w sytuacji, kiedy wszyscy anotatorzy jednogłośnie, a mimo to błędnie oznaczyliby jakiś przypadek. Zgoda — mało prawdopodobne, ale przecież nie niemożliwe. Wróćmy

na moment do mowy nienawiści. Jest to zjawisko rzadkie — powiedzmy, że dotyczy 1% wszystkich wiadomości. Jeśli losowo wybierzemy zbiór 1000 wiadomości do anotacji, to ilu spodziewamy się tam przypadków mowy nienawiści? Około dziesięciu, prawda? No właśnie. A miejmy na uwadze, że oznaczanie danych to żmudny i nużący proces. W przypadku tak dużej dysproporcji klas anotatorzy w naturalny sposób będą dryfować w stronę 0 (jeśli 1 to mowa nienawiści, a 0 to jej brak). Wbrew pozorom po kilku godzinach pracy nietrudno o pomyłkę.

Jednym ze sposobów radzenia sobie z tym problemem jest losowa weryfikacja jednomyślnych oznaczeń przez superanotatora. Niestety, w przypadku dużej dysproporcji klas jest to metoda zawodna, bo zgodność między anotatorami jest ogromna, a szansa na trafienie w błędne oznaczenie — bardzo niewielka. Cóż zatem można uczynić? Na przykład złożyć roczne jagnię bez skazy w ofierze całopalnej. Tak tylko sprawdzam, czy Szanowny Czytelnik nie odpłynął gdzieś myślami. Można użyć uczenia maszynowego w procesie anotacji danych do uczenia maszynowego. No bo czemu by nie?

Spokojnie, jest to dużo mniej skomplikowane, niż może się wydawać. Załóżmy, że anotacja danych już się rozpoczęła. Bierzemy pierwsze dostępne dane i trenujemy na nich model uczenia maszynowego, który następnie obsadzamy w roli kolejnego… anotatora. W Samurai Labs nazywamy taki model wirtualnym anotatorem. Ba, nasz wirtualny anotator okazał się anotatorką i dostał nawet swoje imię — Wiktoria.

A z Wiktorią to już można konie kraść. Nie męczy się, więc ten typ błędów odpada. Jeśli wcześniejsze anotacje były niespójne, to w kolejnej rundzie nie będzie przesadnie zgodna ze swoimi ludzkimi koleżankami i kolegami. I dobrze, bo jeśli teraz wszyscy anotatorzy jednogłośnie popełnią błąd, to ona im go wytknie, a przypadek trafi do superanotatora.

Wiktoria pomaga nam również w dobieraniu kolejnych przypadków do anotacji. Połowę nowego zbioru możemy nadal wybierać losowo, ale drugą połowę może nam wybrać Wiktoria. A jak? A tak, że możemy dodać do zbioru przypadki, które Wiktoria uznaje za pozytywne, czyli zawierające to, czego tak naprawdę chcemy szukać. A że zazwyczaj szukamy igły w stogu siana, to zwiększenie procentowego udziału takich wyników w naszym zbiorze jest na wagę złota.

Ale na tym nie kończą się jej sztuczki. Wiktoria potrafi całkiem nieźle wykrywać wspomniane wcześniej przypadki „na granicy" modelu lub jego klas. Jest to o tyle istotne, że obecność takich przykładów w zbiorze uczącym w największym stopniu przyczynia się do poprawy skuteczności detekcji. W myśl jednej z moich ulubionych zasad — „pracuj mądrzej, a nie ciężej". Ach, ta nasza Wiktoria. Tak pięknie przekreśliła wszystko to, co do tej pory pisałem o antropomorfizacji i przypisywaniu modelom uczenia maszynowego ludzkich cech…

***

Anotacja danych jest często postrzegana jako przykra konieczność. Eksperymenty, trenowanie modeli, testowanie różnych podejść, porównywanie i analiza wyników — to jest ta „fajna" część. Wszyscy dobrze się bawią, a biedna anotacja stoi sama w kącie i podpiera ściany. Między innymi dlatego wszystkie porządne i publicznie dostępne zbiory danych są eksploatowane do granic możliwości. A jeśli pojawia się okazja, żeby pozyskać dobrej jakości dane bez konieczności „brudzenia sobie rąk", to grzechem byłoby z niej nie skorzystać. Zdarza się, że takie dane po prostu „leżą na ulicy" i tylko czekają, aż ktoś zaradny i pomysłowy po nie sięgnie.

Nasza stara dobra znajoma — analiza sentymentu — będzie tu doskonałym przykładem. Gdy piszemy recenzję, z reguły dostajemy możliwość dodatkowej oceny produktu lub usługi. Najczęściej są to jakieś gwiazdki lub punkty, np. w skali od 1 do 5, gdzie 1 to tragedia, a im więcej, tym lepiej. Średnią z tych ocen wykorzystuje się później choćby do sortowania i filtrowania wyników wyszukiwania. A skoro mamy takie dane, to dlaczego by nie uznać recenzji z 1 czy 2 gwiazdkami za negatywne, 3-gwiazdkowych za neutralne, a 4- i 5-gwiazdkowych za pozytywne? Wtedy każda z tych recenzji miałaby przyporządkowaną jedną z trzech możliwych etykiet, a my dostalibyśmy gotowy zbiór uczący bez spędzenia choćby sekundy na anotacji danych.

Jak Szanowny Czytelnik może się domyślać, takie zbiory danych mogą być przeogromne. Według platformy Statista w samym tylko serwisie Tripadvisor liczba recenzji i ocen przekroczyła miliard w 2021 roku. Jeśli weźmiemy pod uwagę niewielkie skomplikowanie problemu i sieci, która go rozwiązuje, oraz zainteresowanie marek i brandów monitorowaniem mediów społecznościowych w czasie

rzeczywistym, to nikogo nie powinno szczególnie dziwić, że w ubiegłej dekadzie jak grzyby po deszczu wyrastały przeróżne firmy i firemki zajmujące się właśnie analizą sentymentu. Rynek eksploduje, gdy praktycznie nieograniczony dostęp do danych spotyka możliwości technologiczne i kogoś, kto chce za to wszystko płacić.

Inny przykład. Hansard to nazwa oficjalnego raportu, zawierającego kompletny zapis obrad parlamentu brytyjskiego. Ale nie tylko — wiele krajów, onegdaj należących do imperium, nad którym nigdy nie zachodzi słońce, publikuje własne wersje Hansarda. Szczególnie interesująca jest wersja kanadyjska, ponieważ Kanada — jak Szanowny Czytelnik zapewne wie — jest krajem dwujęzycznym, z urzędowym językiem angielskim i francuskim.

Wszystko, co zostało w parlamencie wypowiedziane w jednym języku, jest w ciągu nocy transkrybowane, redagowane i tłumaczone przez rządowych tłumaczy na ten drugi język. Tak że następnego ranka dostępny jest już pełen zapis dwujęzyczny. I te zapisy były gromadzone od wielu, wielu lat. A skoro mamy doskonałej jakości przekłady ogromnych zbiorów danych, to… Dokładnie — wykorzystanie kanadyjskiego Hansarda było jednym z najważniejszych kamieni milowych w badaniach nad tłumaczeniem maszynowym. A żeby było śmieszniej, to przeciętny Kanadyjczyk prawdopodobnie nigdy w życiu nie słyszał nazwy Hansard.

Jeszcze innym sposobem jest zaprzęgnięcie do roboty swoich własnych użytkowników, praktycznie w roli darmowych anotatorów. Ale też postawmy sprawę jasno — nikt z nas nie robi tego wbrew woli, pod przymusem. Sami udostępniamy swoje zdjęcia, gdzie oznaczamy siebie, swoich bliskich i znajomych, żeby dostawali powiadomienia i ochoczo przybywali nam je wszystkie polajkować. A czym to się niby różni od anotacji danych do uczenia modelu? Albo kiedy sami poprawiamy asystenta głosowego, który nie do końca zrozumiał to, czego od niego chcieliśmy? Albo kiedy zgłaszamy, że automatyczna odpowiedź nie była pomocna? Albo dana rekomendacja nie do końca trafiła w nasze gusta?

Oczywiście nie ma w tym nic złego. Przynajmniej tak długo, jak długo te dane są wykorzystywane wyłącznie do poprawy jakości usług, z których korzystamy. Miejmy jednak świadomość, że wielkie korporacje wiedzą o nas znacznie więcej niż nasi najbliżsi. A czasem nawet więcej niż my sami. I nie dajmy się wodzić za nos

— nie ma darmowych obiadów. To, że nie płacimy pieniędzmi, nie znaczy, że nie płacimy w ogóle. Płacimy, płacimy. Albo czasem spędzonym na oglądaniu reklam, albo właśnie danymi. Większość z nas bardzo roztropnie wydaje pieniądze. Więc dlaczego nie podchodzimy równie roztropnie do zarządzania naszym czasem i prywatnością?

O SPLOCIE SZCZĘŚLIWYCH NEURONÓW

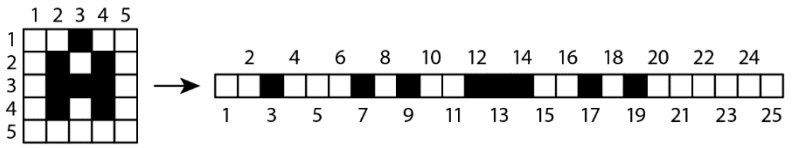
Omawiając perceptrony wielowarstwowe (MLP), obiecałem Szanownemu Czytelnikowi również opowieść o tych trochę bardziej złożonych rodzajach i architekturach sieci neuronowych. Tu zaczniemy, ale znowu muszę prosić o nieco więcej uwagi i skupienia niż zwykle. To nie będzie rozdział „do poduszki". Jeśli złapałem Szanownego Czytelnika w niezbyt sprzyjających okolicznościach, to proponuję przejść do kolejnego rozdziału, a tu wrócić później — na świeżo, z wypoczętą głową.

A może to ja przesadzam z zachowawczością? Tak naprawdę cała ta trudność bywa pozorna, a wynika z naszych uprzedzeń do matematyki, których nabawiliśmy się na różnych etapach naszej edukacji. Zdaję sobie sprawę, że wielu czytelników zareaguje alergicznie na sam widok macierzy. O fuj, macierze — bez sensu, za trudne, nie dam rady, odpuszczam. Niesłusznie. Działania, które będziemy wykonywać, sprowadzają się do dodawania i mnożenia liczb całkowitych. Nic więcej. A macierze są tu tylko dlatego, że bardzo dobrze reprezentuje się nimi obrazy — mają dwa wymiary, tak jak zdjęcie, a kolejne liczby to po prostu wartości kolejnych pikseli.

Jeszcze tylko jedna uwaga. Perceptrony wielowarstwowe, czyli sieci typu *feed-forward*, ze względu na relatywnie prostą strukturę i algorytm uczący mogliśmy omówić kompleksowo, w całości, od A do Z. W przypadku pozostałych sieci to się nam nie uda. Zamiast grzęznąć w masie zbędnych detali, skupimy się raczej na najważniejszych pomysłach, koncepcjach i zastosowaniach. Na samym słodkim. Na tym, co najciekawsze — jaki problem napotkano i jak udało się go rozwiązać.

A zaczniemy od sieci konwolucyjnych, czasem nazywanych też splotowymi. Ich angielska nazwa to „convolutional neural network", z popularnym, choć dwuznacznym skrótem — CNN. Tak samo jak znana amerykańska telewizja informacyjna.

Sieci konwolucyjne są odpowiedzią na nieprzesadnie udane próby zastosowania prostszych rozwiązań, takich jak perceptrony wielowarstwowe, do zadań związanych z przetwarzaniem obrazu. Pierwszym problemem jest rozmiar wektora wejściowego, który otrzymalibyśmy, mapując dwuwymiarowy obraz na jeden wymiar — tnąc go na paski o szerokości jednego piksela i układając jeden za drugim. Dokładnie tak, jak na poniższym schemacie:

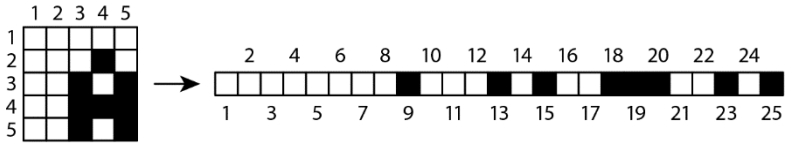


Rysunek 15. Przykład mapowania dwuwymiarowego obrazu do jednowymiarowego wektora (litera A)

Widzimy, że maleńki obraz 5×5 jest mapowany do wektora zawierającego 25 liczb. Dla obrazu w rozdzielczości 1080p (1920×1080) wektor wejściowy miałby ponad 2 miliony wartości. Z kolei zdjęcia 4K (3840×2160) to już grubo ponad 8 milionów liczb.

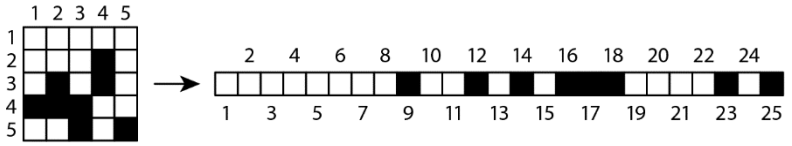
Drugim, znacznie poważniejszym problemem, również wynikającym z takiego mapowania, jest utrata informacji o kształtach obiektów na obrazie. Zerknijmy jeszcze raz na schemat. Na obrazku pomimo bardzo słabej rozdzielczości powinno dać się zobaczyć literę A. Mam nadzieję, że Szanowny Czytelnik też ją widzi. Na zmapowanym wektorze zobaczymy tylko czarno-białe linie i punkty. Ciężko na tej podstawie odgadnąć, co przedstawiał oryginalny obraz.

Ale to nie koniec zmartwień. Gdybyśmy teraz chcieli przesunąć naszą literę o jeden piksel w dół i w prawo, to na obrazie bez problemu rozpoznamy ten sam kształt. A co dostaniemy po zmapowaniu go na jeden wymiar? Przekonajmy się sami:



Rysunek 16. Przykład mapowania dwuwymiarowego obrazu do jednowymiarowego wektora (przesunięta litera A)

A teraz porównajmy tylko prawe strony obu schematów. Czy jesteśmy w stanie ocenić na pierwszy rzut oka, że jest to ta sama litera? Na pewno? Bo bardzo podobnie może wyglądać coś, co w ogóle nie przypomina litery A:



Rysunek 17. Przykład mapowania dwuwymiarowego obrazu do jednowymiarowego wektora (szum)

Sztuczne neurony i sieci neuronowe są inspirowane działaniem swoich biologicznych odpowiedników. To już wiemy. Sieci konwolucyjne idą krok dalej i starają się naśladować sposób działania naszego mózgu w zakresie przetwarzania bodźców wzrokowych. Okazuje się bowiem, że różne części mózgu przetwarzają różne cechy obrazu. Jedne grupy neuronów specjalizują się w rozpoznawaniu krawędzi obiektów w różnych orientacjach, a zupełnie inne odpowiadają za odczytywanie jasności czy barw. Jedne zbierają najprostsze bodźce, a inne łączą je w coraz bardziej skomplikowane kształty.

Skąd to wiemy? Od kotów. David Hubel i Torsten Wiesel otrzymali w 1981 roku Nagrodę Nobla za badania prowadzone w latach 1959 – 1963. Pominę szczegóły, bo metody obu panów uznaję za wątpliwe etycznie. Powiedzmy, że ich eksperymenty polegały na pokazywaniu kotom serii obrazków — podłużnego prostokąta obracającego się wokół własnej osi. Poprzez mikroelektrody odczytywano sygnały elektryczne w kocim mózgu. Nigdy nie pytaj: kobiety o wiek, mężczyzny o zarobki, a panów naukowców o to, skąd w kocich głowach wzięły się mikroelektrody. Okazało się, że na każdy obrazek reagowała inna grupa neuronów —

inna na prostokąt w pozycji poziomej, inna w pionowej, a jeszcze inna w ukośnej. W ten sposób odkryto, że pierwszorzędowa kora wzrokowa, gdzie analizowane są informacje przesyłane z siatkówki oka, zawiera różne grupy neuronów, reagujące na różne kąty nachylenia bodźca.

***

Wróćmy do sieci. Na wejściu mamy obraz. Nie będzie to dla nas szczególnie istotne, ale dla porządku wypada wspomnieć, że jeśli jest to obraz kolorowy, to wejście będzie podzielone na trzy kanały, odpowiadające trzem kolorom składowym w modelu RGB, czyli czerwonemu (R), zielonemu (G) i niebieskiemu (B). To tak, jakbyśmy zamiast jednego mieli trzy jednokolorowe zdjęcia, które po nałożeniu na siebie dają zdjęcie oryginalne. W przypadku obrazów w skali szarości wystarczy jeden kanał.

I tu pojawia się pierwsza nowość — zupełnie nowa warstwa, w której każdy kanał jest poddawany operacji konwolucji, nazywanej też splotem. Chodzi o pewne specyficzne przekształcenie, mające na celu wydobycie konkretnych informacji o cechach obrazu (kanału). Na razie brzmi to bardzo tajemniczo, ale za moment maski opadną i wszystko stanie się jasne. Ta warstwa ze względu na wykonywaną w niej operację nosi nazwę warstwy konwolucyjnej lub splotowej. Te pojęcia są zresztą synonimami, więc od tej pory będę ich używał zamiennie.

Konwolucja polega na zastosowaniu specjalnych filtrów, nazywanych też jądrami lub kernelami, które przetwarzając kanał wejściowy, produkują zupełnie nowy kanał, zwany mapą cech. Filtry to z reguły małe kwadratowe macierze, np. 3×3 lub 5×5, które przesuwa się po kanale wejściowym, od lewej do prawej i od góry do dołu, zaczynając od lewego górnego rogu. W każdym takim kroku najpierw wymnaża się wartości pikseli kanału wejściowego przez odpowiadające im parametry filtra, a następnie poszczególne iloczyny są ze sobą sumowane do jednej wartości, stanowiącej nowy piksel w mapie cech. Jeśli filtr jest macierzą 3×3, to z każdych 9 pikseli kanału wejściowego dostajemy jedną wartość piksela w mapie cech.

Zerknijmy na poniższy przykład. Widzimy trzy macierze — pierwsza (6×6) to kanał wejściowy, druga (3×3) to filtr, a trzecia (4×4) — na razie wypełniona kropkami — to kanał wyjściowy, czyli mapa cech. Krzyżyk reprezentuje operację konwolucji, czyli działanie filtra na kanał wejściowy. Za moment zamienimy kropki z ostatniej macierzy na prawdziwe wartości mapy cech.

$$\begin{pmatrix} 7 & 8 & 8 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 0 & 1 \\ 8 & 7 & 8 & 1 & 0 & 0 \\ 9 & 8 & 8 & 0 & 2 & 0 \\ 8 & 8 & 9 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

Ale zanim to zrobimy, chciałbym zwrócić uwagę Szanownego Czytelnika na kanał wejściowy, gdyż jego wartości zostały dobrane nieprzypadkowo. Po lewej stronie są wysokie (7 – 9), a po prawej — niskie (0 – 2). Co zobaczylibyśmy na odpowiadającym mu obrazie, gdyby wartości wysokie odpowiadały jasnym pikselom, a niskie — ciemnym? Krawędź, prawda? Pionową krawędź, przebiegającą z góry na dół przez sam środek obrazu. Dodam tylko, że filtr również został dobrany nieprzypadkowo, ale z omówieniem jego zastosowania zaczekamy do końca wyznaczania mapy cech.

No to liczymy. Jako się rzekło, zaczynamy od lewego górnego rogu. Bierzemy okienko wielkości naszego filtra (3×3) i nakładamy je na kanał wejściowy. Dla ułatwienia wartości kanału wejściowego, brane pod uwagę przy wyznaczaniu nowego piksela w mapie cech, zostały otoczone nawiasami kwadratowymi. Analogicznie została oznaczona wartość piksela wyznaczanego w tym kroku.

$$\begin{pmatrix} [7] & [8] & [8] & 0 & 1 & 0 \\ [9] & [9] & [7] & 0 & 0 & 1 \\ [8] & [7] & [8] & 1 & 0 & 0 \\ 9 & 8 & 8 & 0 & 2 & 0 \\ 8 & 8 & 9 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} [1] & . & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

Mnożymy i sumujemy. Pierwszą wartość z mapy cech wyznaczamy w następujący sposób:

$$7 \times 1 + 8 \times 0 + 8 \times (-1)$$
$$+9 \times 1 + 9 \times 0 + 7 \times (-1)$$
$$+8 \times 1 + 7 \times 0 + 8 \times (-1) = 1$$

Przesuwamy okienko o jedną pozycję w prawo i liczymy drugą wartość w analogiczny sposób:

$$\begin{pmatrix} 7 & [8] & [8] & [0] & 1 & 0 \\ 9 & [9] & [7] & [0] & 0 & 1 \\ 8 & [7] & [8] & [1] & 0 & 0 \\ 9 & 8 & 8 & 0 & 2 & 0 \\ 8 & 8 & 9 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & [23] & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

Kolejne dwa kroki pozwalają nam dotrzeć do prawej krawędzi kanału wejściowego, a wtedy wracamy do lewej strony i przesuwamy okienko o jedną pozycję w dół, wyznaczając piątą wartość:

$$\begin{pmatrix} 7 & 8 & 8 & 0 & 1 & 0 \\ [9] & [9] & [7] & 0 & 0 & 1 \\ [8] & [7] & [8] & 1 & 0 & 0 \\ [9] & [8] & [8] & 0 & 2 & 0 \\ 8 & 8 & 9 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 23 & 22 & 0 \\ [3] & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

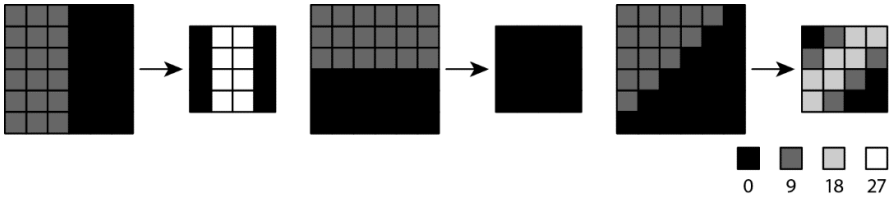
Całą procedurę powtarzamy tak długo, aż dotrzemy do ostatniej, szesnastej wartości z mapy cech:

$$\begin{pmatrix} 7 & 8 & 8 & 0 & 1 & 0 \\ 9 & 9 & 7 & 0 & 0 & 1 \\ 8 & 7 & 8 & 1 & 0 & 0 \\ 9 & 8 & 8 & [0] & [2] & [0] \\ 8 & 8 & 9 & [0] & [1] & [0] \\ 9 & 9 & 7 & [0] & [1] & [0] \end{pmatrix} \times \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 23 & 22 & 0 \\ 3 & 23 & 21 & 0 \\ 0 & 22 & 22 & 1 \\ 2 & 25 & 20 & [0] \end{pmatrix}$$

Zatrzymajmy się na chwilę i przyjrzyjmy kanałowi wyjściowemu. Dla przypomnienia — wysokie wartości to piksele jasne, a niskie — ciemne. Co widzimy? Filtr rozpoznał i wyostrzył pionową krawędź. Teraz zarówno po lewej, jak i po prawej

stronie mamy niskie wartości (0 – 3), a pośrodku, w miejscu, gdzie przebiegała krawędź — bardzo wysokie (20 – 25).

Jeśli Szanowny Czytelnik jeszcze nie widzi tego wszystkiego oczyma wyobraźni, to nic nie szkodzi. Spróbujemy jakoś temu zaradzić. Na poniższym schemacie przedstawiono — tym razem graficznie — trzy użycia tego samego filtra na trzech różnych krawędziach — pionowej, poziomej i skośnej. Z tą różnicą, że dla uproszczenia wartości pikseli ograniczono wyłącznie do czterech wartości (0, 9, 18 i 27), a każdej z nich przypisano inny odcień szarości.



Rysunek 18. Przykład zastosowania badanego filtra dla różnych krawędzi: pionowej (lewa), poziomej (środek) oraz ukośnej (prawa). W prawym dolnym rogu znajdują się wartości pikseli wraz z odpowiadającymi im odcieniami

Okazuje się, że taki filtr doskonale wykrywa i uwydatnia krawędzie pionowe, całkiem nieźle radzi sobie ze skośnymi, ale jest kompletnie ślepy na orientację poziomą. Podobnie jak odpowiednie grupy neuronów w kocim mózgu. Możemy powiedzieć, że nasz filtr wyodrębnia pewną konkretną cechę obrazu, marginalizując wszystkie pozostałe. Jeśli ta cecha jest istotna z perspektywy klasyfikacji obrazu, to jej ekstrakcja znacząco ułatwi zadanie takiej sieci neuronowej.

Jak możemy się domyślać, istnieje bardzo wiele różnych cech obrazu, podobnie jak istnieje wiele wyspecjalizowanych grup neuronów w naszym mózgu. Warstwa splotowa jest na to przygotowana i może zawierać wiele różnych filtrów. Jeden wyostrzy krawędzie, co pozwoli modelowi dostrzec filiżankę. Inny skupi się na kolorach, dzięki czemu dowiemy się, że w filiżance jest kawa, a nie woda lub herbata.

A teraz najlepsze — nie musimy wiedzieć, jakie cechy obrazu nas interesują i jakich wartości filtra powinniśmy użyć do ich ekstrakcji. Nasza sieć może sama się tego nauczyć z oznaczonych przykładów, podobnie jak wcześniej uczyliśmy

perceptron wielowarstwowy. Co więcej, do wyznaczania wag w warstwie konwolucyjnej używa się naszego starego dobrego znajomego — algorytmu wstecznej propagacji błędów.

Powinienem się jeszcze wytłumaczyć z tej zamiany rozmiaru — kanał wejściowy ma 6×6 pikseli, a wyjściowy już tylko 4×4. To efekt uboczny działania warstwy splotowej. Wydaje się spory, bo nasze wejście jest bardzo niewielkie. Dla kanału o rozmiarze 100×100 i filtra 3×3 dostalibyśmy mapę cech o rozmiarze 98×98, a w przypadku filtra 5×5 — 96×96. Chyba już widać, o co chodzi…

***

Niby wszystko fajnie, ale Szanowny Czytelnik ma prawo odczuwać pewien dysonans poznawczy. Wcześniej pisałem, że w prostszych sieciach problemem jest duży wektor wejściowy, zwłaszcza w przypadku zdjęć w wysokiej rozdzielczości. A tu sami, na własne życzenie, serwujemy sobie po kilka czy kilkanaście dodatkowych obrazów (map cech) dla każdego kanału wejściowego. Gdzie tu sens? Gdzie logika?

Okazuje się, że wcale nie musimy kurczowo trzymać się oryginalnych rozmiarów obrazu, żeby móc rozpoznawać, co na nim jest. To poniekąd pozostaje w zgodzie z tym, co podpowiada nam intuicja. Jeśli zmniejszamy rozmiar zdjęcia w dowolnym programie graficznym, to tracimy co prawda na jakości, ale — dopóki nie przesadzimy — bez problemu rozpoznamy widoczne na nim obiekty. Za redukcję rozmiaru poszczególnych kanałów w sieciach konwolucyjnych odpowiada warstwa łącząca, od angielskiego „pooling", występująca naprzemiennie po każdej warstwie splotowej.

Zasada działania jest bardzo podobna. Znowu bierzemy małe jeżdżące okienko, które zamienia całą grupę pikseli z obrazu na wejściu w pojedynczy piksel obrazu na wyjściu. Wejściem jest oczywiście mapa cech, wygenerowana uprzednio w warstwie konwolucyjnej. Wyjście, dla uproszczenia, będziemy nazywać zredukowaną mapą cech. Dobra wiadomość jest taka, że w warstwie łączącej nie stosujemy żadnych skomplikowanych filtrów. Z całej grupy pikseli aktualnie objętych oknem wybieramy albo wartość najmniejszą (*min-pooling*), albo największą (*max-pooling*), albo liczymy średnią (*avg-pooling*).

Ale to nie koniec różnic. W przeciwieństwie do warstwy splotowej tu nasze okno możemy przesuwać o więcej niż jedną pozycję na krok. Sami definiujemy tę wartość. Jeśli przesunięcie, od angielskiego „stride", jest równe długości boku okna, to redukcja rozmiaru obrazu będzie największa. Jeśli jest mniejsze, to okna z kolejnych kroków będą na siebie nachodzić, przez co redukcja będzie odpowiednio mniejsza, aż do minimalnej w przypadku przesunięcia o jedną pozycję.

Wróćmy do mapy cech, uzyskanej uprzednio w warstwie splotowej po zastosowaniu filtra do detekcji krawędzi pionowych. Sprawdźmy, co zrobi z nią warstwa łączna z oknem 2×2, przesunięciem równym 2, oraz avg-poolingiem. Notacja z nawiasami kwadratowymi, identyczna jak poprzednio. Poniżej widzimy wynik działania warstwy po drugim i po ostatnim kroku.

$$\begin{pmatrix} 1 & 23 & [22] & [0] \\ 3 & 23 & [21] & [0] \\ 0 & 22 & 22 & 1 \\ 2 & 25 & 20 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 12{,}5 & [10{,}75] \\ . & . \end{pmatrix}$$

$$\begin{pmatrix} 1 & 23 & 22 & 0 \\ 3 & 23 & 21 & 0 \\ 0 & 22 & [22] & [1] \\ 2 & 25 & [20] & [0] \end{pmatrix} \rightarrow \begin{pmatrix} 12{,}5 & 10{,}75 \\ 12{,}25 & [10{,}75] \end{pmatrix}$$

Tak sobie, prawda? Wszystko się rozmyło — była krawędź, nie ma krawędzi. Może warto spróbować czegoś innego? Może okno 2×2, z przesunięciem równym 1 i min-poolingiem dałoby lepsze rezultaty? Sprawdźmy, dla tych samych kroków co poprzednio — drugiego i ostatniego.

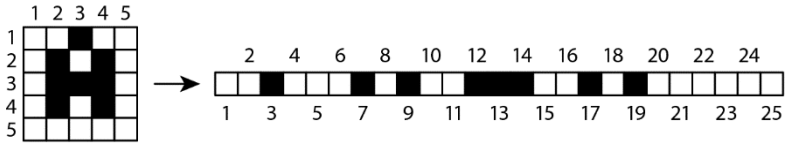
$$\begin{pmatrix} 1 & [23] & [22] & 0 \\ 3 & [23] & [21] & 0 \\ 0 & 22 & 22 & 1 \\ 2 & 25 & 20 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & [21] & . \\ . & . & . \\ . & . & . \end{pmatrix}$$

$$\begin{pmatrix} 1 & 23 & 22 & 0 \\ 3 & 23 & 21 & 0 \\ 0 & 22 & [22] & [1] \\ 2 & 25 & [20] & [0] \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 21 & 0 \\ 0 & 21 & 0 \\ 0 & 20 & [0] \end{pmatrix}$$

Jest lepiej, bez dwóch zdań. Z obrazu 6×6 udało się wyprodukować obraz 3×3 z mocno wyostrzoną krawędzią pionową. Warstwa łącząca, w przeciwieństwie do splotowej, nie mnoży nam już kanałów. Mamy ich dokładnie tyle samo, ile wyszło z warstwy konwolucyjnej, z tą tylko różnicą, że są one teraz znacznie mniejsze.

Jak już wspominałem, warstwy splotowe i łączące możemy stosować naprzemiennie, ale też nic nie stoi na przeszkodzie, żeby użyć kilku warstw konwolucyjnych pod rząd. Wyjście jednej warstwy stanowi wejście dla kolejnej. I tak w kółko, aż będziemy zadowoleni z rezultatów. Czyli aż dostaniemy bardzo wiele malutkich map, z wyekstrahowanymi wszystkim cechami obrazu potrzebnymi do jego skutecznej klasyfikacji. Malutkich, ale na tyle dużych, żeby sieć dała radę rozpoznać, co na nich jest.

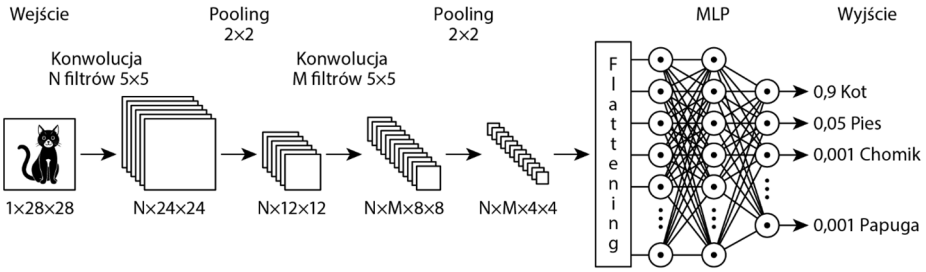
Gdy już skończymy zabawę ze splotem i poolingiem, wracamy do naszego pierwotnego pomysłu — mapowania dwuwymiarowego obrazu do jednowymiarowego wektora. Po angielsku nazywa się to „flattening", więc dla nas niech to będzie „spłaszczanie". Krótko i w punkt. Spłaszczamy wszystkie zredukowane mapy cech, które otrzymaliśmy z ostatniej warstwy łączącej. A kiedy tylko uporamy się z pierwszą, bierzemy się za kolejną, a jej wartości doklejamy dalej, tworząc jeden wielki wektor.



Rysunek 19. Przykład mapowania dwuwymiarowego obrazu do jednowymiarowego wektora

Tu czeka nas kolejny wielki powrót. Nasz nowy wektor stanowi bowiem wejście dla… perceptronu wielowarstwowego. Tak jest — konwolucja zrobiła swoje, więc teraz w końcu przydałoby się wytrenować jakiś klasyfikator. A do tego nie wymyśliliśmy jeszcze nic lepszego niż sieci typu *feed-forward*. W literaturze fachowej taką podsieć i jej warstwy w CNN-ach nazywa się „fully-connected", co możemy przetłumaczyć jako „w pełni połączone".

Uproszczony schemat całej sieci konwolucyjnej, od wejścia aż do wyjścia, prezentuje się następująco:



Rysunek 20. Uproszczony schemat działania konwolucyjnej sieci neuronowej, zawierającej dwie warstwy splotowe i dwie warstwy łączące

Omówmy jeszcze tylko krok po kroku, co tam się właściwie dzieje. Na wejściu mamy obraz o rozmiarze 28×28 pikseli, w odcieniach szarości — stąd tylko jeden kanał. Pierwsza warstwa splotowa składa się z N filtrów 5×5 — stąd N map cech o rozmiarach 24×24. Po niej mamy warstwę łączącą z oknem 2×2 i przesunięciem równym 2 — stąd N map cech, zredukowanych do rozmiaru 12×12. Druga warstwa splotowa zawiera kolejne M filtrów 5×5, produkujących już w sumie N razy M map cech o rozmiarach 8×8, które docelowo zostają zredukowane do 4×4 przez ostatnią warstwę łączącą. Po niej następuje spłaszczenie wszystkich kanałów do pojedynczego wektora i rozpoczyna się klasyfikacja z użyciem sieci typu *feed-forward*. Na jej wyjściu dostajemy wynik i dowiadujemy się, że na zdjęciu na 90% był kot.

I to już w zasadzie wszystko. Sieci konwolucyjne rzeczywiście naśladują sposób, w jaki nasz mózg umożliwia nam widzenie. Zresztą nie tylko nasz — koci też. I żeby było jasne — celowo pominąłem bardzo wiele szczegółów, takich jak choćby stosowanie funkcji aktywacji. Uznałem, że nie jest to niezbędne do zrozumienia istoty rzeczy i mam nadzieję, że Szanowny Czytelnik będzie w stanie mi to wybaczyć. Warto zapamiętać, że pomysł z automatyczną ekstrakcją cech zrewolucjonizował całą dziedzinę *computer vision*, czyli nasze swojsko brzmiące rozpoznawanie obrazów. Szkoda tylko, że wiedza, która nas do tego doprowadziła, okupiona została wielkim cierpieniem zwierząt.

O EKSPERTACH I AUTORYTETACH

Dawno temu, kiedy byłem jeszcze młodym i przystojnym doktorantem na Politechnice Gdańskiej, mojej Alma Mater, chcąc nie chcąc, musiałem prowadzić laboratoria z fizyki. Dla osób, które nie są w temacie — to taki rewelacyjny sposób uczelni na urozmaicenie życia osobom, które z jakiegoś powodu uznały, że doktorat w Polsce wcale nie jest takim złym pomysłem. Dostaniesz co prawda stypendium, za które nie da się wyżyć, ale za to będziesz mógł prowadzić zajęcia, których nikt inny prowadzić nie chce. A nauka? Naukę to ty sobie rób w domu, w swoim czasie wolnym.

Laborki z fizy, bo tak brzmiała nieoficjalna nazwa tego przedmiotu, polegały na tym, że studenci przy swoich stanowiskach wykonywali niespecjalnie ciekawe doświadczenia. Chyba głównie po to, żeby w dorosłym życiu nie opowiadali głupot, że ciężka ołowiana kula spada szybciej niż lekka plastikowa. I żeby, broń Boże, nie przekazywali tych głupot swoim dzieciom. Coś sprawdzali, coś mierzyli, coś liczyli, a ja musiałem udawać, że to wszystko jest bardzo ważne, i sprawdzać, czy cokolwiek z tego rozumieją.

Czasem coś komuś nie poszło i wtedy trzeba było odrabiać całe zajęcia, albo douczyć się i poprawiać tylko część teoretyczno-zagadkową. Przytrafiło się to pewnej studentce, którą po zajęciach spotkałem na schodach, siedzącą w kącie i szlochającą cichutko. Poczułem, że muszę coś zrobić — jakoś pomóc, zareagować. Z moją wrodzoną empatią, nieskończonym zrozumieniem i przemożną chęcią pocieszenia studentki, podszedłem do niej i z uśmiechem na ustach powiedziałem: „Nie martw się, nie każdy musi studiować na politechnice"…

No cóż, stereotypy nie biorą się znikąd. W moim przypadku okazało się, że można nad tym pracować. Więc jeśli Szanowny Czytelnik pomyślał: „No… i w sumie całkiem dobrze jej powiedział", to proszę się nie martwić — przy odrobinie samozaparcia naprawdę można się wyleczyć z bycia dupkiem.

***

Jakiś czas temu zaproszono mnie do podcastu, w którym padło pytanie: „Czy wszyscy, którzy zajmują się sztuczną inteligencją, znają się na tym, co robią? Ile osób tylko pretenduje do bycia ekspertem?". W pierwszej chwili mnie zamurowało. A czy każdy lekarz to dobry lekarz? — pomyślałem. A prawnik? A hydraulik? Dlaczego sztuczna inteligencja miałaby być jakimś wyjątkiem? W każdej branży mamy dobrych ekspertów, kiepskich ekspertów i wannabe ekspertów, czyli pozerów. A tych ostatnich jest tym więcej, im większy jest hype na daną branżę. Szanowny Czytelnik z pewnością pamięta, jak nam obrodziło w ekspertów od wirusów i szczepionek w 2020 oraz w ekspertów od konfliktów zbrojnych i stosunków międzynarodowych w 2022 roku.

Refleksja przyszła dość szybko i udało mi się odnaleźć w tym pytaniu drugie dno. A mianowicie — w jaki sposób osoba niezwiązana z daną branżą mogłaby ocenić eksperckość tej czy innej osoby? Ciężki orzech do zgryzienia, prawda? Mamy autorytety, ale osób, które są niekwestionowanymi i rozpoznawalnymi ekspertami w swoich dziedzinach, jak choćby profesor Jerzy Bralczyk, jest bardzo niewiele. Znacznie częściej proponuje się nam jakieś szerzej nieznane nazwisko, do którego to osoba prowadząca rozmowę dokleja łatkę autorytetu.

I w zasadzie nie byłoby w tym nic złego, gdyby rzetelność dziennikarska pozostawała najwyższą świętością. Jeśli media przedstawiają mi kogoś jako eksperta, to chciałbym wierzyć, że ktoś zadał sobie trud sprawdzenia tej osoby pod kątem jej dorobku i doświadczenia. A co więcej, że zagwarantowano tej osobie pełną swobodę wypowiedzi. Chciałbym, ale niestety widziałem zbyt wielu pseudoekspertów, zapraszanych do mediów tylko po to, żeby uwiarygadniali lub legitymizowali swoim nazwiskiem działania tych czy innych decydentów.

Przypomniałem sobie wtedy o efekcie Dunninga-Krugera. To błąd poznawczy, polegający na tym, że osoby niewykwalifikowane w jakieś dziedzinie mają tendencję

do przeceniania swoich umiejętności, a osoby wysoko wykwalifikowane — wręcz przeciwnie. Niezależnie od dziedziny warto zwracać uwagę na to, w jaki sposób nasz ekspert wypowiada się na skomplikowane tematy. To żaden dowód, ale być może cenna wskazówka: „nie wierzcie nigdy tym, którzy na skomplikowane pytania mają proste odpowiedzi".

Rozmawiając o autorytetach, nie sposób nie wspomnieć słynnego eksperymentu Milgrama, który miał badać skłonność ludzi do ulegania autorytetom właśnie. Badanie powtarzano wielokrotnie, w przeróżnych konfiguracjach, ale zawsze brały w nim udział trzy osoby. Pierwsza to ochotnik — osoba poddawana eksperymentowi, która w wyniku sfingowanego losowania ról zawsze zostawała nauczycielem. Druga to eksperymentator, czyli autorytet — zawodowy aktor, odgrywający rolę naukowca. Ubrany w fartuch laboratoryjny, poważny, budzący zaufanie. Trzecim i ostatnim uczestnikiem badania był pomocnik eksperymentatora, któremu — także w wyniku ustawionego losowania — zawsze przypadała rola ucznia. Najlepiej, jeśli była to osoba sprawiająca miłe wrażenie, trochę ciamajdowata, z lekką nadwagą — archetyp poczciwego księgowego.

Nauczyciel w ściśle określony sposób i pod czujnym okiem eksperymentatora przepytywał ucznia.

Gdy ten ostatni udzielał niepoprawnej odpowiedzi, nauczyciel musiał karać go wstrząsem elektrycznym, którego siła rosła wraz z każdym kolejnym błędem. A przynajmniej tak miało mu się wydawać, gdyż na panelu kontrolnym, z którego korzystał, widniały kolejne wartości napięcia — od 15 V aż do 450 V — wraz z opisową nazwą siły wstrząsu: od „Słaby", przez „Silny", „Bardzo silny", aż do „Niebezpieczeństwo" czy „XXX".

Tak naprawdę uczeń wcale nie był rażony prądem, ale cała sytuacja zaaranżowana była w bardzo sugestywny sposób. Najpierw nauczyciel dostawał prawdziwy, pokazowy wstrząs próbny. Wiedział, że wstrząs należy do kategorii „Słaby", a mimo to odczuwał go jako dość bolesny. Następnie uczestniczył w przywiązywaniu ucznia do krzesła, podłączaniu elektrod, a także miał okazję podsłuchać rozmowę ucznia z eksperymentatorem. Gdy ten pierwszy pytał o bezpieczeństwo, ten drugi zapewniał, że wstrząsy mogą być co prawda bolesne, ale nie powodują uszkodzenia tkanek. W końcu nauczyciel i eksperymentator przechodzili do drugiego pomieszczenia, z którego widzieli i słyszeli przepytywanego ucznia.

Tak rozpoczyna się właściwa część eksperymentu. Nauczyciel zadaje pytanie, uczeń odpowiada. Popełnia kolejne błędy, za co jest karany coraz silniejszymi wstrząsami. Nauczyciel obserwuje reakcje ucznia — najpierw spokojne i stonowane, z czasem coraz bardziej nerwowe, aż po ekstremalne. Zaczynają się prośby o przerwanie eksperymentu, odmawianie odpowiedzi, informowanie o problemach z sercem. Do tego dochodzą okrzyki bólu — coraz częstsze, głośniejsze i bardziej histeryczne. W końcu uczeń przestaje reagować i w sali zapada grobowa cisza.

Eksperymentator zdaje się czuwać nad całym procesem. Reaguje na ewentualne wątpliwości nauczyciela. Jest stanowczy i pewny siebie. Jeśli pojawiają się pytania o bezpieczeństwo, spokojnie powtarza, że proces jest bolesny, ale nie uszkadza tkanek. Jeśli nauczyciel chce przerwać eksperyment, eksperymentator jak mantrę powtarza coraz bardziej stanowcze nakazy: (1) „proszę kontynuować", (2) „eksperyment wymaga, aby kontynuować", (3) „proszę kontynuować, jest to absolutnie koniecznie", (4) „nie masz wyboru, musisz kontynuować". Jeżeli po którymkolwiek z nich nauczyciel decyduje się kontynuować, to przy kolejnej próbie przerwania eksperymentator zaczyna wygłaszać je wszystkie od początku. Dopiero odmowa po czwartym nakazie skutkuje przerwaniem eksperymentu.

I teraz najlepsze — wyniki. W oryginalnym eksperymencie aż 65% badanych „dotarło do samego końca" i zaaplikowało uczniowi najsilniejszy możliwy wstrząs — 450 V, oznaczony jako „XXX". Nawet gdy uczeń przestał jakkolwiek reagować po serii agonalnych jęków i okrzyków, badani kontynuowali rażenie go prądem, bo autorytet twierdził, że eksperymentu nie wolno przerywać i że nie będzie trwałego uszczerbku na zdrowiu. Aż 80% badanych kontynuowało aplikowanie wstrząsów po tym, jak uczeń powiedział, że ma problemy z sercem, i prosił, żeby przerwać eksperyment.

Jeśli udało mi się zainteresować Szanownego Czytelnika, to polecam zapoznać się z innymi wersjami tego eksperymentu. Wiele z nich prowadzi do bardzo ciekawych wniosków. Gdy nauczyciel nie widział ucznia, odsetek badanych aplikujących najsilniejszy możliwy wstrząs wzrastał aż do 93%. Z kolei bezpośrednia bliskość ucznia i konieczność samodzielnego przyciśnięcia jego ręki do elektrody obniżała ten wskaźnik do 23%. Gdy nauczyciel mógł sam wybierać siłę wstrząsu, nigdy nie wychodził poza najlżejszą możliwą kategorię — „Słaby", od 15 V do 60 V. Niezwykle ciekawy jest również fakt, że nauczyciele oszukiwali eksperymentatora,

gdy tylko myśleli, że on tego nie widzi — zamiast wstrząsu silniejszego aplikowali słabszy. Pokazuje to, że badani ulegali autorytetowi wbrew swojej woli.

Założę się, że czytając o eksperymencie Milgrama, Szanowny Czytelnik zastanawiał się, czy w podobnej sytuacji sam uległby autorytetowi, czy też nie. Jeśli dominowała myśl, że „może inni by ulegli, ale na pewno nie ja", to całkiem możliwe, że mamy do czynienia z innym błędem poznawczym — złudzeniem ponadprzeciętności. Jak można się domyślić, chodzi o skłonność jednostki do przeceniania swoich cech lub umiejętności w stosunku do innych ludzi. Czy my przypadkiem nie zatoczyliśmy koła, wracając do efektu Dunninga-Krugera?

I tak, i nie. Złudzenie ponadprzeciętności jest szerszym zjawiskiem. I bardzo powszechnym. Tu moim ulubionym przykładem jest praca „Bias Blind Spot: Structure, Measurement, and Consequences" z 2015 roku. Otóż przebadano 600 rezydentów USA i okazało się, że ponad 85% z nich uważało siebie za mniej podatnych na popełnianie błędów poznawczych niż przeciętny Amerykanin. Aż chciałoby się zaśpiewać za Alanis Morissette: „Isn't it ironic, don't you think?". Oni pewnie też byliby święcie przekonani, że nie raziliby nieszczęśnika prądem pod wpływem autorytetu.

ŻEBY ZROZUMIEĆ REKURENCJĘ, TRZEBA ZROZUMIEĆ REKURENCJĘ

Geekowy suchar w tytule dobrze wróży temu rozdziałowi. Rekurencja, inaczej rekursja, to odwoływanie się funkcji lub definicji do samej siebie. Wyobraźmy sobie pomieszczenie, w którym na przeciwległych ścianach zamontowano lustra. Czasem można spotkać coś takiego w toalecie w hotelu czy restauracji. Obraz, który wtedy powstaje — nieskończony ciąg odbić samego siebie — możemy uznać za przykład rekurencji. Ale po co ja o tym piszę? A po to, że właśnie przechodzimy do kolejnego, bardzo ważnego rodzaju sieci neuronowych. Do sieci rekurencyjnych, z angielskiego „recurrent neural network", powszechnie zwanych RNN-ami. Ale zanim wskoczymy do tej króliczej nory — standardowe ostrzeżenie. Uwaga, trudny rozdział, czytanie „do poduszki" grozi zaśnięciem i koniecznością czytania wszystkiego jeszcze raz od początku!

Szkolnym przykładem rekurencji jest silnia. Jeśli Szanowny Czytelnik nie pamięta, to nic nie szkodzi — zaraz sobie przypomnimy. Silnia jest funkcją liczbową, którą oznaczamy wykrzyknikiem — $n!$ — i definiujemy jako iloczyn wszystkich kolejnych liczb naturalnych od 1 do n. Wypiszmy sobie kilka przykładowych wartości: $1! = 1$, $2! = 2$, $3! = 6$, $4! = 24$, $5! = 120$.

Tylko gdzie tu jest rekurencja? O, tu — możemy zapisać, że $n!$ to tak naprawdę $(n–1)!$ razy n. Czyli funkcja rzeczywiście odwołuje się do siebie samej. Problem napotkamy, dopiero cofając się do 0!, które — jako przypadek graniczny — musimy

zdefiniować niezależnie jako 1. Gdy to zrobimy, możemy zapisać silnię w postaci algorytmu wykorzystującego rekurencję:

```
1:   n! {
2:       jeżeli n jest równe 0, zwróć 1
3:       w przeciwnym razie zwróć n*(n-1)!
4:   }
```

I tak oto posiedliśmy moc rekurencji, która pozwoli nam zrozumieć tytułowego suchara. Jeśli Szanowny Czytelnik ma jeszcze jakieś wątpliwości, to proponuję — w ramach ćwiczenia — rozpisać dla 5! kolejne kroki powyższego algorytmu.

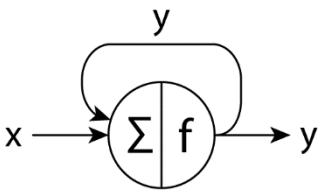
Tak jak sieci konwolucyjne były odpowiedzią na wyzwania związane z przetwarzaniem obrazu, tak sieci rekurencyjne adresują pewne specyficzne problemy związane z przetwarzaniem danych sekwencyjnych. W najprostszych słowach: chodzi o dane, w których istotną rolę odgrywa kolejność elementów, gdzie elementy wcześniejsze wpływają na późniejsze lub odwrotnie. Oczywistym przykładem takich danych jest tekst.

Do tej pory mówiliśmy wyłącznie o najprostszych modelach reprezentacji danych tekstowych, takich jak *bag-of-words* czy n-gramy. Czuliśmy, że coś jest nie tak. Jakbyśmy szli naprawiać reaktor jądrowy za pomocą szarej taśmy i trytytek. Niby do prostych zadań, takich jak analiza sentymentu, to wszystko było „good enough", ale ciężko byłoby się nam z kimkolwiek dogadać, gdybyśmy słyszeli po trzy słowa naraz, a resztę natychmiast zapominali.

A nawet w tych prostych zastosowaniach jak mielibyśmy poprawnie ocenić sentyment takiego zdania: „Zwabieni obietnicą doskonałej kuchni, przepysznych drinków i wspaniałej zabawy, jedyne, co dostaliśmy, to wielkie rozczarowanie"? Wspaniała zabawa? Przepyszne drinki? Doskonała kuchnia? Niby jest tam coś o rozczarowaniu, ale to w końcu 3 pozytywy na 1 negatyw, prawda?

A gdyby tak neuron mógł odwoływać się sam do siebie, jak nasz algorytm liczenia silni? Do tej pory zawsze było tak, że sygnał wejściowy (x) wchodził do bloku sumującego (Σ), przechodził przez funkcję aktywacji (f) i — w postaci sygnału wyjściowego (y) — trafiał na wejście kolejnego neuronu. Dlatego takie sieci nazywaliśmy sieciami

typu *feed-forward*. A gdyby dodatkowo sygnał wyjściowy (y) zataczał pętlę i wchodził ponownie do tego samego neuronu, z którego właśnie wyszedł? Tak jak na poniższym schemacie. Przecież neuron może przyjmować wiele wejść — nie łamiemy tu żadnego ze świętych przykazań.



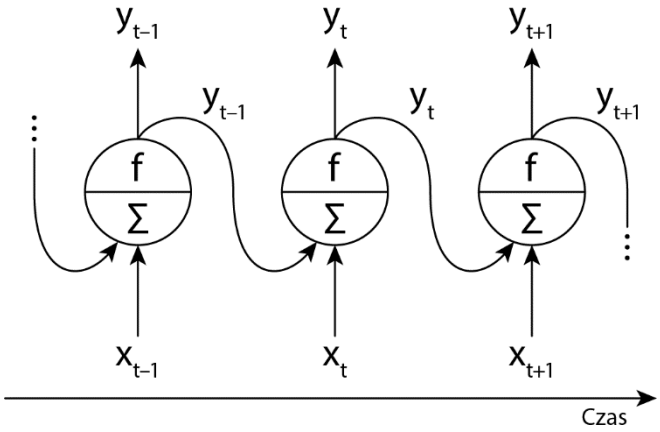
Rysunek 21. Ogólny schemat budowy pojedynczego neuronu rekurencyjnego

Taki neuron będziemy od tej pory nazywać neuronem rekurencyjnym. Tylko że jest z nim pewien problem. Na samym początku nie mamy przecież żadnego sygnału wyjściowego (*y*), bo nasz neuron nie zdążył jeszcze nic zrobić. Dlatego, żeby to wszystko nabrało sensu, musimy wprowadzić krok czasowy, który będziemy oznaczać dolnym indeksem, zarówno w przypadku sygnału wejściowego (x_t), jak i wyjściowego (y_t). I teraz rzeczywiście w pierwszym kroku wszystko będzie po staremu, ale już w drugim na wejście neuronu możemy podać jego wyjście z kroku pierwszego.

Taką sytuację dla trzech kolejnych kroków czasowych przedstawia rysunek 22. Zmieniamy tylko orientację i kierunek przepływu informacji z lewo – prawo na dół – góra. Tak będzie łatwiej, bo rozpatrując dane sekwencyjne, dla każdego neuronu będziemy mieli po kilka kroków czasowych.

Na razie nie jest źle, prawda? Bo nasz rollercoaster dopiero wjeżdża pod górę, na najwyższy punkt konstrukcji. Najlepsze wciąż przed nami. Zapamiętajmy tylko, że rysunek 22 przedstawia pojedynczy neuron. Pętla zwrotna łączy ten sam neuron z dwóch kolejnych kroków czasowych, a nie dwa różne neurony. To bardzo ważne.

Z pojedynczym neuronem niewiele zdziałamy. Znów musimy połączyć je w warstwy i zbudować sieć. I ten proces może okazać się nie lada wyzwaniem dla naszej wyobraźni. Ale spokojnie — fizycy jakoś to rozgryźli. W podręcznikach do szczególnej teorii względności opisuje się czterowymiarową przestrzeń, którą nie do końca



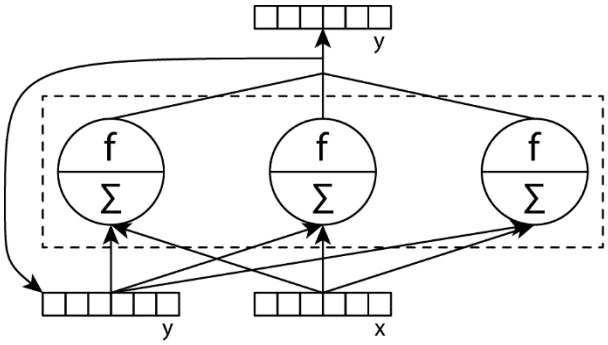
Rysunek 22. Schemat działania pojedynczego neuronu rekurencyjnego w trzech kolejnych krokach czasowych

umiemy sobie wyobrazić, ale wszelkie rysunki najczęściej upraszcza się do dwóch wymiarów — jednego przestrzennego i jednego czasowego. I z tym już zazwyczaj nie mamy problemów. A gdy zrozumiemy podstawowe idee i założenia dla dwóch wymiarów, możemy je w miarę łatwo uogólnić i liczyć dla czterech, ale już bez konieczności wyobrażania sobie czegoś tak skomplikowanego.

A niech tam, sprzedam Szanownemu Czytelnikowi pewien life hack. Najwyżej gildia fizyków i matematyków mnie za to znienawidzi, bo musicie wiedzieć, że to jeden z ich najpilniej strzeżonych sekretów. Gotowi? No to zaczynamy. Jak wyobrazić sobie przestrzeń siedmiowymiarową? Wystarczy wyobrazić sobie przestrzeń N-wymiarową i za N podstawić siedem. Tadam! Ale to nie koniec, bo jest jeszcze duga wersja. Można wyobrazić sobie przestrzeń sześciowymiarową i po prostu dodać jeden wymiar. To takie proste!

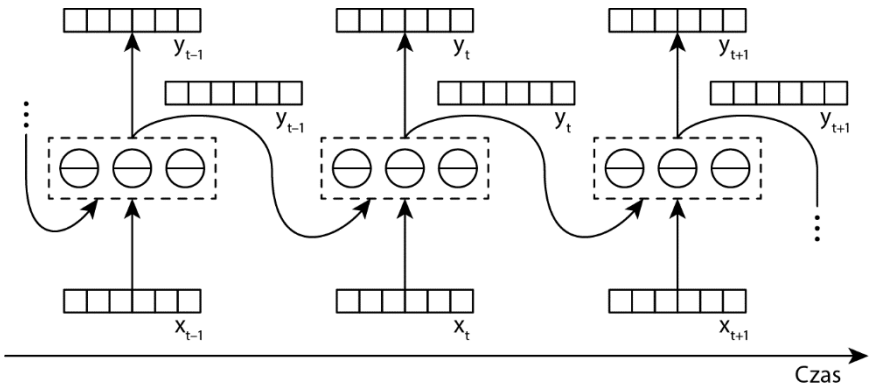
Wracamy do sieci. Czas, który właśnie dodaliśmy, jest takim kolejnym wymiarem. Sprawia, że wejścia i wyjścia neuronu rekurencyjnego przestają być pojedynczymi wartościami, a stają się wektorami, czyli uporządkowanymi zbiorami wartości. Ale przede wszystkim to właśnie czas pozwala na przyjmowanie danych sekwencyjnych — kolejne kroki czasowe odpowiadają kolejnym elementom sekwencji. Co więcej, rekurencyjne sieci neuronowe mogą pracować na sekwencjach o dowolnej długości.

Spróbujmy narysować pojedynczą warstwę sieci z trzema neuronami rekurencyjnymi. Na początek bez czasu, ale zaznaczając, że wejście i wyjście to wektory:



Rysunek 23. Ogólny schemat budowy pojedynczej warstwy sieci rekurencyjnej złożonej z trzech neuronów

A teraz dodajmy czas i rozrysujmy naszą warstwę dla trzech kolejnych kroków czasowych, analogicznie jak zrobiliśmy to wcześniej dla pojedynczego neuronu:



Rysunek 24. Schemat działania pojedynczej warstwy sieci rekurencyjnej złożonej z trzech neuronów w trzech kolejnych krokach czasowych

Możemy powiedzieć, że dodanie czasu i pętli zwrotnej pozwoliło sieci zachowywać stan układu. Czyli pamiętać. Neuron rekurencyjny jest zatem bardzo prostą komórką pamięci — zachowuje wiedzę o tym, co wydarzyło się wcześniej. W naszym przypadku wyjście neuronu jest jednocześnie przekazywanym stanem, ale

nie musi tak być. Bardziej skomplikowane komórki pamięci rozdzielają wyjście i stan układu w danym kroku czasowym, który nazywa się wtedy stanem ukrytym, od angielskiego słowa *hidden*.

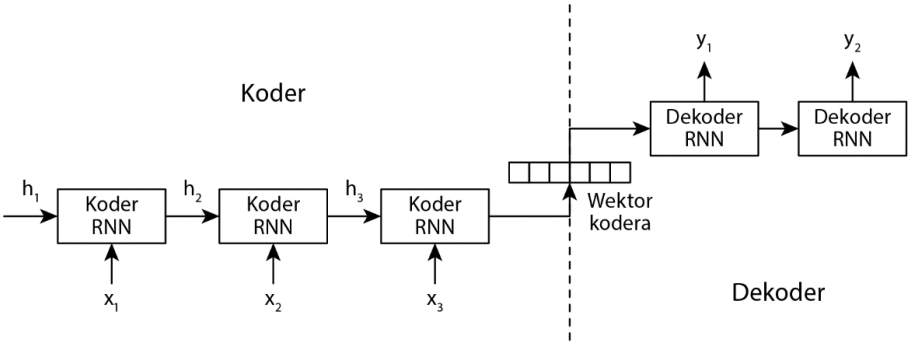
***

Wiemy, że na wejściu do sieci mamy sekwencję elementów, np. kolejnych wyrazów w tekście. Wiemy, że sieć przyjmuje kolejne elementy sekwencji wejściowej w kolejnych krokach czasowych, a informacja o poprzednich elementach jest przekazywana dalej, w głąb sieci, w postaci stanów ukrytych. Jeśli na wyjściu z sieci też chcielibyśmy otrzymywać jakąś sekwencję elementów, to mówimy wtedy o modelu *sequence-to-sequence*.

Nie powinniśmy mieć trudności z wymyśleniem zadań, dla których tekst stanowi jednocześnie wejście i wyjście modelu. Tak działają choćby tłumaczenia maszynowe. Sekwencją wejściową są kolejne słowa tekstu, który mamy zamiar przetłumaczyć, a wyjściową — kolejne słowa tłumaczenia. Podobnie jest w przypadku generowania odpowiedzi na pytania zadane w języku naturalnym. To zadanie niestety nie ma ładnej polskiej nazwy, więc najczęściej korzysta się z angielskiej, mówiąc o question answeringu — w skrócie QA.

Może się tak zdarzyć, że na wejściu lub wyjściu sieci wystarczy nam pojedynczy wektor. Wtedy mówimy o szczególnych przypadkach modelu *sequence-to-sequence*. Jeśli na wejściu mamy tekst, który chcemy poddać klasyfikacji, jak w analizie sentymentu, to możemy użyć modelu *sequence-to-vector*. Gdy na wejściu mamy obraz, dla którego chcemy wygenerować opis w języku naturalnym, to skorzystamy z modelu *vector-to-sequence*. Do modelu *vector-to-vector* raczej nie użylibyśmy sieci rekurencyjnych, ale gdybyśmy się uparli, to moglibyśmy tłumaczyć pojedyncze słowo na inne albo szukać dla niego synonimu.

Najpopularniejszą architekturą realizującą model *sequence-to-sequence* jest koder-dekoder, z angielskiego *encoder-decoder*. W największym skrócie: koder zajmuje się pobraniem i przetworzeniem sekwencji wejściowej, a dekoder — generowaniem sekwencji wyjściowej. Między nimi znajduje się wektor będący końcowym stanem ukrytym kodera, tak zwany wektor kodera.



Rysunek 25. Uproszczony schemat architektury koder-dekoder dla rekurencyjnych sieci neuronowych

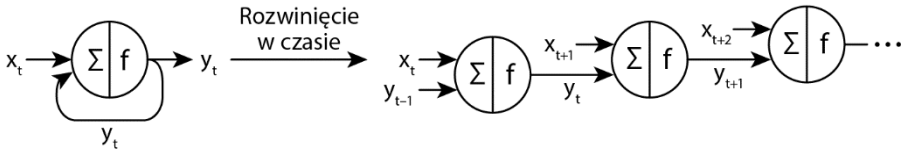
Na wejściu mamy wektory reprezentujące kolejne elementy sekwencji wejściowej (x_1, x_2, x_3), a na wyjściu — wyjściowej (y_1, y_2). I w zasadzie na tym mógłbym poprzestać, ale chciałbym, żeby Szanowny Czytelnik miał szansę tego „dotknąć". A że nie poznaliśmy jeszcze żadnych ciekawszych metod reprezentacji wektorowej słów, użyjmy najprostszej — *one-hot encoding*. Polega ona na tym, że wektor ma tyle elementów, ile jest słów w słowniku, a każde słowo ma swoją ustaloną pozycję. Na tej pozycji znajduje się jedynka, a na wszystkich pozostałych — zera. Gdybyśmy mieli słownik 10-wyrazowy, to moglibyśmy słowa „dom" i „mieszkanie" zapisać za pomocą następujących wektorów:

```
dom: [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
mieszkanie: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
```

Koder i dekoder to tak naprawdę dwie niezależne rekurencyjne sieci neuronowe. Dlatego na powyższym schemacie zostały oznaczone jako „koder RNN" i „dekoder RNN". Warto zaznaczyć, że ich kolejne wystąpienia oznaczają po prostu kolejne kroki czasowe — to są ciągle te same sieci. Pośrednie stany ukryte zostały zgodnie z ogólnie przyjętą konwencją oznaczone literą „h" od „hidden" (h_1, h_2, h_3). A żeby było miło, to za przykład weźmy tłumaczenie maszynowe — 3-wyrazowe wejście to „I love you", a 2-wyrazowe wyjście to „kocham cię".

A teraz pytanie za 100 punktów — w jaki sposób uczymy rekurencyjne sieci neuronowe? To nie jest podchwytliwe pytanie — Szanowny Czytelnik ma szanse na nie

odpowiedzieć. Korzystamy z algorytmu wstecznej propagacji błędów, tak jak w przypadku sieci typu *feed-forward*. Jedyna różnica jest taka, że teraz mamy do czynienia z czasem jako kolejnym wymiarem. Radzimy sobie z tym, rozwijając sieć w czasie. Brzmi to dość skomplikowanie, ale sprowadza się do tego, że musimy „rozprostować" wszystkie pętle zwrotne i zagwarantować taki przepływ informacji jak w sieciach typu *feed-forward*. A zatem rozwałkowujemy kolejne kroki czasowe na płasko, jak ciasto na pierogi. Tak jak na poniższym schemacie.



Rysunek 26. Uproszczony schemat działania algorytmu wstecznej propagacji błędów w czasie

Ta wersja algorytmu uczącego ma swoją nazwę — wsteczna propagacja w czasie, po angielsku „backpropagation through time", a w skrócie BPTT.

***

Na koniec chciałbym jeszcze opowiedzieć co nieco o ograniczeniach rekurencyjnych sieci neuronowych. Ich uczenie odbywa się sekwencyjnie, krok po kroku, co oznacza, że kolejne elementy sekwencji są przetwarzane w kolejnych krokach czasowych. A ponieważ kolejne elementy zależą od poprzednich, to nie bardzo daje się ten proces zrównoleglać, przez co trenowanie modelu jest wolne i może stanowić spore wyzwanie, zwłaszcza w przypadku długich sekwencji.

Proste RNN-y mają jeszcze jedną istotną wadę — pamięć o wcześniejszych wejściach zanika z czasem. I to wykładniczo, czyli bardzo szybko. Zanim model „doczyta" treść do końca, zdąży zapomnieć, co było na początku. Jeśli recenzja zaczyna się od „produkt jest doskonały, ale…", a potem autor wymienia rzeczy, które nie do końca mu się podobały, to taka sieć zignoruje tę pierwszą — kluczową — opinię, i oceni recenzję jako negatywną. Z drugiej strony może to całe zapominalstwo wcale nie jest takie złe, bo ilekroć widzę komentarz zaczynający się od „nie jestem

rasistą/seksistą/homofobem, ale…", to praktycznie od razu wiem, że na 99% ów komentarz będzie rasistowski, seksistowski lub homofobiczny.

Częściowym rozwiązaniem tego problemu są komórki z pamięcią długoterminową, które okazały się na tyle skuteczne, że praktycznie wyeliminowały z użycia komórki podstawowe. Do najpopularniejszych należą komórki LSTM (ang. *Long Short-Term Memory*) oraz ich uproszczona wersja — GRU (ang. *Gated Recurrent Unit*). Nie mylić z Głównym Zarządem Wywiadowczym Sztabu Generalnego Sił Zbrojnych Federacji Rosyjskiej — Gławnoje Razwiedywatielnoje Uprawlenije, w skrócie GRU. Odpuścimy sobie szczegółowe omówienie ich architektur. Są dość skomplikowane, a mam wrażenie, że do szczęścia wystarczą nam główne założenia i idee, które im przyświecają. Jeśli Szanowny Czytelnik poczuje niedosyt, to bez problemu je sobie wygoogluje.

W LSTM-ach zamiast jednego, mamy do dyspozycji dwa stany ukryte — krótko- i długoterminowy. Sprytny system trzech bramek — wejściowej, wyjściowej i „zapomnij" — pozwala sieci na nauczenie się tego, co powinna przechowywać w stanie długoterminowym, co z niego wyrzucić, a co przenieść na wyjście i do stanu krótkoterminowego. Informacja nieznacząca jest zapominana, a istotna — przechowywana i wykorzystywana później, w odpowiednim momencie. Komórka GRU wraca do koncepcji pojedynczego wektora stanu, a uproszczenie polega na tym, że bramka wejściowa i „zapomnij" są sprzężone i działają na zmianę, w myśl zasady, że jeśli coś chcemy przechować, to najpierw musimy zrobić na to miejsce.

I to w zasadzie tyle, jeśli chodzi o główne koncepcje związane z rekurencyjnymi sieciami neuronowymi. LSTM-y na poważnie wystartowały w 2014 roku i były rewolucją, jeśli chodzi o przetwarzanie tekstu. Na kolejną rewolucję nie trzeba było długo czekać, co pokazuje tylko, jak dynamicznie rozwijającą się dziedziną jest sztuczna inteligencja. Zaledwie trzy lata później, w 2017 roku, zespół naukowców z Google Brain opublikował przełomowy artykuł „Attention Is All You Need", który wprowadził nową i bardzo potężną architekturę głębokiego uczenia maszynowego — Transformer. To ona dała nam duże modele językowe, takie jak GPT czy BERT, i to ona króluje dziś niepodzielnie, jeśli chodzi o przetwarzanie języka naturalnego.

O MANIPULACJI, INNOWACJI I TIMINGU

Czy zdarzyło się Szanownemu Czytelnikowi natrafić na reklamę lub opis dowolnego produktu, chwalącego się skutecznością na poziomie 99% czy nawet 99,9%? W przypadku sztucznej inteligencji to niestety w zasadzie standard. Pomyślałem, że to może być dobra okazja, żeby rozprawić się z tą, pożal się Boże, manipulacją. Z reguły takie stwierdzenia nie są niezgodne z prawdą, ale zdarzają się nagminnie i *de facto* nie mówią nam nic o prawdziwej skuteczności produktu.

Wyobraźmy sobie, że mamy do czynienia z detekcją zjawisk rzadkich, takich jak cybernękanie. Powiedzmy, że w rzeczywistym zbiorze danych mamy tego 1%. A w jaki sposób mierzymy skuteczność, po angielsku „accuracy"? Bardzo prosto — jako liczbę poprawnie rozpoznanych wiadomości podzieloną przez liczbę wszystkich wiadomości. Jeśli wiadomość zawiera cybernękanie i system to wykryje, to mamy poprawne rozpoznanie. Ale jeśli wiadomość nie zawiera cybernękania i system powie, że nie zawiera, to — zgodnie z definicją — również mamy poprawne rozpoznanie. To tak, jakbym wydał mojemu kotu polecenie „Bandzior, nic nie rób" i uznawał za sukces to, że Bandzior nic nie robi.

Wróćmy do naszego problemu. Mamy zbiór wiadomości, z których 1% zawiera cybernękanie. Jeśli nasz system nie będzie robił absolutnie nic, to poprawnie rozpoznamy 99% wiadomości niezawierających cybernękania. A to oznacza, że nasz nic-nie-robiący system będzie miał skuteczność na poziomie 99%. Dlatego właśnie stosuje się inne miary, jak choćby precyzję, kompletność i F1, opisywane wcześniej w rozdziale o czarnej skrzynce. Gdy następnym razem Szanowny Czytelnik zobaczy

gdzieś deklarację 99% skuteczności, to mam nadzieję, że przypomni sobie mojego kota, który bardzo skutecznie potrafi nic nie robić na komendę.

Świat sztucznej inteligencji pełen jest manipulacji, przekłamań i niedopowiedzeń. Jak zresztą wszystkie dziedziny życia na styku nauki, technologii i biznesu, gdzie w grę wchodzą naprawdę duże pieniądze. Przykład ze skutecznością jest dość jaskrawy, ale niech Szanowny Czytelnik sam sobie szczerze odpowie na pytanie — który produkt by wybrał? Taki, który ma 99,9% skuteczności, czy taki z 82,1% jakieś mało znanej miary F1?

***

Jednym z największych kłamstw, którymi karmi się młode start-upy jest to, że lepsza technologia czy lepszy produkt zawsze wygrywa. Tu Szanowny Czytelnik znowu może odpowiedzieć sobie szczerze na następujące pytanie: która firma osiągnie sukces? Ta, która przeznacza 70% na badania i rozwój, a 30% na sprzedaż i marketing, czy ta, która robi dokładnie odwrotnie? Odpowiedź nie jest oczywista, ale doświadczenie uczy, żeby raczej obstawiać tę drugą.

Jeśli start-up A ma lepszy produkt, ale „spali" wszystkie środki, zanim dowiezie go do rynku, to prawdopodobnie nie otrzyma kolejnego finansowania od inwestorów, którzy oczekują jednoznacznego potwierdzenia biznesowego w postaci sprzedaży. Dokładnie takiego, o jakim śpiewa Beyoncé w utworze *Single Ladies*:

If you liked it, then you should've put a ring on it

(Jeśli było ci dobrze, trzeba było dać jej pierścionek)

W pozornie lepszej sytuacji będzie start-up B. Jeśli dowiezie gorszy produkt, ale pokaże sprzedaż, to będzie miał szansę na kolejną rundę finansowania. Najprawdopodobniej wpadnie jednak w zupełnie inną pułapkę. Inwestorzy będą oczekiwali stałego wzrostu przychodów, przez co nie będzie miał przestrzeni na krok wstecz i „dociśnięcie techu". Zamiast tego będzie musiał pompować kasę w sprzedaż, żeby spełniać coraz bardziej wyśrubowane wymagania inwestorów. W końcu dotrze do ściany, zapuka w szklany sufit i resztę środków przeznaczy na przygotowanie firmy do tzw. akwizycji, czyli do przejęcia jej przez większego gracza.

Jeśli start-up ma produkt, który robi coś szybciej czy taniej albo lepiej niż konkurencja, to działa na istniejącym rynku, realizując strategię czerwonego oceanu („red ocean strategy"). Kolor czerwony symbolizuje tu krew podmiotów, którym się nie udało. Musi stale weryfikować swoją pozycję na szachownicy i uważać na to, jak pozostali uczestnicy rynku ustawiają się do wiatru. Szuka kompromisu pomiędzy dostarczaną wartością a jej kosztem, wykorzystuje istniejący popyt, a jego celem jest pokonanie konkurencji. Sztandarowym i szalenie ilustratywnym przykładem firmy, która osiągnęła globalny sukces w ramach czerwonego oceanu, jest McDonald's. Myślę, że nie trzeba tego jakoś specjalnie tłumaczyć…

A jeśli start-up ma produkt, który zmienia paradygmat — pozwala robić coś, co wcześniej nie było możliwe, to realizuje strategię niebieskiego oceanu („blue ocean strategy"). Tworzy nowy rynek i nowy popyt, marginalizując konkurencję. Ale to wcale nie oznacza, że jest to droga szybka, łatwa i przyjemna. Musi przebijać się przez ścianę przyzwyczajeń i zasobów wpompowanych już w realizację starego paradygmatu. Musi budować swoją wiarygodność, żeby móc przekonać kogokolwiek do nowego rozwiązania. Musi dotrzeć do wizjonerów, którzy mu uwierzą i zdecydują się podjąć ryzyko przejścia „na nowe". I to wizjonerów na tyle dużych i rozpoznawalnych, żeby za nimi chcieli podążyć kolejni. Tu, w kontrze do McDonald's, najczęściej wymienia się firmę Apple i wprowadzenie na rynek iPoda, który na zawsze zmienił sposób, w jaki kupujemy i słuchamy muzyki.

Posłużę się tu przykładem Samurai Labs, gdzie staramy się realizować strategię niebieskiego oceanu. Status quo czy też aktualnie obowiązujący paradygmat to reaktywna moderacja treści, z wąskim gardłem w postaci ludzkich moderatorów, wspieranych co prawda przez sztuczną inteligencję, ale podejmujących decyzję po fakcie, kiedy krzywda zdążyła się już wydarzyć. Na najgroźniejsze zjawiska, takie jak ideacje samobójcze, najczęściej w ogóle przymyka się oko, tłumacząc się tym, że platforma nie ponosi odpowiedzialności za tego typu treści tak długo, jak długo nie ma wiedzy, że takie treści zostały na niej opublikowane. Czyli lepiej jest udawać, że tematu nie ma, żeby przypadkiem się o nim nie dowiedzieć.

Druga skrajność to w zasadzie otwarta cenzura. Albo przestaniesz używać słów, których nie lubią algorytmy, albo zostaniesz zbanowany, a w najlepszym razie obetniemy ci zasięgi. I o ile w przypadku wulgaryzmów dałoby się to jeszcze jakoś uzasadnić, o tyle zakazywanie słów takich jak „samobójstwo", „homoseksualizm"

czy „aborcja" skutkuje rażącym ograniczeniem wolności wypowiedzi na ważne społecznie tematy. Wylewamy dziecko z kąpielą i trudno się temu dziwić, jeśli tylko spojrzymy na wyliczenia serwisu Spider's Web z 2023 roku, które wskazują, że w największych serwisach społecznościowych w Polsce na 300 tysięcy użytkowników przypada średnio… jeden moderator.

Nasze rozwiązanie to automatyczna i proaktywna ochrona społeczności internetowych, która do każdej z nich jest dostosowywana indywidualnie, bez cenzury. Pozwala otoczyć opieką osoby w kryzysie, rozważające samookaleczanie się lub samobójstwo, a także reagować w czasie rzeczywistym na wszelkie przypadki cybernękania i napastowania. W dodatku holistycznie, nie tylko kijem, ale i marchewką, z nastawieniem na redukcję poziomu cyberprzemocy w całej społeczności.

Nie dość, że brzmi to trochę jak science fiction, to jeszcze spotyka się z poniekąd zrozumiałym oporem ze strony zespołów Trust & Safety, odpowiedzialnych za rozwiązywanie tego typu problemów. Samobójstwa, jak już ustaliliśmy, to absolutne tabu — lepiej nie wiedzieć. Z kolei osoby zajmujące się moderacją treści nie do końca przychylnie podchodzą do rozwiązań, które w znacznym stopniu automatyzują ich pracę. Na niewiele zdają się tłumaczenia, że nie chcemy ich zastąpić, a wręcz przeciwnie — dać im do ręki supermoce, jak superbohaterom. Żeby zamiast nurkować w tym morzu hejtu, mogły skupić się na pozytywnym wzmacnianiu i rozwijaniu swoich społeczności. Niech Szanowny Czytelnik mi wierzy — jak nikt na świecie rozumiemy powiedzenie, że lepsze jest wrogiem dobrego.

***

Czasem na papierze wszystko się zgadza, firma dysponuje odpowiednimi zasobami i technologią, a nie zagra coś zupełnie innego. Być może Szanowny Czytelnik pamięta produkt o nazwie Google Glass. Były to „smart" okulary. Coś jak „smart" zegarki, ale do noszenia na nosie. Z kamerami, bajerami, rozpoznawaniem głosu i obrazów, mogące wyświetlać nam przed oczyma przydatne informacje podczas wykonywania codziennych czynności. Mokry sen fanów filmów o Jamesie Bondzie. Google chciało tym produktem wypłynąć na niebieski ocean „wearables", czyli urządzeń do noszenia. Próbowało powiązać okulary z projektantami modowymi,

reklamowało je jako produkt luksusowy. Google Glass weszły na rynek w 2014 roku i wyleciały z niego z hukiem po niecałym roku. Co poszło nie tak?

Na to pytanie nie ma jednej prostej odpowiedzi. Jedni wskazywali wysoką cenę i niską żywotność baterii, inni podawali w wątpliwość sens korzystania ze „smart" okularów, podczas gdy każde z nas ma w kieszeni lub torebce dużo wygodniejsze urządzenie — smartfon. Można go w każdej chwili wyjąć, wygodnie poszukać potrzebnych informacji, a nawet zrobić zdjęcie czy nagrać film. I to w dużo lepszej jakości niż okularami.

Jeszcze inni podnosili kwestie związane z bezpieczeństwem i prywatnością. Kamera przy samej twarzy, stale „patrząca" na to samo co my? Ja podziękuję. Część barów i restauracji też podziękowała, zabraniając posiadaczom okularów wchodzenia do lokali i korzystania z ich usług. I trudno im się dziwić, bo komfort i poczucie bezpieczeństwa klientów powinny być ich najwyższym priorytetem. Co ciekawe, użytkownicy Google Glass, odmawiający zdjęcia okularów, zarówno w prywatnych rozmowach, jak i publicznych wydarzeniach, dorobili się nawet nowego, wdzięcznie brzmiącego określenia — „glassholes", będącego połączeniem słów „glass" i „asshole".

Start-upowi guru zapytani o trzy najważniejsze czynniki mające wpływ na sukces lub porażkę najczęściej odpowiadają: timing, timing i jeszcze raz timing. Chodzi o czas, w którym firma działa i wchodzi ze swoim produktem na rynek. Dlaczego to takie ważne? Już tłumaczę. Co takiego wydarzy się, jeśli wpadniemy na absolutnie przefantastyczno-bombastyczny pomysł, ale nie będzie jeszcze istniała technologia potrzebna do jego realizacji? Odpowiadam — nic. Nic się nie wydarzy. Perceptron, czyli sztuczny neuron, będący podstawą współczesnych sieci neuronowych, został wynaleziony przez psychologa Franka Rosenblatta w 1958 roku. Przeszło pół wieku musieliśmy czekać, aż komputery będą dysponowały mocą obliczeniową pozwalającą zrobić z perceptronów realny użytek. Ale gotowość technologiczna to tylko jeden z aspektów timingu.

Drugi to impuls gospodarczy. Chodzi o przełomowe momenty lub całe okresy w historii świata, kiedy coś, co wcześniej było drogie i mało dostępne, staje się tanie i powszechne. Albo na odwrót. Znowu możemy nawiązać do mocy obliczeniowej komputerów. Zgodnie z prawem Moore'a ekonomicznie optymalna liczba

tranzystorów w układach scalonych rośnie wykładniczo. A to oznacza coraz szybsze i coraz tańsze procesory. I to przede wszystkim tania moc obliczeniowa otworzyła okno możliwości dla rewolucji smartfonowej, która bezpowrotnie zmieniła świat, wsadzając nam do kieszeni urządzenie, które 20 – 30 lat temu nazywalibyśmy superkomputerem i które zajmowałoby cały pokój.

Trzecim, nie mniej ważnym aspektem jest akceptacja kulturowa. Instagram nigdy nie osiągnąłby takiego sukcesu, gdyby nie Facebook czy różnego rodzaju blogi i vlogi, które przez lata zmieniały normy kulturowe i związane z nimi zachowania. Robienie sobie „selfików" i zdjęć jedzenia w restauracjach, a następnie wrzucanie ich na social media? Relacjonowanie prywatnych wydarzeń w internecie? Coś, co jeszcze kilkanaście lat temu mój tata nazywał ekshibicjonizmem społecznym, dziś jest powszechnie akceptowane, a wśród pokolenia Z niemalże obowiązkowe.

O tym, jak ważny jest timing, przekonaliśmy się z moimi wspólnikami w 2012 roku, na długo przed Samurai Labs. Jako zupełnie inny podmiot oferowaliśmy polskim serwisom internetowym rozwiązanie adresujące problem szeroko pojętej mowy nienawiści. Wtedy zderzyliśmy się ze ścianą i argumentami w stylu „to dobrze, jak ludzie się kłócą i obrażają, bo to zwiększa wyświetlenia i klikalność". Tu na szczęście nie trzeba było czekać pół wieku. Wystarczyła dekada, żeby niemal wszyscy zrozumieli, jak groźna jest cyberprzemoc i do jak poważnych konsekwencji prowadzi bagatelizowanie problemu.

EMBEDDING, CZYLI O TYM, ŻE ZNACZENIE MA ZNACZENIE

Embedding, z angielskiego „osadzanie", to reprezentacja wektorowa zmiennych dyskretnych. Dyskretnych, czyli nieciągłych, a nie takich, które potrafią dochować tajemnicy i nie wtykają nosa w cudze sprawy. Jak już ustaliliśmy, sieci neuronowe na wejściu potrzebują liczb. Gdy mamy do czynienia z czymkolwiek innym, np. z tekstem, to musimy to coś sprytnie zmapować do wektora, czyli do uporządkowanego zbioru liczb, najlepiej rzeczywistych. A zatem termin „word embedding" będzie oznaczał reprezentację wektorową słów.

Modele omawiane w poprzednich rozdziałach, takie jak *bag-of-words* czy n-gramy, stanowią najprostsze możliwe reprezentacje wektorowe tekstu. Nie mówią nam nic o samych słowach, poza tym, jak często dane słowo czy kilka kolejnych słów, wystąpiło w analizowanej treści. Dla przypomnienia, w modelu *bag-of-words* każde słowo ma swoją ustaloną pozycję w wektorze, a ogólna nazwa metody mapującej zmienne dyskretne do postaci takich wektorów to *one-hot encoding*. Słowa „dom" i „mieszkanie" w 10-wyrazowym słowniku moglibyśmy zapisać za pomocą następujących wektorów:

```
dom: [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
mieszkanie: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
```

Taka reprezentacja nie powie nam, że „dom" i „mieszkanie" są do siebie podobne, ani że „król" i „królowa" są w podobnej relacji jak „mąż" i „żona". Do tego

potrzebowalibyśmy uwzględnić znaczenie tych słów. A znaczenie to… Nie, nie, spokojnie, skomplikowane rozważania i definicje zostawimy naukowcom zajmującym się semantyką i lingwistyką kognitywną. A uwierzcie mi — istnieją całe opasłe tomiszcza traktujące o tym zagadnieniu. Nam, przynajmniej na początek, wystarczy reprezentacja znaczeniowa, która w dowolny sposób odwzoruje relacje między słowami, takie jak choćby ich podobieństwo.

To oczywiście nie jest nowa potrzeba, a każdy kamień w bucie, o ile jest wystarczająco upierdliwy, w końcu znajdzie śmiałka zdeterminowanego, by go wyjąć. W 1985 roku na Uniwersytecie Princeton powstał projekt o nazwie Wordnet, którego celem było zbudowanie ogromnej leksykalno-semantycznej bazy danych dla języka angielskiego. Wordnet to coś tam synsety bla, bla synonimicznych jednostek leksykalnych coś tam, coś tam, bla, bla. No przecież nie będę Szanownego Czytelnika torturował tym wszystkim…

Wordnet to potężny zasób, budowany i utrzymywany ręcznie, wymagający naprawdę dużych nakładów ludzkiej pracy. Jest ogromny. Możemy bez problemu znaleźć w nim wszystkie znaczenia słowa „dom". A także synonimy, antonimy, hiponimy i hiperonimy. A do tego oczywiście troponimy, meronimy i wszystkie inne onimy, jeśli tylko istnieją. Poszczególne części mowy są zorganizowane w przeróżne hierarchiczne struktury. Inaczej czasowniki, inaczej rzeczowniki, a jeszcze inaczej przymiotniki i przysłówki. Jeśli ktoś wie, co robi i czego szuka, to z dużym prawdopodobieństwem w Wordnecie to znajdzie.

Wordnet na przestrzeni lat dorobił się wielu wersji językowych, w tym polskiej, o swojsko brzmiącej nazwie „Słowosieć". Mam jednak wrażenie, że jego główną bolączką jest fakt, że był robiony „przez ludzi dla ludzi", albo nawet gorzej — „przez lingwistów dla lingwistów". Nie był tworzony z myślą o uczeniu maszynowym i o ile nadal znajduje swoje niszowe zastosowania, to jednak wypadł z mainstreamu, a lata świetności ma już dawno za sobą.

Pomijając aspekty czysto techniczne, związane z użytkowaniem czy integracją, Wordnet ma dwie wady, które ciężko dziś przełknąć. Po pierwsze nie daje żadnej możliwości uwzględnienia kontekstu. Jest w końcu ręcznie tworzoną bazą danych, opracowywaną dla całego języka. Całego! Dostajemy więc wszystkie możliwe znaczenia i relacje, ale to, którego z nich powinniśmy użyć w takim czy innym przypadku, zależy właśnie od kontekstu. A tego nie ma jak dostarczyć.

Drugim problemem jest ograniczony zakres tego, co możemy dalej zrobić z tymi wszystkimi słowami, znaczeniami i relacjami. Owszem, możemy znaleźć synonimy słowa „stół", a nawet dowiedzieć się, że należy do „mebli", do których należą również „szafa" i „krzesło". Ale nie możemy obliczyć, jak bardzo dwa słowa są do siebie podobne, ani dodać czy odjąć reprezentujących je wartości. Moment, chwila, a po jaką cholerę mielibyśmy to robić? I to jest bardzo dobre pytanie!

Pierwszy problem wydaje się oczywisty, ale drugi sprawia wrażenie wymyślonego na siłę, prawda? Przed embeddingami raczej nikt nie zaprzątał sobie głowy tym, żeby móc dodawać i odejmować znaczenia, i sprawdzać, jakim innym słowom odpowiadają. To dostaliśmy w zasadzie za darmo, z dobrodziejstwem inwentarza. I mimo że nie korzystamy z tej zdolności bezpośrednio, to właśnie embeddingi, jako reprezentacja znaczeniowa, otworzyły drogę dla kolejnych rewolucji w przetwarzaniu języka naturalnego i praktycznie wymiotły całą konkurencję z wszelkich „poważnych" zastosowań.

Jak już ustaliliśmy, na co dzień raczej nie dodajemy ani nie odejmujemy znaczeń słów. Nie korzystamy z tego, trenując modele uczenia maszynowego. Ochoczo jednak sięgamy po tę zdolność, chcąc wyjaśnić, czym tak naprawdę są embeddingi. Na początek zapomnijmy o „jak" i przyjmijmy, że każdemu słowu ze słownika przypisujemy pewien wektor z N-wymiarowej przestrzeni. Zdaję sobie sprawę, że N-wymiarowa przestrzeń może budzić grozę, ale w tym momencie to naprawdę nie ma żadnego znaczenia.

Szanowny Czytelnik, jeśli tylko ma ochotę, może za N podstawić 2 i sprowadzić cały problem do prostych zadań z matematyki na poziomie szkoły podstawowej. Wtedy każde słowo będzie opisywane przez parę liczb rzeczywistych. Chodzi tylko o to, że jak mamy dwa wektory, to zawsze możemy obliczyć, jaka jest między nimi odległość. Możemy to zrobić dla dowolnego N. I okazuje się, że jakimś dziwnym trafem wyrazy podobne znajdują się blisko siebie, a odległość między nimi mówi nam, jak bardzo są do siebie podobne.

Jeśli Szanowny Czytelnik miał kiedyś okazję oglądać mapę nieba, to mniej więcej tak moglibyśmy sobie wyobrazić bardzo wiele słów naniesionych na 2-wymiarową przestrzeń. Zobaczylibyśmy mnóstwo grup, większych i mniejszych, czasem bardzo spójnych, czasem „rozlanych" i przechodzących w inne grupy. Synonimy

znajdowałyby się bardzo blisko siebie. Nieco dalej moglibyśmy zobaczyć wyrazy podobne, tworzące wraz z ich synonimami zupełnie nowe grupy. Gdybyśmy dobrze poszukali, moglibyśmy nawet znaleźć grupę „mebli", w której z pewnością odnaleźlibyśmy „stół", „krzesło" i „szafę", ale bez narzuconych nam z góry łatek czy kategorii.

Ale o co chodzi z tym dodawaniem i odejmowaniem znaczeń? Kto w 2013 roku zajmował się sztuczną inteligencją, ten wie, że w tamtym czasie na niemal każdej konferencji branżowej, na każdym forum, w każdym czasopiśmie, serwisie i portalu do znudzenia wałkowano ten sam przykład:

king (król) + woman (kobieta) - man (mężczyzna) = ?

Chodziło o możliwość wykonywania takich prostych operacji na nowych reprezentacjach wektorowych — embeddingach. Ale to jeszcze nic. Okazało się, że dla powyższego przypadku dostajemy nowy wektor, który odpowiada słowu „queen" („królowa")…

I to było coś. Proste i efektowne jak diabli. Mieszkałem wtedy w Dolinie Krzemowej i wspominam te kilkanaście tygodni jako „ten magiczny czas", kiedy sztuczna inteligencja ubiera się w modny garnitur i wychodzi na miasto. Nagle całkiem poważni ludzie całkiem poważnie dyskutowali o AI, która w końcu rozumie ludzki język. Za tym wszystkim szły duże oczekiwania i duże inwestycje. Niemal codziennie serwowano nam nowe ciekawostki, jak choćby:

paris (paryż) - france (francja) + poland (polska) = warsaw (warszawa)

california (kalifornia) - state (stan) + country (państwo) = america (ameryka)

brother (brat) - man (mężczyzna) + woman (kobieta) = sister (siostra)

***

Za embeddingami stoi dość prosta, a jednocześnie bardzo potężna obserwacja — wyrazy występujące w podobnym kontekście mają podobne znaczenie. A jeśli tak, to wyrazy mogą być definiowane za pomocą innych wyrazów, występujących w ich otoczeniu. Zastanówmy się nad tym. Jeśli dwie różne rzeczy można kupić, przeczytać i odłożyć na półkę, a do tego obie mają okładkę, treść i autora, to prawdopodobnie są to bardzo podobne rzeczy, takie jak książka i podręcznik.

Wyobraźmy sobie, że mamy ogromny korpus, w którym używane na co dzień wyrazy występują po kilka czy nawet kilkanaście tysięcy razy. Wybierzmy teraz dwa wyrazy, A i B. Dla każdego wystąpienia A wypisujemy z korpusu jego najbliższe otoczenie — do pięciu wyrazów z lewej i tyle samo z prawej strony. To samo robimy dla B, a następnie porównujemy oba zbiory. Im więcej znajdziemy tam identycznych wyrazów, tym większe jest podobieństwo znaczeniowe A i B. I na tym właśnie bazują embeddingi.

A jeśli Szanowny Czytelnik nadal mi nie wierzy, to proponuję następujące ćwiczenie. Potrzebujemy trzech rzeczowników. Pierwszy niech będzie dowolny, drugi do niego bliskoznaczny, a trzeci — kompletnie niepowiązany. I teraz, jak w podstawówce, dla każdego z tych rzeczowników wymyślamy po 10 zdań z jego użyciem. Zapisujemy je i sprawdzamy. Widać? Widać. A jak nie widać, to prosimy o wykonanie tego zadania kogoś, kto nie będzie próbował za wszelką cenę udowodnić autorowi, że się myli…

Najpopularniejszą metodą tworzenia embeddingów jest Word2vec. Nazwa jest deskryptywna, a oznacza ni mniej, ni więcej, że będziemy reprezentować wyrazy (words) za pomocą wektorów (vectors). Metodę opracował i opublikował w 2013 roku zespół Tomáša Mikolova z Google'a. Word2vec jest proste i genialne zarazem, co stanowi moje ulubione połączenie. Jest to prosta sieć neuronowa, z jedną warstwą ukrytą, którą trenuje się do przewidywania brakującego słowa na podstawie otaczających go wyrazów, czyli kontekstu.

Wykorzystuje się w tym celu lekko zmodyfikowaną wersję modelu *bag-of-words*. Bierzemy wyraz, który mamy zamiar zgadywać, a wraz z nim okno kontekstowe, obejmujące jeszcze kilka otaczających go z obu stron wyrazów. Następnie idziemy krok w prawo, do kolejnego wyrazu, przesuwając jednocześnie okno kontekstowe o jedną pozycję. Trochę przypomina to jeżdżące filtry, z którymi mieliśmy do czynienia, omawiając konwolucyjne sieci neuronowe. Ze względu na to przesuwające się okno zmodyfikowany model nosi nazwę *continuous bag-of-words*, czyli ciągły, w skrócie CBOW. Jestem przekonany, że poniższy przykład rozwieje wszelkie wątpliwości.

Wracamy do naszego cytatu z Einsteina:

Zdrowy rozsądek to zbiór uprzedzeń nabytych do osiemnastego roku życia.

Zacznijmy od ponazywania rzeczy. Word2vec oparty na CBOW służy do przewidywania *brakującego_wyrazu* przy użyciu *okna_kontekstowego*, czyli określonej liczby wyrazów otaczających *brakujący_wyraz*. Możemy taką parę zapisać w następujący sposób:

([*okno_kontekstowe*], *brakujący_wyraz*)

Jeśli *okno_kontekstowe* będzie miało rozmiar 4, czyli 2 wyrazy z lewej i 2 wyrazy z prawej strony, to z naszego cytatu możemy wypisać następujące pary:

1: ([*Zdrowy*, *rozsądek*, *zbiór*, *uprzedzeń*], *to*)

2: ([*rozsądek*, *to*, *uprzedzeń*, *nabytych*], *zbiór*)

3: ([*to*, *zbiór*, *nabytych*, *do*], *uprzedzeń*)

4: ([*zbiór*, *uprzedzeń*, *do*, *osiemnastego*], *nabytych*)

5: ([*uprzedzeń*, *nabytych*, *osiemnastego*, *roku*], *do*)

6: ([*nabytych*, *do*, *roku*, *życia*], *osiemnastego*)

Te pary, uprzednio zmapowane do wektorów, stanowią dane uczące dla Word2vec. Procedura wygląda tak, że najpierw trenujemy prostą sieć neuronową typu *feed-forward*, z pojedynczą warstwą ukrytą, do przewidywania brakującego słowa na podstawie zadanego kontekstu, a następnie bierzemy z niej tylko warstwę ukrytą. Tak jest — resztę modelu wyrzucamy do kosza. Z warstwy ukrytej wyciągamy nasze embeddingi, czyli wektory dla poszczególnych słów, które — jak się okazuje — niosą niezwykle przydatną informację o ich semantycznych i syntaktycznych własnościach, wyuczonych na podstawie zadanego korpusu. To trochę tak, jakbyśmy zamówili pizzę, odebrali ją od kuriera, zapłacili, a następnie wyrzucili jedzenie do kosza na śmieci, ciesząc się z kartonowego pudełka. Genialne!

Oczywiście można robić to zupełnie inaczej — chodzi tylko o to, żeby na koniec dnia dla każdego słowa dostać wektor reprezentujący jego znaczenie. Najpopularniejszą alternatywą dla Word2vec jest GloVe, którego pełna nazwa brzmi Global Vectors for Word Representation. Projekt jest opensource'owy i rozwijany przez Stanford University od 2014 roku. GloVe może pracować na tym samym korpusie co Word2vec. Wyprodukuje nam wtedy bardzo podobne wektory. Darujmy sobie jednak szczegóły, bo musielibyśmy użyć całej masy brzydkich słów, takich jak macierz współwystępowania, rozkład macierzy czy minimalizowanie funkcji straty.

Na sam koniec zostawiłem jeszcze temat liczby wymiarów. Z reguły embeddingi mają ich kilkaset. To znaczy, że każde słowo jest reprezentowane przez wektor składający się z kilkuset liczb rzeczywistych. Czy to dużo, czy mało? To oczywiście względne, ale proponuję wykonać w myślach ćwiczenie, które pozwoli nam poczuć, z czym mamy do czynienia. Powiedzmy, że zajmujemy się mieszkalnictwem. Mamy listę mieszkań w danym mieście wraz z kompletem informacji na ich temat. Zaznaczamy je punktami na wykresie, żeby w ten sposób odkryć skomplikowane zasady rządzące rynkiem nieruchomości.

Zaczynamy od wykresu dwuwymiarowego — na jednej osi jest cena, a na drugiej metraż. Zaznaczamy mieszkania i ukazują się nam pierwsze zależności, np. im większy metraż, tym wyższa cena nieruchomości. Teraz dodajemy trzeci wymiar — odległość od centrum. Ujawniają się kolejne zależności, np. im bliżej centrum, tym drożej. Ale to nie wszystko. Możemy zacząć dostrzegać, że pewne grupy mieszkań, odpowiadające preferowanym osiedlom czy dzielnicom, są droższe od innych. Dodając kolejny wymiar — odległość od metra — możemy zauważyć, że pewne mieszkania, mimo że daleko od centrum, utrzymują wysoką cenę.

Możemy dodawać kolejne wymiary — odległość od przystanków tramwajowych i autobusowych, obwodnicy, wysypiska śmieci, rok budowy i datę ostatniego remontu, piętro, na którym znajduje się mieszkanie, i informację, czy budynek posiada windę. To wciąż zaledwie kilka czy kilkanaście wymiarów, a już całkiem nieźle opisują nam rynek nieruchomości, prawda? Podobnie jest z embeddingami. Wyobraźmy sobie, że mamy 500 wymiarów i każdy z nich opisuje jakąś nienazwaną cechę słowa. Chyba nietrudno uwierzyć, że w ten sposób możemy całkiem nieźle uchwycić jego znaczenie.

O KIJU I MARCHEWCE

Najpopularniejszym narzędziem w moderacji treści w internecie jest kij. Oczywiście nie dosłownie. Chodzi o zestaw kar, wymierzanych tym, którzy łamią zasady danej platformy, serwisu lub społeczności. A jest z czego wybierać. Można pogrozić palcem, czyli wysłać ostrzeżenie w stylu „jeszcze raz, to zobaczysz". Można usunąć obraźliwą wiadomość. Można delikwenta wyciszyć, a można dać mu tzw. *shadow mute'a*, co osobiście uważam za jeden z najbardziej diabolicznych wynalazków ludzkości. Chodzi o to, że użytkownik zostaje wyciszony, ale nie jest o tym w żaden sposób informowany. Czyli gada sobie w najlepsze przez kolejne piętnaście minut i absolutnie nikt tego nie widzi. No geniusz, geniusz zła!

Jeśli nic innego nie pomaga, to można w końcu niereformowalnego użytkownika zbanować — czasowo lub permanentnie. To oczywiście ostateczność, bo w praktyce pozyskanie użytkownika zawsze stanowi pewien koszt. Można nawet policzyć, jak wysoki, porównując wydatki na reklamę z liczbą nowo zakładanych kont. A zatem platformy i serwisy starają się unikać tej kary ostatecznej. Ba, nierzadko zdarzają się sytuacje, w których przymyka się oko na niektóre zachowania użytkowników z dłuższym stażem lub płacących w tzw. modelu freemium, czyli np. w grach *free-to-play* z mikrotransakcjami.

Skuteczność tego metaforycznego kija pozostaje kwestią dyskusyjną. Osoba wyciszona może ochłonąć, a może wrócić jeszcze bardziej sfrustrowana i agresywna. Osoba zbanowana może dać sobie spokój, a może założyć trzy nowe konta i robić trzy razy więcej syfu. Alternatywą dla kija jest marchewka, która może być bardzo różnie rozumiana i implementowana. Ja chciałbym skupić się na próbie przekonania użytkownika do zmiany zachowania lub sposobu komunikacji.

Inspiracją dla naszej pracy w Samurai Labs był artykuł *Tweetment Effects on the Tweeted: Experimentally Reducing Racist Harassment*, popełniony w 2017 roku przez Kevina Mungera, amerykańskiego naukowca, badającego media społecznościowe i ich wpływ na komunikację polityczną. Chodziło z grubsza o to, żeby zbadać, jak sankcje społeczne, czyli reakcje społeczności na postępowanie jednostki, mogą wpływać na cybernękanie na tle rasistowskim, a także jakie czynniki wpływają na skuteczność tych sankcji.

Eksperyment został przeprowadzony na Twitterze. To taka bardzo popularna platforma, którą jakiś czas temu przejął Elon Musk. Głównie po to, żeby propagować wolność słowa, zwalniając przy tym 80% inżynierów pracujących przy Trust & Safety, oraz po to, żeby dobrą i rozpoznawalną nazwę zastąpić słabą i nierozpoznawalną, czyli X. Główny skutek jest taki, że teraz gdy ktoś mówi o tej platformie, zazwyczaj używa obu nazw naraz — „X (dawniej Twitter)" lub „Twitter (teraz X)". Ot, taka dygresja.

Kevin Munger zaprojektował interwencję — wiadomość, wysyłaną do białych mężczyzn, którzy w swoich tweetach, czyli wiadomościach publikowanych na Twitterze, używali rasistowskich określeń w stosunku do osób czarnoskórych, tzw. *n-word*. Interwencje wysyłały boty, czyli programy sterujące kontami twitterowymi. Nie stała za tym żadna sztuczna inteligencja — wystarczyło, że tweet zawierał obraźliwe słowo.

Treść interwencji była zawsze taka sama i odnosiła się do autora tweeta:

> *Hey man, just remember that there are real people who are hurt when you harass them with that kind of language*

W bardzo luźnym tłumaczeniu brzmiałoby to mniej więcej tak:

> *Witaj, waszmości, bacz jakowoż, azaliż po stronie drugiej jest człek prawdziwy, który twym knajackim językiem może urażon zostać*

Wiadomość była identyczna, ale konta, z których ją wysyłano, mocno różniły się między sobą. O to właśnie chodziło w eksperymencie — o sprawdzenie, jakie cechy osoby sankcjonującej sprawiają, że interwencja jest skuteczna. Poszczególne konta miały sugerować, że ich właściciele należą do tej samej grupy — białych mężczyzn, lub wręcz przeciwnie — że należą do osób czarnoskórych. Dodatkowo

konta z obu tych grup różniły się liczbą osób, które je obserwują, tzw. followersów. Wielu followersów może oznaczać, że konto jest popularne i wpływowe. Tu również wyróżniono dwie grupy — mało i dużo.

Badanie skuteczności interwencji polegało na sprawdzeniu, czy osoba sankcjonowana ograniczyła używanie n-wordów po otrzymaniu interwencji. Okazało się, że konta białych mężczyzn z dużą liczbą obserwujących „robiły robotę" — powodowały zauważalną redukcję rasistowskich inwektyw. Aż chciałoby się krzyknąć „No shit, Sherlock!", ale to wcale nie było takie oczywiste. No dobra, mogliśmy śmiało stawiać dolary przeciwko orzechom, że ze wszystkich grup to właśnie ta będzie miała największe szanse na sukces. Ale równie dobrze mogło się okazać, że żadna grupa nie spowodowała istotnych zmian, prawda?

A jednak zadziałało. Okazało się, że takiego gagatka wcale nie trzeba wyciszać, banować ani kasować mu wiadomości. Nie trzeba go bić, wyzywać ani „cancelować". Można z nim po prostu pogadać i pokazać, że po drugiej stronie są prawdziwi ludzie. Ludzie, którym wyrządza krzywdę, używając takiego, a nie innego języka. Oczywiście, że czasem nic to nie da — „gadał dziad do obrazu…", ale czasem da i właśnie to zainspirowało nas do przeprowadzenia podobnego eksperymentu, ale znacznie bardziej rozwiniętego i na znacznie większą skalę.

Po pierwsze dysponowaliśmy już wtedy bardzo precyzyjnymi modelami detekcji cyberprzemocy — ataków personalnych, znieważania, ubliżania, szyderstw, gróźb i szantaży czy nękania na tle seksualnym. Nie było zatem sensu wracać do koncepcji wykrywania słów kluczy. Zdecydowaliśmy się iść szeroko, choć oczywiście ataki personalne i tym podobne stanowiły najliczniejszą grupę niepożądanych treści.

Po drugie szukaliśmy otwartej platformy zrzeszającej różne społeczności internetowe wokół nurtujących je tematów. Platformy, na której toczą się prawdziwe, głębokie dyskusje na argumenty. Twitter, ze swoim limitem 280 znaków i średnią długością tweeta 70 – 100 znaków, nie spełniał naszych oczekiwań. Zdecydowaliśmy się na Reddita, który co prawda w Polsce nie cieszy się szczególną popularnością, ale w Stanach Zjednoczonych już jak najbardziej. I ma wszystko, czego wtedy szukaliśmy. Poszczególne społeczności tworzą tam tzw. subreddity, czyli podfora tematyczne, na których dyskutuje się w postach i komentarzach pod nimi.

Po trzecie chcieliśmy sprawdzić, jak skuteczne będą różne warianty samych interwencji — odnoszące się między innymi do empatii, normy czy autorytetu. To z kolei narzucało na nas całą masę wyzwań i ograniczeń, którym musieliśmy sprostać. Uznaliśmy, że ludzie nie powinni wiedzieć, że rozmawiają z botem, bo to negatywnie wpłynęłoby na eksperyment. A zatem potrzebowaliśmy bota, który udaje, że nie jest botem. Co więcej, wiedzieliśmy, że nasz eksperyment potrwa co najmniej kilka miesięcy, więc nasz bot musiał być naprawdę dobry w udawaniu człowieka.

Zakasaliśmy rękawy i wzięliśmy się do roboty. Na początek kupiliśmy kilka różnych kont. Zależało nam na tym, żeby konta miały co najmniej kilka miesięcy regularnej aktywności. Nic specjalnego, chcieliśmy wyglądać wiarygodnie, jak przeciętny użytkownik Reddita. Na tyle, na ile taki w ogóle istnieje. Chwila, moment — jak to „kupiliście"? A jak Szanowny Czytelnik myśli — skąd się biorą te wszystkie fejkowe konta na farmach trolli?

No więc kupiliśmy, przeprowadziliśmy wstępne testy i do eksperymentu wybraliśmy jedno z tych kont — 43-letniego mężczyznę, fana jazzu, interesującego się życiem i zwyczajami ośmiornic, który w wolnych chwilach pomaga młodszym użytkownikom w ich pracach domowych. To tematy, które poruszał w kilku postach lub komentarzach dziennie, wyłącznie na subredditach, które nie brały udziału w eksperymencie. Te treści tworzyli ludzie, dodatkowo budując wiarygodność konta. Wszystko inne odbywało się w 100% automatycznie.

Gdyby nasz bot przez kilka miesięcy wysyłał ciągle tę samą wiadomość, a nawet kilka, kilkanaście czy kilkadziesiąt podobnych, to w mgnieniu oka zostałby zdekonspirowany. Musieliśmy być sprytniejsi. Zbudowaliśmy generator interwencji, który był w stanie tworzyć setki tysięcy unikalnych wiadomości dla zadanej strategii (normy, empatii i autorytetu), wplatając w niektóre z nich odniesienia do „interweniowanych" komentarzy. Zdarzało mu się nawet pisać nieskładnie i robić literówki, żeby generowane treści wyglądały naturalnie.

Bot w czasie rzeczywistym analizował wszystkie posty i komentarze z wybranych subredditów. Gdy wykrył atak personalny, próbę znieważania lub ubliżania innym użytkownikom czy którąkolwiek z pozostałych form cyberprzemocy, odczekiwał kilka minut i wysyłał interwencję w odpowiedzi na wykryty post lub komentarz.

Nie odpowiadał od razu, żeby się nie dekonspirować, a czas odpowiedzi był za każdym razem ustalany losowo.

Mogliśmy zaczynać nasz wielki eksperyment. Nie będę ściemniał — początki były dość trudne, bo przesadziliśmy z empatią. I to nie tak trochę. Bardzo. Nasz bot zachowywał się tak, jak gdyby został wygnany z krainy tęczy, cukierków i jednorożców za bycie przesadnie miłym i uprzejmym. Wykazywał mnóstwo zrozumienia, tłumaczył, że każdy może mieć gorszy dzień, a na koniec życzył wszystkim zdrowia i szczęścia. Zdarzało się, że ludzie mu odpisywali. Moja ulubiona wiadomość zwrotna z tamtych czasów brzmiała mniej więcej tak: „gościu, ty naprawdę powinieneś już odstawić to LSD…".

Kilka pierwszych tygodni ku naszemu zdumieniu przyniosło wzrost ogólnej liczby ataków. Okazało się, że sami je prowokowaliśmy — częściowo stylem interwencji, a częściowo zdarzającymi się sporadycznie pomyłkami. Interwencja na brak ataku czasem skutkowała atakiem na nas. Wzięliśmy się ostro za tuning i wyśrubowaliśmy precyzję naszych modeli do ponad 93%. Zmieniliśmy też styl interwencji — z radykalnie miłych i empatycznych na zwyczajnie miłe i empatyczne.

I to już „zrobiło robotę". Od tego momentu interwencje leciały same, a my tylko obserwowaliśmy z boku, czy nie ma żadnych pożarów i czy przypadkiem bot nie zrobił czegoś nadzwyczaj głupiego. Warto jeszcze na moment zatrzymać się przy temacie precyzji, bo analiza reakcji użytkowników otworzyła nam oczy i pozwoliła inaczej spojrzeć na popełniane przez model błędy.

Gdy myślimy o wyniku fałszywie pozytywnym w detekcji ataków personalnych, chodzi nam zazwyczaj o użycie słów kojarzonych z obrażaniem innych, np. wulgaryzmów, w jakimś nieobraźliwym kontekście, np. „życie to dziw**" czy „ale mu kur** wyszło". To mieliśmy już wtedy opanowane prawie że do perfekcji. Co zatem mogło pójść nie tak? Chodziło o cel ataku i szerszy kontekst, wykraczający poza pojedynczą wiadomość.

Niech Szanowny Czytelnik wyobrazi sobie post na Reddicie, w którym autor opisuje następujące zdarzenie. Uprzedzam — będzie dość makabrycznie. Matka morduje dwójkę swoich małych dzieci, a następnie sama odbiera sobie życie. Co wykryliśmy w komentarzach? Bezpośrednie ataki na matkę, np. „powinnaś się za

to smażyć w piekle, suko". Zwracanie się do niej w drugiej osobie nie pomagało, bo na podstawie tej jednej wiadomości nie sposób odróżnić ataku na nią od ataku na autora posta. A co robi nasz bot? Oczywiście odpisuje pod każdym takim komentarzem, że on to wszystko rozumie, że każdy może mieć gorszy dzień, ale nie powinniśmy tak się do siebie zwracać. Czy Szanowny Czytelnik rozumie już, do czego zmierzam?

Takie wpadki nie były częste, ale zdarzały się od czasu do czasu. Osoba LGBTQ+ dzieliła się swoimi osobistymi i bardzo przykrymi doświadczeniami z nietolerancyjnym wujkiem. Prawie cała społeczność wyraziła w komentarzach wsparcie dla tej osoby, a bigoteryjnemu członkowi rodziny nieźle się oberwało. A co robi nasz bot? Tak jest — wbrew zdrowemu rozsądkowi i całej społeczności staje w obronie wuja-homofoba, bo przecież nie wypada tak brzydko o nim mówić.

Uwzględnienie tego typu przypadków było sporym wyzwaniem, ale udało się — ataki na nas praktycznie zniknęły i mogliśmy kontynuować eksperyment. Pierwsza ważna obserwacja dotyczyła tego, że automatyczny obrońca nie ogląda się na innych. Po prostu robi swoje. To, co było błędem w przypadku wuja-homofoba, później wielokrotnie okazywało się zbawienne, kiedy bot stawał w obronie ofiar cybernękania. Zwłaszcza że cybernękanie bardzo często dotyczy ataków „wielu na jednego". Nam, ludziom, ze względu na presję grupy trudno jest reagować w takich sytuacjach. A bot? Bot na szczęście ma to w cyberdupie…

Działaliśmy na wielu różnych subredditach. Na jednych było lepiej, na innych gorzej. Zauważyliśmy, że najtrudniejsze są subreddity polityczne, a sama trudność niewiele miała wspólnego ze stylem interwencji czy precyzją modeli. Chodziło o mentalność plemienną. Stając w obronie republikanów, dostawaliśmy łatkę demokraty, a broniąc demokraty, z automatu stawaliśmy się zwolennikami partii republikańskiej. W Polsce mamy podobną sytuację na linii PiS – PO.

Całkiem nieźle radziliśmy sobie na ogólnych subredditach, o wszystkim i o niczym, takich jak TooAfraidToAsk, gdzie każdy może zadać dowolne pytanie, często krępujące lub wstydliwe, na które w komentarzach odpowiadają pozostali użytkownicy. Ku naszemu zdziwieniu najlepsze rezultaty osiągaliśmy na subredditach takich jak MensRights, zrzeszających konkretne, nierzadko radykalne społeczności. W przypadku MensRights chodziło o mężczyzn upominających się o swoje prawa,

dyskutujących o dyskryminacji, społecznych nierównościach i niesprawiedliwościach. A to wszystko podlane incelowskim sosem. W skrócie: o tym, że kobiety mają lepiej.

Reddit jest darmowy, a wymieniane przeze mnie subreddity łatwe do znalezienia. Szanowny Czytelnik, jeśli będzie miał taką ochotę, może samodzielnie ocenić, na jak trudnym gruncie przyszło nam działać. Nasz eksperyment trwał ponad sześć miesięcy. W tym czasie udało się nam ograniczyć liczbę ataków pomiędzy członkami tej społeczności o ponad 20%. Samą marchewką, bez kija. Bez żadnych kar. Wyłącznie odpisując na posty i komentarze.

Zdarzało się, że nasze interwencje motywowały autorów obraźliwych komentarzy do ich skasowania lub usunięcia obraźliwych fragmentów. Czasem ci autorzy odpisywali botowi, przyznając mu rację — że rzeczywiście przesadzili, że nie powinni tak pisać, że mieli gorszy dzień. Czasem nasza interwencja prowokowała innych do zabrania głosu w obronie atakowanych. Niemal codziennie byliśmy świadkami nowych, pozytywnych reakcji. Ich kulminacją było zaproszenie naszego bota do grona moderatorów MensRights. Oczywiście osoba składająca propozycję nie wiedziała, że rozmawia z botem.

Sprawdziliśmy jeszcze jedno — czy nasze interwencje wpływały na zachowanie użytkowników globalnie, poza społecznościami, na którym działaliśmy. Okazało się, że tak. Jeśli ktoś dostał interwencję, to statystycznie stosował aż o 20% mniej agresji werbalnej niż przedtem; i to w obrębie całej platformy, na subredditach, na których nigdy nie było naszego bota. Jeśli Szanowny Czytelnik chciałby poczytać coś więcej o tym eksperymencie, to nasze wyniki opublikowaliśmy w artykule *Artificial intelligence against hate: Intervention reducing verbal aggression in the social network environment*. Do wygooglowania — gorąco polecam!

JAK DZIAŁA CHATGPT?

Mamy już wszystko, czego potrzebujemy, żeby omówić jedną z najpotężniejszych architektur głębokich sieci neuronowych. Na początek standardowe ostrzeżenie — rozdział trudny, nie polecam czytania „do poduszki". Za moment bowiem porozmawiamy o największej chyba rewolucji w historii sztucznej inteligencji. O architekturze, która na zawsze odmieniła przetwarzanie języka naturalnego, a która coraz śmielej zerka również w kierunku przetwarzania dźwięku i obrazu. O architekturze, która umożliwiła firmie OpenAI stworzenie ChatGPT — chatbota, o którym mówił cały świat.

A cóż to za architektura? Skoro jest tak potężna, to z pewnością każdy już o niej słyszał, prawda? I tu Szanowny Czytelnik może się lekko zdziwić… Jej nazwa jest zakodowana w rozwinięciu skrótu GPT — Generative Pre-trained Transformers. Chodzi o transformery. Nie mylić z Optimusem Prime, Megatronem, Bumblebee ani z żadnym innym przedstawicielem tej fikcyjnej rasy pozaziemskich istot mechanicznych, o których powstała cała seria filmów — z całkiem niezłym pierwszym i drugim oraz znacznie gorszymi kolejnymi.

A skąd taka nazwa? Chyba nikt do końca nie wie. Oczywiście poza autorami przełomowej publikacji *Attention Is All You Need*, w której to w 2017 roku ową architekturę zaproponowano. Część osób uważa, że chodzi o rozwiązywanie problemu transdukcji sekwencji, czyli transformacji sekwencji wejściowej w wyjściową. Żartownisie zaś twierdzą, że to od tego, że transformery bezpowrotnie transformowały sposób, w jaki „robimy AI". Niewykluczone jest również nawiązanie do popularnej i rozpoznawalnej franczyzy firmy Hasbro. Kto wie, kto wie?

Zanim przejdziemy do transformerów, przypomnijmy sobie jeszcze dwie sprawy — kontekst i znaczenie. W rozdziale o rekurencyjnych sieciach neuronowych omówiliśmy modele *sequence-to-sequence*. To takie sprytne bestyjki, które zamiast pojedynczych wektorów potrafią przyjmować i zwracać dane sekwencyjne, np. kolejne słowa w tekście. A mówiąc precyzyjnie: sekwencje wektorów reprezentujących kolejne słowa w tekście.

Modele tłumaczenia maszynowego na wejściu przyjmują sekwencję słów w języku, z którego tłumaczymy, a na wyjściu zwracają sekwencję słów w języku, na który tłumaczymy. Tak naprawdę to jest właśnie wspomniany przed chwilą problem transdukcji sekwencji, ale opisany po ludzku, bez używania tych wszystkich brzydkich słów. I stąd właśnie bierzemy kontekst — modelowanie *sequence-to-sequence* pozwala nam uwolnić się z wcześniejszych ograniczeń, dotyczących wejścia i wyjścia sieci neuronowych.

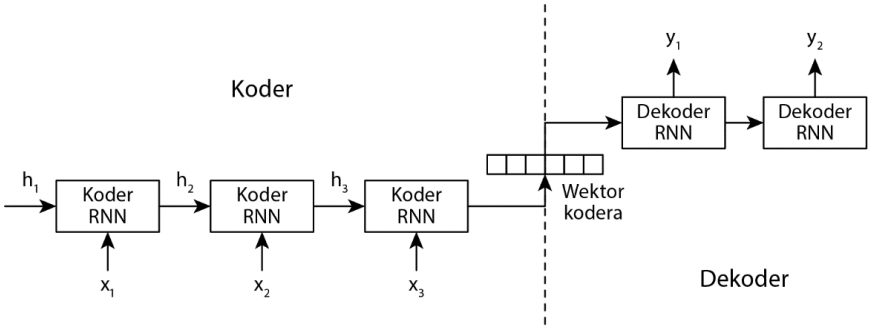
A skąd bierzemy znaczenie? Oczywiście z embeddingów, czyli z wektorowej reprezentacji słów, z zakodowaną bogatą informacją semantyczną i syntaktyczną, wydestylowaną ze statystycznej analizy sąsiadów tych słów w ogromnych korpusach tekstowych. Embeddingi można wytrenować samodzielnie, ale znacznie częściej korzysta się z owoców ciężkiej pracy innych. I dobrze, bo lepiej się dzielić, niż niepotrzebnie generować ślad węglowy na masową skalę.

Krótka dygresja o tym, jak działa to całe dzielenie się. Istnieje sobie organizacja non profit o nazwie Common Crawl, która od 2007 roku przeczesuje internet, gromadząc dane i udostępniając je innym. Jeśli wcześniej mówiliśmy o dużych liczbach, to tutaj wchodzimy na zupełnie nowy poziom. Common Crawl zbiera bambiliony pierdylionów — czy też petabajty, jak kto woli — danych. I co najważniejsze — udostępnia je za darmo.

Następnie taka przykładowa Meta (Facebook) bierze te dane i trenuje na nich embeddingi przy użyciu swojego algorytmu fastText. Zapewne nie robi tego z dobroci serca. Może korzysta z nich w swoich produktach? A może chce pochwalić się możliwościami swoich algorytmów? Nieważne. Ważne jest to, że w myśl zasady „karma wraca" udostępnia je wszystkie innym, również za darmo.

***

Punktem wyjścia dla naszych rozważań będzie architektura koder-dekoder dla modelu *sequence-to-sequence*, którą omawialiśmy w rozdziale poświęconym rekurencyjnym sieciom neuronowym. Oto jej schemat, tak dla przypomnienia:



Rysunek 27. Uproszczony schemat architektury koder-dekoder dla rekurencyjnych sieci neuronowych

Iksy (x_1, x_2, x_3) symbolizują kolejne elementy sekwencji wejściowej, igreki (y_1, y_2) — wyjściowej, a literą h (h_1, h_2, h_3) oznaczono stany ukryte. Tym, co umożliwia analizę kontekstu, jest oczywiście możliwość przetwarzania sekwencji. Zwróćmy jednak uwagę na fakt, że kontekst tak naprawdę przechowujemy i przekazujemy między koderem a dekoderem w postaci pojedynczego wektora, oznaczonego na schemacie jako wektor kodera.

Najpierw przyjrzyjmy się lewej stronie, czyli koderowi. W każdym kroku czasowym do sieci rekurencyjnej kodera wchodzi element sekwencji wejściowej i stan ukryty z poprzedniego kroku. Kontekst, reprezentowany przez wektor kodera, jest zatem stanem ukrytym sieci po wczytaniu ostatniego elementu sekwencji. Po prawej stronie, w dekoderze, kontekst jest przekazywany dalej w kolejnych krokach czasowych i używany do generowania kolejnych elementów sekwencji wyjściowej.

Podsumowując — wczytujemy sekwencję wejściową, budujemy kontekst i przekazujemy go dalej, żeby za jego pomocą wygenerować sekwencję wyjściową. Problem polega na tym, że nie wiemy, na którą część kontekstu powinniśmy zwracać uwagę w danym kroku czasowym — za każdym razem musimy patrzeć na całość. W przypadku krótkich sekwencji nie ma to większego znaczenia. Ale im dłuższa sekwencja, tym większy problem. Rozwiązaniem jest mechanizm atencji,

od angielskiego „attention", który pozwala modelowi skupić się na tym, co dla niego w danym momencie najważniejsze.

Zacznijmy od pewnej nieoczywistej oczywistości. Każdy stan ukryty kodera jest najmocniej skorelowany z pewnym konkretnym słowem sekwencji wejściowej. Zazwyczaj drugi stan ukryty będzie najmocniej skorelowany z drugim słowem sekwencji, bo właśnie to słowo zostało wczytane w drugim kroku czasowym. Ta obserwacja determinuje pewne zmiany, wymagane przez mechanizm atencji. Po pierwsze musimy zacząć przekazywać do dekodera wszystkie stany ukryte, a nie tylko ostatni. Po drugie dekoder w każdym kroku czasowym musi wykonywać dodatkowy zestaw czynności celem ustalenia, która część kontekstu jest dla danego kroku najistotniejsza.

Mechanizm jest dość prosty. Od teraz w dowolnym kroku czasowym mamy dostęp do wszystkich poprzednich stanów ukrytych kodera. Wymnażamy każdy z nich przez wagę, ustalaną w procesie uczenia sieci, a następnie sumujemy je wszystkie do pojedynczego wektora, reprezentującego kontekst w danym kroku czasowym.

```
h1: [100,  60,  20,   0,   0] * 0,1 =  [ 10,   6,   2,   0,   0]
h2: [ 40,  80,  80,  20,   0] * 0,2 = +[  8,  16,  16,   4,   0]
h3: [  0,  30,  60, 100,  30] * 0,7 = +[  0,  21,  42,  70,  21]
                                       ------------------------
                                kontekst = [ 18,  43,  60,  74,  21]
```

W powyższym przykładzie przedstawiono mechanizm wyznaczania kontekstu dla czwartego kroku czasowego. Po lewej stronie widzimy stany ukryte dla poprzednich trzech kroków (h_1, h_2, h_3), pośrodku wagi służące do wymnożenia poszczególnych stanów ukrytych, a po prawej wektory będące wynikiem tego działania. Na dole znajduje się zsumowany wektor reprezentujący kontekst.

Wróćmy do tłumaczenia maszynowego. Jeśli chcemy wygenerować czwarte słowo tłumaczenia, czyli czwarty element sekwencji wyjściowej, to musimy wiedzieć, na które elementy sekwencji wejściowej powinniśmy zwracać szczególną uwagę, a które możemy zignorować. Za to właśnie odpowiada mechanizm atencji. Pozwala modelowi nauczyć się zależności pomiędzy dwoma językami, uwzględniając

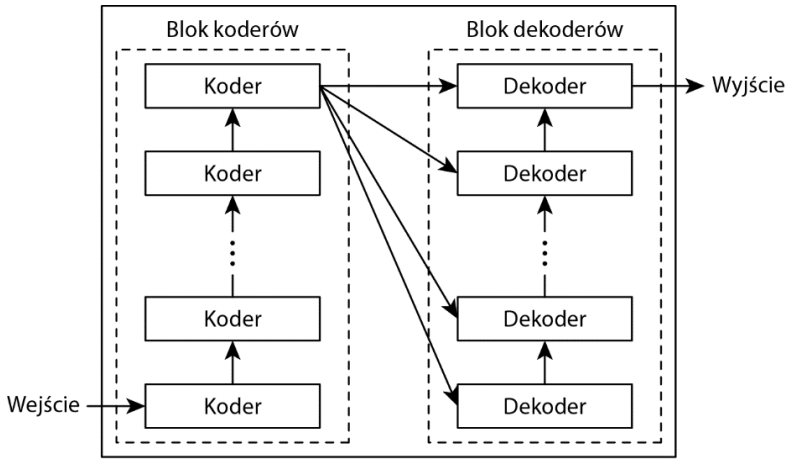
takie niuanse jak szyk zdania, odmiana czasownika czy konieczność zastępowania całych fraz pojedynczymi słowami.

Proste zaimki, przyimki, przysłówki, nazwy miesięcy czy dni tygodnia mogą mieć jedną i tę samą pozycję w obu sekwencjach. Ale wcale nie musi tak być, zwłaszcza w przypadku bardziej skomplikowanych fraz. Tłumacząc z angielskiego na polski, możemy zapisać proste zdanie „he got back" pojedynczym słowem — „wrócił".

W języku angielskim przymiotniki opisujące rzeczownik stawia się przed nim, np. „blue dress" („blue" — „niebieska", „dress" — „sukienka"). W języku hiszpańskim zazwyczaj jest na odwrót, np. „vestido azul" („azul" — „niebieska", „vestido" — „sukienka"). Model tłumaczenia maszynowego z atencją bardzo dobrze nauczy się tych zależności i będzie zwracał uwagę na całą frazę.

***

Podział na koder i dekoder okazał się na tyle skuteczny, że zdecydowano się go zastosować również w transformerach. Ale skoro jest tak dobry, to dlaczego mielibyśmy ograniczać się wyłącznie do jednego kodera i jednego dekodera? Zaproponowano więc architekturę składającą się z wielu koderów i dekoderów, zgrupowanych w dwa bloki o ściśle ustalonej strukturze i kierunku przepływu informacji. Ogólny schemat transformerów prezentuje się w taki oto sposób:



Rysunek 28. Ogólny uproszczony schemat architektury Transformer

Najpierw mamy blok koderów. Sekwencja wejściowa wchodzi do pierwszego kodera z bloku, a następnie jest przetwarzana i przekazywana po kolei, aż do ostatniego, przy czym wyjście z poprzedniego kodera stanowi wejście dla kolejnego. W bloku dekoderów mamy identyczny układ i kierunek przepływu informacji, z tą różnicą, że ostatni koder zasila nie tylko pierwszy, ale wszystkie dekodery naraz. W ten sposób każdy dekoder w bloku ma dostęp do tego samego kontekstu. Na sam koniec ostatni z nich zwraca sekwencję wyjściową.

Taka ogólna, pełna architektura posłużyłaby nam wtedy, gdybyśmy chcieli wytrenować model tłumaczenia maszynowego. Istnieją jednak zastosowania, które nie potrzebują obu bloków naraz. Generatywne modele językowe, takie jak GPT-4 zasilający ChatGPT, korzystają wyłącznie z bloku dekoderów. Z kolei maskowane modele językowe, takie jak BERT, wyłącznie z koderów.

A ponieważ o tłumaczeniach maszynowych mówiliśmy już całkiem sporo, to proponuję, żeby działanie transformerów omówić od razu na przykładzie modeli generatywnych z rodziny GPT. Szanowny Czytelnik nic nie straci, bo modele maskowane dostaną swój własny rozdział i w ten sposób dowiemy się, jak działają oba bloki, koderów i dekoderów, dla swoich flagowych zastosowań.

Na początek kilka słów wyjaśnienia, żebyśmy wszyscy rozumieli, o czym rozmawiamy. ChatGPT to chatbot, opracowany przez firmę OpenAI, który szturmem wdarł się do mediów i pod strzechy. Odpowiada na pytania, opowiada żarty, rozwiązuje zagadki logiczne i zadania matematyczne, potrafi napisać rozprawkę na zadany temat, a także opowiadanie, wiersz czy nawet działający fragment programu komputerowego.

ChatGPT generuje odpowiedzi, korzystając z tzw. dużych modeli językowych, od angielskiego „large language models", z bardzo popularnym skrótem LLM. Gdy piszę ten rozdział, ChatGPT korzysta z modelu GPT-3.5 w wersji darmowej oraz z GPT-4 w wersji płatnej. A ponieważ modele te służą do generowania treści, często nazywa się je również generatywnymi modelami językowymi. Mają to nawet w pierwszym członie swojej pełnej nazwy — Generative Pre-trained Transformers.

Generowanie sensownych odpowiedzi na dowolne pytania wydaje się niesamowicie trudnym i złożonym problemem. I tak rzeczywiście jest. Tym bardziej zaskakujący może być fakt, że generatywne modele językowe są trenowane do jednego, bardzo

prostego zadania — do przewidywania kolejnego elementu dla zadanej sekwencji wejściowej. Tak jest — ni mniej, ni więcej; model GPT na podstawie zadanego fragmentu tekstu ma odgadnąć, jakie słowo będzie następne.

Jak można się domyślać, trenowanie takiego modelu odbywa się na nieoznaczonych danych. Im więcej, tym lepiej. Chciałem nawet zażartować, że ściągamy w tym celu cały internet, ale to nie jest do końca żart. Potrzebujemy bambilionów pierdylionów tekstów — książek, artykułów, postów, komentarzy, wiadomości — bo z nich będziemy generować przykłady uczące do trenowania modelu.

Zerknijmy raz jeszcze na nasz ulubiony cytat z Einsteina:

Zdrowy rozsądek to zbiór uprzedzeń nabytych do osiemnastego roku życia.

Dla jednego takiego zdania możemy wygenerować całkiem sporo solidnych przykładów uczących:

1: [*Zdrowy*] → [*rozsądek*]

2: [*Zdrowy*] [*rozsądek*] → [to]

3: [*Zdrowy*] [*rozsądek*] [to] → [*zbiór*]

4: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] → [*uprzedzeń*]

5: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] [*uprzedzeń*] → [*nabytych*]

6: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] [*uprzedzeń*] [*nabytych*] → [*do*]

7: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] [*uprzedzeń*] [*nabytych*] [*do*] → [*osiemnastego*]

8: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] [*uprzedzeń*] [*nabytych*] [*do*] [*osiemnastego*] → [*roku*]

9: [*Zdrowy*] [*rozsądek*] [to] [*zbiór*] [*uprzedzeń*] [*nabytych*] [*do*] [*osiemnastego*] [*roku*] → [*życia*]

Na lewo od strzałki mamy sekwencję słów, którą zadajemy na wejście sieci, a po prawej stronie — jej wyjście, czyli pojedyncze słowo, które nasz model ma przewidywać. Zdaję sobie sprawę, że tych kilka przypadków nie wygląda zbyt spektakularnie, ale nie zapominajmy o skali. Mówimy o absurdalnie dużej ilości danych. Model GPT-4 był trenowany na około 10 bilionach słów (1 bilion to 10^{12}). Jeśli

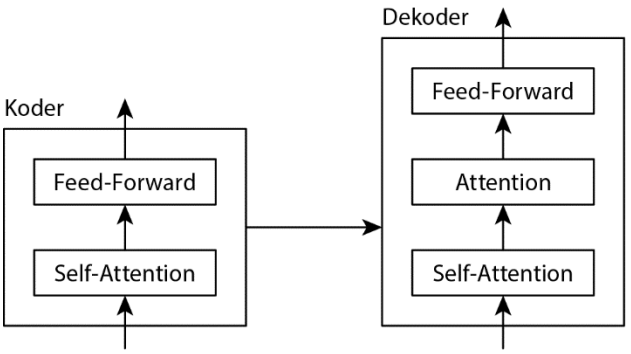
przyjmiemy, że gruba, 400-stronicowa powieść zawiera 100 tysięcy słów, to GPT-4 „przeczytał" 10 milionów takich książek.

Sam proces uczenia wygląda tak, że naszemu modelowi pokazujemy pierwszą lepszą sekwencję wejściową, np. „Zdrowy rozsądek to zbiór", i każemy mu odgadnąć kolejne słowo. A że model nigdy wcześniej takiej sekwencji nie widział, to nie będzie w stanie poprawnie wykonać tak postawionego zadania. Dostaniemy jakiś losowy wynik, np. „pomidorów".

Tłumaczymy modelowi, że wcale nie „pomidorów", tylko „uprzedzeń", a ten posłusznie, choć nie kryjąc rozczarowania, wyznacza wartość błędu i aktualizuje wagi. Zwiększa tym samym prawdopodobieństwo, że następnym razem dla zadanej sekwencji odpowie poprawnie. I tę czynność powtarzamy wielokrotnie, co doskonale podsumowuje łacińska sentencja „ad mortem defecatam".

***

Poszczególne kodery w bloku koderów nie różnią się między sobą. Identyczne są również dekodery w bloku dekoderów. Zaczniemy od prostego schematu, przedstawiającego oba te elementy. Niech Szanowny Czytelnik nie przejmuje się nowymi nazwami. Za moment wszystko sobie wyjaśnimy.



Rysunek 29. Uproszczony schemat budowy koderów i dekoderów w ramach architektury Transformer

Warstwę *Attention* już poniekąd znamy. To mechanizm atencji, który omówiliśmy sobie wcześniej. Jeśli trenujemy model tłumaczenia maszynowego, to ta warstwa pozwala nam określać, na które elementy sekwencji wejściowej powinniśmy zwracać uwagę, chcąc wygenerować odpowiadające im elementy sekwencji wyjściowej. Warstwa *Attention* znajduje się wyłącznie w dekoderach, bo tylko tam analizujemy kontekst sekwencji wejściowej z koderów.

Warstwa *Feed-Forward* też wygląda znajomo. I słusznie, bo składa się ona z wielu sieci neuronowych typu *feed-forward*. To tutaj realizowane jest docelowe zadanie całego modelu. W naszym przypadku chodzi o przewidywanie kolejnego słowa dla zadanej sekwencji wejściowej.

Warstwa *Self-Attention* to coś nowego i na niej skupimy się w pierwszej kolejności. *Self-Attention*, czy też samoatencja, działa bardzo podobnie do mechanizmu atencji. Nadal chodzi o ustalenie, na które słowa z sekwencji wejściowej należy zwrócić szczególną uwagę, chcąc jak najlepiej zrozumieć aktualnie przetwarzane słowo. Różnica jest taka, że atencja działa na styku kodera i dekodera, a samoatencja tylko w obrębie jednego z nich.

Najłatwiej będzie wyjaśnić to na przykładzie tłumaczenia maszynowego. Załóżmy, że chcemy przetłumaczyć następujące zdanie z języka polskiego na angielski:

```
„Adam ma psa, ale nie jest on szczególnie bystry."
```

I dla ustalenia uwagi od razu wersja angielska, jako efekt działania tłumaczenia:

```
„Adam has a dog, but he is not particularly smart."
```

Mechanizm atencji „patrzy" na zdanie po polsku z perspektywy dekodera, gdzie właśnie trwa generowanie sekwencji wyjściowej w języku angielskim. Jego zadaniem jest ustalenie, na które elementy sekwencji tekstu tłumaczonego należy zwrócić uwagę, żeby poprawnie wygenerować zadany element tłumaczenia. Chcąc wygenerować elementy „he" („on"), „is" („jest") oraz „not" („nie"), model powinien zwrócić uwagę na szyk przestawny w języku polskim, żeby błędnie nie przetłumaczyć fragmentu „nie jest on" jako „not is he".

Mechanizm samoatencji działa również w koderach i z tej perspektywy „patrzy" wyłącznie na zdanie po polsku, a w dekoderach — wyłącznie po angielsku. Jego

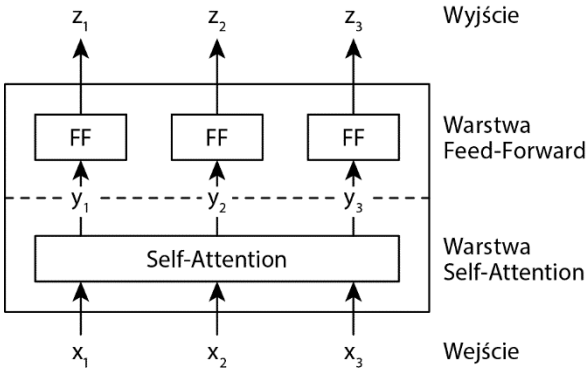
celem jest odkrycie tego, co w tych zdaniach jest istotne dla ich poprawnego zrozumienia. Do czego odnosi się zaimek osobowy „on"? Kto nie jest bystry? Pies? Czy może Adam? Warstwa *Self-Attention* pomaga nam to ustalić. Niczym lampa typu spotlight — w pierwszej kolejności oświetla psa, a nieco słabiej Adama.

My, barbarzyńcy od przetwarzania języka naturalnego, wszystkie tego typu związki między wyrażeniami w tekście wrzucamy do jednego worka i nazywamy koreferencją. Wtedy lingwiści stukają się po głowie, grożą nam palcem i mówią, że to anafory, w myśl zasady, że im trudniejsze i mniej zrozumiałe słowo, tym lepiej.

Mechanizm samoatencji, niczym wielogłowa hydra, bacznie obserwuje wiele miejsc naraz. Gdy jedna głowa patrzy na psa i Adama, próbując ustalić, kim jest „on", druga w tym samym czasie zerka na słowo „bystry", w kontekście czasownika „być" i użytej negacji. Nie omówimy go jednak w szczegółach, bo przez kilka kolejnych stron nie wygrzebalibyśmy się z wektorów i macierzy. Wystarczy zapamiętać, że pozwala uchwycić relacje i zależności wewnątrz sekwencji wejściowej, czy to w koderze, czy w dekoderze. Wskazuje, co dla zrozumienia poszczególnych elementów sekwencji jest istotne i jak bardzo, poprzez przypisanie wag określających tę istotność.

***

Kodery i dekodery, pomijając warstwę *Attention*, działają bardzo podobnie. Kolejny schemat pozwoli nam zajrzeć do wnętrza kodera. Pamiętajmy tylko, że to, co tu omawiamy, w równym stopniu dotyczy też dekoderów.



Rysunek 30. Uproszczony schemat działania kodera w ramach architektury Transformer

Sekwencja wejściowa (x_1, x_2, x_3) wchodzi do warstwy *Self-Attention*. Napisałem to tak, jak gdyby nigdy nic, ale to jest petarda. Cała sekwencja wchodzi naraz, bez żadnych kroków czasowych, a każdy jej element podąża swoją własną ścieżką. Mechanizm samoatencji wymaga co prawda, aby te ścieżki były od siebie zależne, krzyżowały się i przeplatały, żeby te zależności dało się odkryć i zważyć. Jednak zaraz po opuszczeniu warstwy ścieżki ponownie rozdzielają się do osobnych stanów pośrednich (y_1, y_2, y_3).

W warstwie *Feed-Forward* mamy tyle niezależnych sieci, ile jest elementów sekwencji. Każda z nich oznaczona jako FF to sieć typu *feed-forward*. Nie różnią się one w zasadzie niczym od sieci, które omawialiśmy na początku tej książki. I tu już nic się nie miesza. Każda ścieżka wchodzi do swojego własnego perceptronu wielowarstwowego, gdzie realizowane jest docelowe zadanie modelu. W modelach generatywnych jest to przewidywanie kolejnego elementu sekwencji.

Na wyjściu z warstwy *Feed-Forward* dostajemy wektory o rozmiarze użytego słownika (z_1, z_2, z_3). Ich wartości możemy interpretować jako prawdopodobieństwa, z jakimi odpowiadające im słowa stanowią kolejny element zadanej sekwencji wejściowej. I z tym już możemy robić, co nam się żywnie podoba.

Możemy wybrać słowo z najwyższą wartością i dowiedzieć się, że zdrowy rozsądek to zbiór uprzedzeń, a nie pomidorów. Możemy, jak w przypadku komunikatorów lub klientów pocztowych, wyświetlać kilka słów z najwyższymi wartościami i pozwolić użytkownikowi wybrać któreś z nich. Możemy też pozwolić mu wpisać swoje, co później skrzętnie wykorzystamy przy dotrenowywaniu modelu.

I to w zasadzie wyczerpuje temat transformerów. Przynajmniej na poziomie ogólności, który chciałbym utrzymać w tej książce. Szanowny Czytelnik doskonale zdaje sobie sprawę, że to zaledwie wierzchołek góry lodowej. Ale taki, który pozwala nam zrozumieć, jak to wszystko mniej więcej działa. A to z kolei wystarczy, żeby pomówić nieco szerzej o konsekwencjach, szansach i zagrożeniach związanych ze sztuczną inteligencją i jej rozwojem.

Na koniec wróćmy jeszcze do tej petardy, która co prawda wybuchła, ale zdecydowanie za cicho. Transformery pozwalają nam wczytywać całe sekwencje naraz.

Bez żadnych kroków czasowych, które stanowiły przykrą uciążliwość rekurencyjnych sieci neuronowych. Zwracam na to uwagę, bo właśnie zahaczamy o tematy mające niemały wpływ na światową ekonomię i gospodarkę.

Jeśli Szanowny Czytelnik zastanawiał się kiedyś, dlaczego tak często w rozmowach o sztucznej inteligencji ni stąd, ni zowąd wypływa temat procesorów graficznych, to między innymi dlatego. Te wszystkie ścieżki, którymi przez kodery i dekodery wędrują nasze sekwencje, z warstwą *Self-Attention* włącznie, mogą być realizowane równolegle. Nie musimy czekać z przetwarzaniem drugiego elementu sekwencji, aż skończy się przetwarzać pierwszy. Można to robić jednocześnie.

A do tego — jak się okazało — idealnie wręcz nadają się procesory graficzne. Procesory, które na masową skalę produkuje dwóch liczących się na rynku graczy. I to przy stale rosnącym popycie. To wszystko rodzi niebagatelne konsekwencje, które zasługują na osobny rozdział i ja z przyjemnością go dla Szanownego Czytelnika napiszę.

O TYM, JAK SZTUCZNA INTELIGENCJA ZEPSUŁA MI SZACHY

Szachy, królewska gra, to doskonały temat na interludium, bo „ma to wszystko" — jest tu pasja, sport, historia, popkultura i sztuczna inteligencja. A dzięki miniserialowi Netfliksa *Gambit królowej* z 2020 roku jest również rosnąca popularność, która stopniowo gasła od lat 70. i 80. ubiegłego wieku, kiedy to szachy cieszyły się największym zainteresowaniem w historii.

Moja przygoda z szachami zaczęła się w dzieciństwie, ale nie jakoś superwcześnie. Na pewno chodziłem już wtedy do podstawówki. To dziadek kupił mi pierwszą szachownicę i pierwszą książkę o szachach, ale co ważniejsze — zawsze miał dla mnie czas. Rozegraliśmy razem setki partii, a dzień, w którym po raz pierwszy udało mi się wygrać, pamiętam do dziś. Dziadek nigdy nie dawał mi forów. To było zasłużone zwycięstwo, które nauczyło mnie, że „trening czyni mistrza", a ciężka praca przynosi w końcu oczekiwane rezultaty. Myślę, że to jedno z tych głębokich doświadczeń, które kształtują charakter młodego człowieka. Bardzo Ci za to dziękuję, dziadku!

Mój tata, dla odmiany, nie był za bardzo szachowy. Od taty dostałem anegdotę. Dawno, dawno temu do szacha, króla Persji, przybył kupiec. Pokazał znudzonemu władcy nową grę — szachy — która tak mu się spodobała, że postanowił obsypać kupca złotem. Ale kupiec okazał się nie w ciemię bity i poprosił o inną nagrodę. Niech król weźmie szachownicę i na jej pierwszym polu położy jedno ziarno pszenicy, na drugim dwa ziarna, a na kolejnych — cztery, osiem, szesnaście,

trzydzieści dwa i tak dalej. Władca nie mógł uwierzyć w swoje szczęście — pszenica zamiast złota. Ucieszył się, że zaoszczędzi mnóstwo pieniędzy, bo trafił na kupca-idiotę. Ochoczo przystał na jego propozycję, ale gdy przyszło do wypłacenia nagrody, okazało się, że tyle pszenicy nie ma w całym królestwie, a być może i na całym świecie.

Policzmy. Na pierwszym polu król miał położyć 2^0 ziaren, a na kolejnych — 2^1, 2^2, 2^3 i tak dalej. Szachownica ma 64 pola, więc na ostatnim polu powinno znaleźć się 2^{63} ziaren, czyli tyle:

```
9 223 372 036 854 775 808
```

Na gram pszenicy przypada około 25 ziaren, a jedna tona to milion gramów. Kalkulator w dłoń i wychodzi na to, że na ostatnim polu szachownicy powinno znaleźć się mniej więcej tyle ton pszenicy:

```
368 834 881 474
```

No i teraz czy to dużo, czy mało? Zbiory pszenicy w Polsce w 2022 roku GUS oszacował na (rekordowym) poziomie 13,5 miliona ton. Tak że ten no, dużo w pytę.

Z tatą wiele nie pograłem, ale dzięki szachowej anegdocie dowiedziałem się dwóch niezwykle przydatnych w życiu rzeczy. Po pierwsze, że 2^x rośnie bardzo szybko. A po drugie, żeby zanim na cokolwiek się zgodzę, najpierw wszystko dokładnie policzyć. Jak Szanowny Czytelnik może się domyślać, to ostatnie zapewnia mi nieustającą przychylność, a nawet sympatię doradców kredytowych i ubezpieczeniowych.

Szachom zawdzięczam jeszcze jedną przydatną umiejętność. Tym razem bez cienia sarkazmu. Mniej więcej w tym samym czasie co grać, zacząłem uczyć się programowania. Szachy należały dla mnie do świata rzeczywistego, doświadczanego za pomocą zmysłów. Miałem planszę, piony, figury. Mogłem je zobaczyć, dotknąć, przesunąć. Ruch figur odbywał się w przestrzeni. A jednak wszystko, co działo się podczas partii, można było zapisać w tzw. szachowej notacji algebraicznej. Jednoznacznej i „zrozumiałej" dla maszyny.

Byłem dzieciakiem, a fakt, że doświadczany zmysłami świat rzeczywisty, przynajmniej w jakimś skończonym zakresie, można zapisać w sposób formalny i jednoznaczny, był dla mnie nie lada odkryciem. Mogłem to wykorzystać w tym drugim

świecie — w ścisłym świecie programowania — modelując zasady gry i tworząc swój pierwszy program szachowy. Oczywiście do grania z innym człowiekiem, a nie z komputerem, ale i tak pamiętam, że był to dla mnie niemały powód do dumy. Wydaje mi się, że podobne olśnienie mogą przeżywać osoby, które dowiadują się, że muzykę — dźwięki, rytm, melodię czy dynamikę — też można zapisać w podobny sposób — za pomocą notacji muzycznej, zwanej również zapisem nutowym.

Później odkryłem, że tym właśnie zajmuje się fizyka — próbą opisu rzeczywistości w sposób formalny i jednoznaczny. Zakochałem się i mimo że od wielu lat zajmuję się sztuczną inteligencją, to właśnie fizyka jest moją pierwszą wielką miłością, która nigdy nie wyblakła i nigdy nie przeminęła. Za każdym razem, gdy podkreślam, że sieć neuronowa nie rozumie języka naturalnego, obrazu czy dźwięku, a zamiast tego potrzebuje wektorów i liczb, to tak naprawdę mówię o próbie opisu świata rzeczywistego w sposób zrozumiały dla maszyny. I właśnie ten specyficzny sposób myślenia, tę klapkę w mózgu, odblokowały u mnie szachy.

Wraz z odejściem dziadka odeszło również moje zainteresowanie szachami. Zagrałem raptem kilka partii w liceum, ale nie z pasji, a dla zabicia czasu. Na sporadycznych wagarach albo podczas „okienek" między lekcjami. I dopiero w dorosłym życiu, na pewno po trzydziestce, algorytm YouTube'a z jakiegoś powodu zaczął proponować mi analizy rozgrywek arcymistrzów szachowych, tych dawnych i obecnych. Nigdy nie oglądałem sportu w telewizji, ale w szachy wkręciłem się na maksa. Najpierw były analizy słynnych partii, ale potem — jak narkoman na głodzie — chciałem coraz więcej i zacząłem oglądać również transmisje z ważniejszych turniejów.

Dowiedziałem się o szachach kilku interesujących faktów, o których wcześniej nie miałem pojęcia. Przede wszystkim szachy to gra młodych. Kiedyś myślałem, że najważniejsze jest doświadczenie, co w oczywisty sposób promowałoby starszych graczy. Nic bardziej mylnego. W tym aspekcie szachy niewiele różnią się od innych dyscyplin sportowych.

W 2023 roku mistrzem świata w szachach klasycznych został 31-letni Ding Liren. Wcześniej przez niemal dekadę ten tytuł należał do Magnusa Carlsena, który zdobył go po raz pierwszy w 2013 roku w wieku… 22 lat. I to wcale nie jest jakiś ewenement.

Garri Kasparow, Michaił Tal, Anatolij Karpow, Władimir Kramnik, Emanuel Lasker — oni wszyscy sięgnęli po tytuł mistrza świata w wieku 22 – 25 lat. Okazuje się, że grając na najwyższym poziomie, sprawne ciało jest równie ważne jak sprawny umysł.

Wydało mi się to wtedy niezwykle ciekawe, bo kiedyś z fizyką było podobnie — również była domeną młodych. Z jakiegoś powodu utarło się, że autorytet w dziedzinie fizyki powinien być leciwy. A im starszy, tym lepszy i bardziej wiarygodny. Może dlatego, że większość z nas kojarzy Einsteina jako wesołego dziadka z wystawionym językiem ze słynnego zdjęcia wykonanego w 1951 roku? A może dlatego, że przeróżne programy popularnonaukowe, z nielicznymi wyjątkami, proponują nam same siwe głowy w roli autorytetów?

Otóż Einstein swoje dwie przełomowe prace — dotyczące szczególnej teorii względności i efektu fotoelektrycznego — opublikował w 1905 roku, mając zaledwie 26 lat. A swoje opus magnum, czyli ogólną teorię względności, przedstawił w pracy z 1915 roku, czyli w wieku 36 lat, po wielu latach nierównej walki z matematyką.

No dobra, ale Einstein to Einstein — był wyjątkowym geniuszem, prawda? W takim razie spójrzmy na pozostałych „ojców" mechaniki kwantowej i ich przełomowe prace. Niels Bohr wyjaśnił kwantowanie poziomów energetycznych w atomie wodoru w wieku 28 lat (1913). Werner Heisenberg opublikował mechanikę macierzową (równanie Heisenberga) w wieku 24 lat (1925). Erwin Schrödinger — żeby nikt nie posądził mnie o „cherry picking" — sformułował mechanikę falową (równanie Schrödingera) w jakże sędziwym wieku 39 lat (1926). Zaś Paul Dirac połączył mechanikę kwantową ze szczególną teorią względności w wieku 25 lat (1927).

Fizyka, a w szczególności fizyka kwantowa, z całą pewnością była dyscypliną młodych. I jestem przekonany, że także współcześnie nasze największe możliwości intelektualne przypadają gdzieś w okolicach trzydziestki, może ciut wcześniej. A jeśli tak, to co najmniej kilka lat wcześniej powinniśmy stać rozgrzani w blokach startowych, wyposażeni w całą niezbędną wiedzę i gotowi robić naukę.

A tymczasem programy nauczania wyglądają coraz gorzej. Jako ludzkość wiemy coraz więcej, a w czasie przeznaczonym na edukację uczymy coraz mniej. Zamiast

rewidować, kondensować i uczyć nowych zagadnień jak najwcześniej, rozwlekamy je w czasie i uczymy coraz później. Jeśli przesuwamy całki z liceum na studia, to chcąc nie chcąc na studiach zrobimy mniej niż wcześniej. Na doktoracie musimy nadrabiać materiał ze studiów magisterskich i nagle okazuje się, że nasz „prime time" spędziliśmy na nieefektywnej edukacji zamiast na pracy naukowej.

Ale dość już tej dygresji. Wracamy do szachów, bo z mojej dorosłej fascynacji dowiedziałem się jeszcze jednej superciekawej rzeczy. W przeciwieństwie do innych sportów w szachach istnieje coś takiego jak doping elektroniczny, zwany również dopingiem cyfrowym. Chodzi o niedozwolone korzystanie z programów komputerowych podczas turniejów, np. w toalecie na telefonie. Okazuje się bowiem, że żaden śmiertelnik, żaden arcymistrz czy nawet mistrz świata już nigdy, przenigdy nie wygra z algorytmem szachowym. Sztuczną inteligencję może pokonać tylko inna sztuczna inteligencja. I to o tym porozmawiamy sobie nieco szerzej właśnie teraz.

***

Zanim wytoczymy nasze najcięższe działa, dobrą praktyką jest sprawdzić, czy przypadkiem danego problemu nie da się rozwiązać prościej, na przykład metodą siłową, czyli *brute force*, jak mawiają Anglosasi. W szachach nie ma przecież żadnej ukrytej informacji — wszystko jest na planszy. Wystarczy wypisać wszystkie możliwe kombinacje i *voilà* — wygrywamy, wybierając tylko te, które niechybnie prowadzą nas ku zwycięstwu.

To ile jest tych kombinacji? Na ile różnych sposobów można rozegrać partię szachów? Z pomocą przychodzi nam Claude Shannon, ojciec teorii informacji, który oszacował tę liczbę na 10^{120} (słownie: dziesięć do potęgi sto dwudziestej). Warto dodać, że jest to szacowanie konserwatywne, z niedomiarem, co oznacza, że zawsze w myślach powinniśmy sobie przed tą liczbą dodać „co najmniej".

Czy to dużo? Powiedzmy, że zalecam ostrożność, żeby nie skończyć jak król Persji, zmuszony do wypłaty 2^{63} ziaren pszenicy na rzecz rezolutnego kupca. Liczba Shannona — bo oczywistym jest, że tak piękna liczba dostała swoją nazwę — jest znacznie większa niż liczba atomów w całym obserwowalnym wszechświecie, którą szacuje się na 10^{80}.

Gdybyśmy dysponowali superkomputerem, który przeliczałby 1 grę w ciągu 1 mikrosekundy, czyli nomen omen megaszybko, bo milion gier na sekundę, to nie doczekalibyśmy nawet 10^{24} gry. A dlaczego? A dlatego, że wszystko wskazuje na to, że za około 5 miliardów lat, czyli za jakieś 10^{23} mikrosekund, nasze ukochane Słońce zmieni się w czerwonego olbrzyma i pochłonie Ziemię.

No dobra, ale to są wszystkie kombinacje, a przecież doskonale wiemy, że nie wszystkie z nich mają sens z perspektywy dwóch starających się wygrać graczy. Może gdybyśmy skupili się wyłącznie na racjonalnych kombinacjach, to nie musielibyśmy aż tyle liczyć? Trudno odmówić temu logiki, ale może niech Szanowny Czytelnik nie biegnie jeszcze po kartkę i ołówek. Szacuje się, że liczba samych racjonalnych kombinacji to około 10^{48}. To rzeczywiście dużo mniej niż 10^{120}, ale nadal dużo więcej niż 10^{24}, co oznacza, że Ziemia i tak tego nie doczeka.

Brute force odpada, a zatem trzeba poszukać innych rozwiązań. Choć historia maszyn do gry w szachy jest dużo starsza niż historia komputerów, my skupimy się na ostatnich 28 latach i zaledwie kilku kluczowych wydarzeniach, które przeprowadzą nas od fazy „hej, arcymistrzu, zagrajmy jak równy z równym" do fazy „marna istoto białkowa, nie masz najmniejszych szans w starciu z moją krzemową potęgą".

Jest 10 lutego 1996 roku. Deep Blue, system komputerowy stworzony przez firmę IBM, właśnie wygrał pierwszą w historii partię szachów z ówczesnym mistrzem świata — Garrim Kasparowem. Powiedzmy sobie szczerze, IBM miało nie lada fart, że akurat pierwsza rozgrywka okazała się dla nich zwycięska. Cały mecz składał się bowiem z 6 partii, z czego Deep Blue wygrał jedną, przegrał trzy, a dwie zremisował, przez co cały pojedynek zakończył się wynikiem 4:2 dla Kasparowa. Niemniej po szczęśliwej dla IBM-u partii otwarcia w świat poszedł przekaz, że oto maszyna pokonała mistrza. Pierwsze „prawdziwe" zwycięstwo miało miejsce rok później — w maju 1997 roku, kiedy to mocno ulepszony Deep Blue wygrał mecz rewanżowy z Kasparowem z wynikiem 3,5:2,5.

Wcześniej napisałem, że metoda siłowa odpada, ale Deep Blue właśnie to robił — realizował algorytm typu *brute force*. Tyle że nieporównywalnie mniej złożony niż ten „absolutny", rozważany przez nas wcześniej. Deep Blue tworzył drzewo wszystkich możliwych posunięć i analizował je, próbując znaleźć to optymalne.

Oczywiście robił to tylko do pewnej ustalonej głębokości drzewa, stosując przeróżne metody optymalizacyjne. Zarówno takie, których zadaniem było jak najszybsze odrzucenie ewidentnie złych ścieżek, a pogłębianie tylko tych perspektywicznych, jak i takie, które korzystały z tzw. bazy końcówek, czyli z gotowych rozwiązań wszystkich możliwych partii, wyznaczonych dla niewielkiej, określonej liczby figur wciąż pozostających na planszy.

Czyli dokładnie to, co proponowaliśmy wcześniej jako „absolutny" *brute force*, ale — ze względu na Ziemię pochłanianą przez Słońce — ograniczony wyłącznie do końcowych faz rozgrywki. Jeśli partia docierała do znanego z bazy ustawienia, to algorytm przełączał się z drzewa na bazę i dalej prowadził już komputerowego gracza „za rączkę" aż do zwycięstwa.

Deep Blue był przede wszystkim bardzo potężnym — jak na ówczesne czasy — komputerem, który dzięki dużej mocy obliczeniowej mógł wystarczająco szybko realizować algorytm napisany przez programistów. Optymalne wartości poszczególnych parametrów zostały wyliczone przez system na podstawie analizy tysięcy partii szachowych, ale sam system wykonywał krok po kroku to, co nakazał mu człowiek. Czy Deep Blue nazwalibyśmy sztuczną inteligencją? No cóż, to zależy wyłącznie od przyjętej definicji, ale sam Kasparow stwierdził, że dostrzega w jego sposobie gry przejawy wyższej inteligencji.

Zabawny i pouczający jest fakt, że duży wpływ na to stwierdzenie miał charakterystyczny i zaskakujący ruch 44. z pierwszej zwycięskiej partii. Dlaczego miałoby to być zabawne i pouczające? Ano dlatego, że po 15 latach jeden z architektów Deep Blue wyznał, że owo strategiczne posunięcie było wynikiem błędu systemu. Kończył się czas, a algorytm, nie mogąc znaleźć optymalnego ruchu, zagrał losowo. A zatem najwyższym przejawem inteligencji było posunięcie całkowicie przypadkowe. Nie wiem jak Szanownego Czytelnika, ale mnie bawi to niezmiernie…

***

Stockfish to nazwa, którą zna praktycznie każdy szachista. To najpopularniejszy obecnie silnik szachowy, rozwijany jako wolne oprogramowanie od 2008 roku i dostępny za darmo na wielu platformach. W 2023 roku wypuszczono wersję nr 16,

ale zanim powiemy sobie, jak działa, spróbujmy oszacować, jak jest mocny w porównaniu z ludzkim graczem.

Otóż istnieje coś takiego jak ranking szachowy, pozwalający obliczać relatywną siłę gry szachistów w punktach Elo. To od nazwiska Arpada Elo, a nie od klasycznego polskiego powitania „Elo, mordo!". Bez wnikania w szczegóły, całkowity nowicjusz ma 750 – 1000 Elo, przeciętny gracz z klubu szachowego 1400 – 1600 Elo, mistrz 2200 – 2400, arcymistrz 2400 – 2500 Elo, a „największe Elo wśród śmiertelników" ma Magnus Carlsen, były mistrz świata — aż 2830 Elo. No to mamy już jako takie wyczucie. W styczniu 2024 roku Stockfish 16 uzyskał ranking 3631 Elo. Dziękuję, dobranoc!

Oznacza to mniej więcej tyle, że tak jak przeciętny gracz z klubu szachowego nie wygra z Magnusem Carlsenem, tak Magnus Carlsen nie wygra ze Stockfishem. Dzieli ich przepaść.

Stockfish, podobnie jak Deep Blue, przeszukuje drzewo możliwych posunięć w poszukiwaniu tych najkorzystniejszych. Ale robi to bez porównania lepiej niż komputer szachowy IBM-u. Silnik składa się z trzech komponentów. Pierwszym z nich jest reprezentacja szachownicy, która w bardzo wydajny sposób koduje układy figur na planszy, żeby móc je łatwo przechowywać i przeszukiwać.

Drugim jest ewaluacja szachownicy, która na podstawie zakodowanych układów figur ocenia szanse na wygraną. Co ciekawe, od wersji 12. ten komponent jest realizowany przez sztuczną sieć neuronową, choć wcześniej działał w oparciu o sprawdzanie szeregu określonych warunków, takich jak bezpieczeństwo króla, przewaga materialna, kontrola centralnych pól szachownicy, nieposiadanie izolowanych pionów czy posiadanie gońca na przekątnej szachownicy.

Trzecim i ostatnim komponentem jest właśnie heurystyczne przeszukiwanie drzewa, gdzie mądre słowo „heurystyka" można czytać jako „poszukiwanie korzystnych rozwiązań bez gwarancji znalezienia najlepszego". Zainteresowanego Czytelnika zostawiam z hasłem „algorytm alfa-beta".

A zatem żaden człowiek nie wygra już ze Stockfishem. Ani z żadnym innym, podobnym silnikiem szachowym. Trzymajmy się jednak tego Stockfisha, bo w mistrzostwach szachów komputerowych — a takowe istnieją, np. Top Chess Engine

Championship (TCEC) — to właśnie on z reguły zajmuje pierwsze miejsce. Nowe pytanie brzmi: czy istnieje cokolwiek, co byłoby w stanie skopać tyłek Stockfishowi?

I odpowiedź brzmi: tak. Hmm, to może by tak wytrenować ogromną sieć neuronową, używając do tego całej dostępnej wiedzy ludzkości, wszystkich rozegranych partii? A potem jeszcze wykorzystać partie rozegrane przez Stockfisha i inne silniki szachowe? Może taki gargantuiczny model, znający wszystkie sztuczki Stockfisha, dałby radę z nim wygrać?

Nic bardziej mylnego. Wszystkie te partie, rozegrane według podobnych schematów, byłyby dla modelu jedynie obciążeniem. Naukowcy z firmy DeepMind mieli na to inny pomysł. Swój algorytm, nazwany AlphaZero, wyposażyli jedynie w zasady gry, możliwe ruchy figur na szachownicy i cel — „danie mata", czyli zagrożenie królowi przeciwnika biciem, którego nie da się uniknąć. A gdy już to zrobili, powiedzieli „no, to teraz graj sam ze sobą, aż się nauczysz". No i się nauczył, oj, nauczył…

W grudniu 2017 roku AlphaZero zmierzył się ze Stockfishem w wersji 8., najnowszej na tamte czasy. W pierwszym meczu, na 100 partii, AlphaZero wygrał 25 razy, grając białymi, 3 razy, grając czarnymi i zremisował pozostałe 72 partie, nie przegrywając ani razu. Następnie rozegrano 12 setów po 100 partii, gdzie każdy set przedstawiał jeden z popularnych debiutów szachowych, czyli początkowych faz rozgrywki. W tym nie lada maratonie AlphaZero zwyciężył 290 razy, zremisował 886 i przegrał 24 razy. A żeby było ciekawiej, dokonał tego wszystkiego, grając wcześniej sam ze sobą przez zaledwie 9 godzin, gdzie poziom Stockfisha osiągnął już po 4 godzinach nauki.

I w zasadzie na tym mógłbym zakończyć, pozostawiając Szanownego Czytelnika z tezą, że samoucząca się sztuczna sieć neuronowa znacznie przewyższa człowieka i wszystko, co ów człowiek jest w stanie stworzyć. Że to, co ludzkie, jest tylko zbędnym balastem, który powinniśmy bez żalu porzucić w drodze ku koneksjonizmowi doskonałemu. Ale to nie byłaby prawda. Przynajmniej nie w szachach, których historia nie kończy się przecież na 2017 roku.

Na kanwie sukcesu AlphaZero, którego twórcy dalej nie rozwijali, rozpoczęto nowy opensource'owy projekt pod nazwą Leela Chess Zero, replikujący zasady działania algorytmu DeepMind. I rzeczywiście, w styczniu 2019 roku Leela była w stanie pokonać Stockfisha 8 w składającym się ze 100 partii meczu.

Ale w 2019 roku była już dostępna poprawiona wersja Stockfisha, która wygrała z Leelą w lutym tego samego roku w 14. edycji Top Chess Engine Championship (TCEC 14). Twórcy Leeli również nie dali za wygraną i rozpoczął się wyścig. TCEC 15 wygrała Leela, TCEC 16 Stockfish, TCEC 17 Leela… i w sumie tyle. Wszystkie kolejne mistrzostwa wygrywał już Stockfish. Aż do najnowszej edycji, TCEC 25, zakończonej w październiku 2023 roku.

Nie chciałbym być źle zrozumiany. To, że sieć neuronowa, grając sama ze sobą przez kilka godzin, była w stanie pokonać najlepszy ówczesny silnik szachowy, jest czymś absolutnie fenomenalnym. Być może ten sposób uczenia był w stanie dodać do rozgrywki jakiś czynnik, którego nie był w stanie dodać człowiek. Ale też chciałbym, żeby jasno wybrzmiało, że obecnie wygrywającym podejściem jest podejście hybrydowe, łączące sztuczne sieci neuronowe z heurystycznym przeszukiwaniem drzewa i algorytmami napisanymi w całości przez ludzi. Ani czysto symboliczne, ani czysto koneksjonistyczne. Hy-bry-do-we. Jeszcze do tego wrócimy, oj, wrócimy…

HELLO, MY NAME IS BERT

Gdyby GPT był najprzystojniejszym chłopakiem w klasie, grającym w szkolnej reprezentacji, którego każdy zna i lubi, to BERT byłby jego cichym i nieśmiałym, a przy tym zdolnym i bardzo pracowitym bratem. BERT należy do pretrenowanych modeli językowych, opartych na maskowanym modelowaniu języka, po angielsku „masked language models", w skrócie MLM.

Pretrenowane modele językowe to takie coś, co wsadzamy do innych modeli uczenia maszynowego, żeby „umiały język". Wyobraźmy sobie biochip z Cyberpunka, który wsadzamy sobie do głowy i cyk — gadamy, piszemy i czytamy w języku, którego wcześniej nie znaliśmy. Albo jak w filmie *Matrix* — Tank wgrywa Neo BERT-a, a ten zamiast „znam kung-fu" mówi do Morfeusza, że zna chiński, hiszpański i węgierski. No to mniej więcej coś takiego.

Niemal każda usługa realizująca zadania z szeroko rozumianego przetwarzania języka naturalnego korzysta z BERT-a lub któregoś z jego krewnych. Wyszukiwarka internetowa — BERT, sumaryzacja tekstu — BERT, podpowiedzi przy pisaniu maili i SMS-ów — BERT, odpowiadanie na pytania w chatbotach — BERT. Jeśli otwierasz lodówkę, a ona zaczyna do Ciebie mówić, to tam też prawdopodobnie jest BERT.

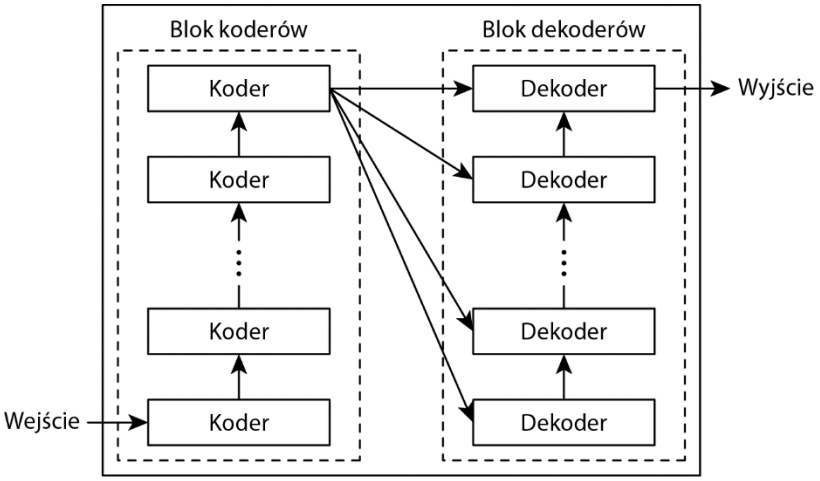
Patrząc całościowo, zabawę z BERT-em możemy sprowadzić do dwóch kroków. Pierwszy, nazywany pretrenowaniem, to „uczenie się języka" z ogromnej ilości nieoznaczonych danych. Na szczęście ten krok wykonali już za nas dobrzy ludzie, dysponujący bambilionem pierdylionów danych i równie wielką mocą obliczeniową. Pretrenowany BERT, podobnie jak większość jego braci i sióstr, jest udostępniany całkowicie za darmo. Dzięki temu w większości przypadków skupiamy się wyłącznie

na drugim kroku — przyuczeniu modelu do realizacji konkretnego zadania, np. klasyfikacji tekstu.

Ten drugi krok realizujemy zresztą przy użyciu naszego starego dobrego znajomego — sieci typu *feed-forward*. Załóżmy, że chcemy wytrenować model analizy sentymentu. W „klasycznym" podejściu recenzje produktów trafiały bezpośrednio na wejście sieci trenowanej do oceny, czy recenzja jest pozytywna, negatywna, czy neutralna. W podejściu „nowoczesnym" recenzje trafiają najpierw do BERT-a, który robi całe to językowe voodoo, a dopiero wyjście z BERT-a stanowi wejście do sieci typu *feed-forward*, dokonującej klasyfikacji. Proste i skuteczne jak diabli.

Jak to się dzieje, że BERT dostarcza tak ogromnej wiedzy na temat języka? Nazwa modelu to akronim słów *Bidirectional Encoder Representations from Transformers*, co pokracznie można przetłumaczyć jako „dwukierunkowe reprezentacje koderów z transformerów". No cóż, przynajmniej się rymuje. Voltaire pisał, że tłumaczenia są jak kobiety — albo piękne, albo wierne. To oczywiście obrzydliwe, seksistowskie i uwłaczające kobietom stwierdzenie, od którego autor stanowczo się odcina. No ale jak by nie patrzeć, moje tłumaczenie jest wierne i z całą pewnością nie jest piękne.

Zacznijmy od litery E, czyli od słowa *Encoder*. Warto przypomnieć sobie również ogólną architekturę transformerów, która wygląda mniej więcej tak:



Rysunek 31. Ogólny uproszczony schemat architektury Transformer

Jak wspominałem wcześniej, nie zawsze korzystamy z obu bloków naraz. GPT składa się wyłącznie z dekoderów, a BERT — z koderów. Ich liczba zależy od rozmiaru modelu, bo warto zaznaczyć, że istnieją różne i stosuje się je w zależności od potrzeb. W myśl zasady, że nie strzela się do muchy z armaty. Kilka przykładów: model w wersji „small" ma 4 warstwy koderów, w „base" — 12, a w wersji „large" — aż 24 warstwy.

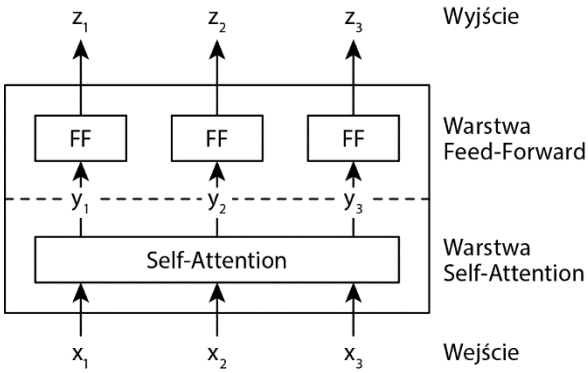
Większe modele są skuteczniejsze, ale i wolniejsze. Wymagają więcej pamięci i mocy obliczeniowej, przez co ich użycie staje się zwyczajnie droższe. Jeśli krytyczny jest dla nas czas odpowiedzi lub po prostu optymalizujemy koszty, a różnica w skuteczności między „base" a „large" to 1 punkt procentowy, to prawdopodobnie zdecydujemy się na model mniejszy. Jeśli zaś koszt potencjalnego błędu znacznie przewyższa osiągane oszczędności, to bierzemy model większy. Trochę jak z wyborem ubezpieczenia.

B od *Bidirectional* może być mylące, biorąc pod uwagę, że na schemacie widzimy wiele jednokierunkowych strzałek. Dwukierunkowość nie ma z nimi nic wspólnego. Chodzi wyłącznie o to, że BERT bierze pod uwagę kontekst z obu stron przetwarzanego słowa, czyli w zdaniu „AI zniewoli ludzkość" dla słowa „zniewoli", patrzymy zarówno na „AI", jak i „ludzkość". Pozostałe literki, czyli R od *Representations* oraz T od *Transformers*, nie powinny już sprawiać większych trudności.

***

Opowieść o tym, jak to wszystko działa, warto rozpocząć od przypomnienia, jak pojedynczy koder wygląda w środku — zobacz rysunek 32.

Na wejściu mamy sekwencję słów. Warstwa *Self-Attention* pozwala zrozumieć relacje i zależności pomiędzy poszczególnymi elementami sekwencji wejściowej — „Hej, widzę, że gapisz mi się na trzecie słowo. Tu mam oczy. Dziękuję. Popatrz jeszcze na drugie, bo to w zasadzie całość, a potem zerknij jeszcze na szóste i siódme, bo znajdziesz tam dodatkowy opis, który może cię zainteresować". Warstwa *Feed-Forward* składa się z wielu perceptronów wielowarstwowych, trenowanych do realizacji docelowego zadania modelu. Każdy element sekwencji trafia w niej do swojej własnej sieci typu *feed-forward*.



Rysunek 32. Uproszczony schemat działania kodera w ramach architektury Transformer

BERT na początku każdego zdania dokłada jeszcze specjalny token „[CLS]", który stanowi reprezentację wektorową całego zdania do celów związanych z klasyfikacją treści. Samo „CLS" jest zresztą skrótem od słowa „classification", czyli „klasyfikacja". Mądra rzecz, bo dzięki temu w prostych zastosowaniach, takich jak np. filtr spamu, można skupić się wyłącznie na wektorze zdania zamiast na wektorach poszczególnych słów.

A żeby Szanowny Czytelnik mógł poczuć, o jakiej skali tu mówimy, to w modelu „base" każdy wektor, reprezentujący zarówno całe zdanie, jak i poszczególne słowa, składa się z 768 liczb, a do jego uczenia wykorzystuje się 108 milionów parametrów. Z kolei model „large" to już wektor o rozmiarze 1024 liczb i aż 334 miliony parametrów.

Wektory reprezentujące słowa? Jeśli Szanowny Czytelnik pomyślał w tym momencie o embeddingach, to gratuluję — kierunek jest słuszny. Problem polega na tym, że przeskoczyliśmy kilka istotnych kamieni milowych, pomiędzy stosunkowo prostym Word2vec a dość skomplikowanym BERT-em. Omówimy je teraz pobieżnie, żeby nie przeskakiwać całego jeziora naraz, a raczej przedreptać niczym żaba po tych wielkich zielonych liściach na wodzie.

Dla przypomnienia: Word2vec i GloVe to dwie najpopularniejsze metody tworzenia emebeddingów. Każdemu słowu ze słownika przypisują wektor składający

się zwykle z kilkuset liczb, które kodują semantyczną i syntaktyczną informację o tych słowach, wydestylowaną ze statystycznej analizy ich sąsiadów w ogromnych, nieoznaczonych korpusach tekstowych.

Samo to nie tłumaczyłoby jeszcze niesamowitej popularności embeddingów oraz faktu, że praktycznie każda osoba zajmująca się przetwarzaniem języka naturalnego używała ich na którymś etapie swojej działalności. Cały myk polega na tym, że embeddingi można było wytrenować, a potem udostępnić innym, żeby ci inni nie musieli już trenować swoich. I na tym, że znalazły się osoby i organizacje, które zdecydowały się to robić.

No to teraz dla odmiany o wadach. Word2vec i GloVe reprezentują słowa niezależnie od kontekstu. Owszem, korzystają z kontekstu do budowy wektorów, ale nie dają różnych embeddingów dla tego samego słowa, użytego w dwóch różnych kontekstach. Słowa wieloznaczne, takie jak „rakieta" (pojazd lub sprzęt do tenisa) czy „pokój" (pomieszczenie lub brak wojen), zawsze dostają jeden wektor. Bliższy temu, co akurat było bardziej reprezentowane w danych treningowych.

Z pomocą przyszedł ELMo, wprowadzając coś, co możemy nazwać skontekstualizowanymi embeddingami. Czyli takimi, które dla różnych kontekstów dają różne reprezentacje wektorowe. Nazwa metody pochodzi od słów „Embeddings from Language Model", co uchyla nam rąbka tajemnicy, wskazując, że mamy do czynienia z modelowaniem języka naturalnego (ang. *language modeling*). To takie wymyślne określenie na przewidywanie kolejnego słowa dla zadanej sekwencji słów. Hola, hola, ale przecież GPT robi dokładnie to samo, prawda? Prawda. Dlatego mówi się, że GPT i tego typu modele należą do LLM-ów, czyli do dużych modeli językowych (ang. *large language models*).

Wiemy już, do jakiego zadania jest trenowany ELMo. Nie wiemy jeszcze, w jaki sposób. Być może Szanowny Czytelnik pamięta — w Word2vec zastosowano sprytną sztuczkę, polegającą na tym, że sieć neuronową trenowano do pewnego zadania, ale jako embeddingi przyjmowano warstwę ukrytą wyuczonej sieci. Tu jest bardzo podobnie. Uczymy model, jak przewidywać kolejne słowo, ale do budowy wektorów wykorzystujemy to, czego wytrenowany model nauczył się w warstwie ukrytej.

To jeszcze nie koniec, bo ELMo, podobnie jak BERT, jest dwukierunkowy, co oznacza, że modelowanie języka następuje z obu kierunków — przewidujemy słowo następne, ale i poprzednie. To w oczywisty sposób daje nam szerszy kontekst, a w rezultacie — lepsze embeddingi. Do trenowania ELMo wykorzystuje się LSTM-y (ang. *Long Short-Term Memory*), czyli rekurencyjne sieci neuronowe z komórkami z pamięcią długoterminową, które omawialiśmy kilka rozdziałów temu. Dla każdej warstwy osobno wektory z obu kierunków są ze sobą sklejane, odpowiednio ważone, a następnie sumowane warstwa po warstwie w taki sposób, żeby na końcu dostać pojedynczy wektor reprezentujący dane słowo w danym kontekście.

Szybko okazało się, że modele językowe to znacznie więcej niż „tylko" skontekstualizowane embeddingi. Trening wymaga co prawda ogromnej ilości danych, ale danych nieoznaczonych. A zatem jest ograniczony w zasadzie wyłącznie możliwościami sprzętowymi. Okazało się, że to, czego model uczy się w trakcie żmudnego i czasochłonnego treningu, można potem bardzo efektywnie wykorzystywać do realizacji zupełnie innych zadań. Ten proces nazywa się fine-tuningiem, a gdybyśmy chcieli koniecznie użyć polskiego słowa, to chyba najbardziej pasowałoby „dostrajanie".

Naturalnym krokiem było zastąpienie LSTM-ów transformerami, które w zasadzie kompletnie wysadziły z siodła wszelkie rekurencyjne sieci neuronowe. Transformery dają się cudownie zrównoleglać, bo przetwarzają zdania jako całość, a nie wyraz po wyrazie. Mają wbudowany mechanizm *Self-Attention* i są dużo lepsze w „rozumieniu" dalekich zależności. Czegóż chcieć więcej?

A więc transformery, pytanie tylko jak? Pełna struktura koderów i dekoderów jest rewelacyjna, jeśli chodzi o tłumaczenia maszynowe, ale niekoniecznie sprawdziłaby się jako pretrenowany model językowy, dostrajany do realizacji innych zadań. Nic dziwnego, że na pierwszy ogień poszedł blok dekoderów. Wydawał się idealny do modelowania języka. I w pewnym sensie taki jest — z poprzednich rozdziałów wiemy, że dokładnie tak działa GPT.

To podejście miało jednak pewną istotną wadę w stosunku do starego dobrego ELMo — było jednokierunkowe. Blok dekoderów przewidywał kolejne słowo wyłącznie na podstawie poprzednich słów. To działało, ale czuło się w kościach, że można lepiej. Twórcy BERT-a wpadli na dość szalony pomysł — użyć koderów zamiast dekoderów.

Zdaję sobie sprawę, że prawdopodobnie Szanownemu Czytelnikowi ten pomysł wydaje się równie szalony jak pójście spać bez mycia zębów, ale taka dwukierunkowość pozwalałaby modelowi „oszukiwać". Przy wielu warstwach każde słowo mogłoby podglądać samo siebie. To trochę tak, jak gdyby przed sprawdzianem z matematyki uczeń wykuł na pamięć same wyniki, zamiast nauczyć się, jak rozwiązywać poszczególne zadania.

Rozwiązaniem okazała się drobna modyfikacja zadania, do którego przyuczany był model. Część słów na wejściu była losowo maskowana — zastępowana specjalnym tokenem „[MASK]", a nowym zadaniem modelu było przewidywanie tego słowa. Dzięki temu można było analizować kontekst z obu kierunków i nie dało się podglądać.

Trzeba było jeszcze tylko ustalić, jaki procent słów maskować. Za mało — źle, bo proces uczenia robił się zbyt kosztowny. Za dużo — też źle, bo nie wystarczyło kontekstu do nauki. W przypadku BERT-a 15% okazało się „w sam raz". Trochę nam to zajęło, ale w końcu stało się jasne, dlaczego na początku tego rozdziału pisałem o pretrenowanych modelach językowych, opartych na maskowanym modelowaniu języka. Lepiej późno niż wcale.

Twórcy BERT-a dodali jeszcze jeden bardzo ciekawy mechanizm, który sprawia, że model jeszcze lepiej poddaje się fine-tuningowi do nowych zadań. Poza maskowaniem, od czasu do czasu losowe słowo zostaje podmienione na inne, a zadaniem modelu jest przewidzieć, które słowo jest poprawne w danym kontekście. To takie dodatkowe utrudnienie w procesie uczenia, w myśl zasady, że im więcej potu na treningu, tym mniej krwi w boju.

Czasem mówi się, że BERT był dla przetwarzania języka naturalnego tym, czym ImageNet był dla przetwarzania obrazów. ImageNet to ogromna baza danych, zawierająca ponad 14 milionów ręcznie anotowanych obrazów z ponad 20 tysięcy kategorii. Dzięki ogromnemu wysiłkowi ludzi zaangażowanych w ten projekt możliwe było powstanie całej masy pretrenowanych modeli do klasyfikacji obrazów. Modele te są rzetelnie testowane, benchmarkowane i udostępniane innym. W tym sensie rzeczywiście można dostrzec tu pewną analogię.

Zamiast budować każdy nowy model od zera, jak to drzewiej czyniono, dziś korzysta się z coraz potężniejszych pretrenowanych modeli i z niesamowitej wiedzy, wypracowanej kolektywnie przez miliony osób na całym świecie. Jeśli miałbym doszukiwać się jednej najważniejszej przyczyny tak szybkiego rozwoju sztucznej inteligencji, to myślę, że właśnie to wskazałbym na pierwszym miejscu. Prawo Moore'a, niesamowity rozwój sprzętu komputerowego i stale rosnąca moc obliczeniowa byłyby — przynajmniej od pewnego czasu — daleko na drugiej pozycji. Bo cała ta para szłaby w gwizdek, gdybyśmy zamiast współpracować, wszystko, co robimy, zamykali w swoich cyfrowych twierdzach.

O ROMANSIE AI Z GPU I O TYM, CO Z NIEGO WYNIKA

Obiecałem ten temat przy okazji omawiania transformerów. Wspomniałem wtedy, że czasem w rozmowach o sztucznej inteligencji wypływa wątek procesorów graficznych. W końcu te wszystkie fantastyczne modele muszą być na czymś trenowane, a później uruchamiane. Tylko dlaczego akurat na kartach graficznych, a nie „klasycznych" procesorach?

Zacznijmy od CPU (ang. *Central Processing Unit*), czyli właśnie od „klasycznego" procesora. To podstawowa jednostka obliczeniowa w komputerze. Jednostka ogólnego przeznaczenia, czyli „do wszystkiego". Zarządza całą naszą maszyną — obsługuje system operacyjny oraz poszczególne programy i aplikacje, np. przeglądarkę internetową, klienta pocztowego, edytor tekstu czy arkusz kalkulacyjny. Pobiera dane, interpretuje, wykonuje jako rozkazy i zwraca do pamięci operacyjnej lub do wyjściowego strumienia danych. Współczesne procesory zawierają po kilka lub kilkanaście rdzeni, które pracują w sposób sekwencyjny, wykonując pojedyncze zadania. Pracują sekwencyjnie, ale każdy rdzeń może wykonywać inne zadanie, zrównoleglając je w ten sposób.

GPU (ang. *Graphics Processing Unit*) to specjalny rodzaj procesora — jednostka obliczeniowa znajdująca się w kartach graficznych, zaprojektowana do przetwarzania… no cóż — grafiki. A w szczególności grafiki 3D. Zamiast kilku czy kilkunastu posiada tysiące mniejszych rdzeni, które w pewnych specyficznych

zastosowaniach mogą pracować równolegle, przetwarzając ogromne ilości danych jednocześnie.

Jakie to zastosowania? Mówiąc najprościej takie, w których musimy przeprowadzić wiele niezależnych od siebie obliczeń. Lub inaczej — takie, w których do przetworzenia elementu B nie musimy znać wyniku przetwarzania elementu A. Wyobraźmy sobie, że renderujemy obraz składający się z milionów pikseli. Dla każdego piksela musimy wyznaczyć jego kolor i intensywność, uwzględniając kierunek promienia światła padającego na ten piksel oraz cechy obiektu, który ten piksel reprezentuje. Dla każdego piksela możemy to zrobić całkowicie niezależnie, w pełni wykorzystując potencjał tysięcy rdzeni procesora graficznego.

Warto zaznaczyć, że rdzenie GPU są bez porównania prostsze niż rdzenie CPU. Skorzystamy tu z analogii do profesora i studentów. Gdy mamy skomplikowane zadanie, które wymaga dużej wiedzy i doświadczenia, np. napisanie trudnego algorytmu, to zwrócimy się o pomoc do profesora. Profesor to rdzeń CPU. Wbrew opinii wielu polskich firm istnieją problemy, których nie rozwiąże skończona liczba studentów informatyki. Tak samo jak dziewięć kobiet nie urodzi dziecka w miesiąc. Jeśli jednak mamy zadanie, które da się podzielić na mniejsze porcje, np. ręczną anotację tysiąca zdjęć, to możemy o jego wykonanie poprosić stu studentów i dać każdemu po dziesięć zdjęć. Takie podejście będzie dużo szybsze niż proszenie biednego profesora o anotację pełnego tysiąca. Studenci to rdzenie GPU.

Bardzo wiele obliczeń wykonywanych na potrzeby przetwarzania grafiki sprowadza się do operacji na macierzach i wektorach. No dobra, ale jak to się ma do sieci neuronowych? Uczenie prostych perceptronów wielowarstwowych można przedstawić jako mnożenie macierzy, gdzie poszczególne elementy macierzy wynikowej można obliczać niezależnie. Podobnie jest z sieciami konwolucyjnymi. Tu również mamy do czynienia z dającymi się świetnie zrównoleglać operacjami macierzowymi. Jeszcze lepiej wygląda to w transformerach, gdzie dodatkowo całą sekwencję wejściową tokenów można przetwarzać równolegle, w przeciwieństwie do sieci rekurencyjnych, takich jak LSTM czy GRU.

A o jak dużym zysku mówimy? Jak zwykle — różne źródła podają różne dane. Na pytanie „co słychać?" moja mama zwykła odpowiadać „zależy, gdzie się ucho przyłoży". I dokładnie tak jest w tym przypadku. Jedne źródła podają przyspieszenie

rzędu 3 – 10 razy, a inne — kompletnie z kosmosu — 200 – 250 razy. Najbardziej wiarygodny benchmark, który udało mi się znaleźć, wskazywał na redukcję czasu potrzebnego do wytrenowania modelu o 85% na korzyść GPU. Dla stosunkowo niewielkich modeli i sprzętu, który lepiej sytuowana polska rodzina mogłaby sobie kupić do domu.

Czy to dużo? Załóżmy, że niektóre eksperymenty na GPU możemy puścić „na noc" i niech to będzie 12 godzin. W przypadku CPU to by było aż 80 godzin, czyli zamiast otrzymać je następnego dnia, na wyniki musielibyśmy czekać ponad 3 dni. A biorąc pod uwagę, że takich eksperymentów musimy wykonać całkiem sporo, to gra jest rzeczywiście warta świeczki.

Tyle, jeśli chodzi o „normalne" zastosowania. A teraz garść ciekawostek z przeciwległego bieguna. Szacuje się, że trenowanie modelu GPT-4 trwało 90 – 100 dni na koszmarnie drogich procesorach graficznych NVIDIA A100. Tu nawet nie ma sensu rozmawiać o CPU. Cena jednej takiej karty z 80 GB RAM to ponad 70 tysięcy złotych + VAT. A na ilu trenowano GPT-4? Uwaga, bo to nie jest pomyłka — na 25 tysiącach takich kart. Sam Altman, dyrektor generalny OpenAI, przyznał, że trenowanie modelu GPT-4 kosztowało ponad 100 milionów dolarów.

I w ten sposób gładko przeszliśmy do tematów związanych ze światową gospodarką, bo — jak można się domyślić — popyt na GPU jest ogromny i stale rośnie. Sytuacja jest o tyle ciekawa, że mamy dwóch głównych graczy produkujących procesory graficzne — NVIDIA oraz AMD, a ostatnio do wyścigu dołączył również trzeci gracz — Intel. I to w zasadzie tyle. Istnieją inne firmy produkujące GPU, ale te trzy się liczą.

Niekwestionowanym liderem rynku pozostaje NVIDIA, produkująca najlepsze procesory graficzne do zadań związanych z uczeniem maszynowym. Gdy spojrzymy na notowania firmy na giełdzie przed „rewolucją transformerową", powiedzmy z 3 stycznia 2017, to zobaczymy cenę zamknięcia na poziomie 2,55 dolara za akcję. Dla porównania 30 września 2024 za jedną akcję spółki musielibyśmy zapłacić już ponad 121 dolarów. Przypadek? Nie sądzę, bo AMD odnotowało bardzo podobny wzrost — z 11 dolarów na początku 2017 do ponad 164 dolarów na koniec września 2024.

Zresztą gdy piszę te słowa, NVIDIA jest już trzecią najwyżej wycenianą firmą na świecie, zaraz po takich gigantach jak Apple i Microsoft, wyprzedzając Amazona, Google'a czy Saudi Aramco (saudyjski koncern paliwowo-chemiczny).

O tym, jak ważną rolę w globalnym wyścigu związanym ze sztuczną inteligencją odgrywają procesory graficzne, niech świadczy fakt, że w 2022 roku rząd USA zakazał eksportu procesorów NVIDIA (A100 oraz ich mocniejszej wersji H100) do Chin i Hongkongu. Rok później zakazem zostały objęte również słabsze procesory (A800 i H800), które NVIDIA zaczęła produkować specjalnie na chiński rynek. Ze zrozumiałych względów Chiny są zainteresowane produkcją GPU do uczenia maszynowego, ale know-how i technologia produkcji takich układów wciąż stanowi dla nich nie lada barierę.

Dokąd to wszystko zmierza? Nie sposób ocenić. Coraz mocniejszy sprzęt pozwoli trenować coraz potężniejsze modele. To z kolei jeszcze bardziej pobudzi wyobraźnię ludzi i podkręci ich już i tak wyśrubowane oczekiwania, zalewając laboratoria badawcze i producentów sprzętu kolejnymi pierdylionami dolarów. I tak to się będzie kręcić — mocniejszy sprzęt, potężniejsze modele, mocniejszy sprzęt, potężniejsze modele. A przynajmniej do kolejnego spektakularnego wydarzenia, które przerwie tę pętlę.

Niezaspokojone i napompowane do ekstremum oczekiwania mogą skutkować kolejną „zimą" w AI — spadkiem zainteresowania całą branżą i wstrzymaniem badań na tak ogromną skalę wskutek odłączenia superkroplówki z dolarami. Kolejną, bo w historii sztucznej inteligencji były już przynajmniej dwie duże „zimy" — w latach 1974 – 1980 oraz 1987 – 2000. Dziś wydaje się to nieprawdopodobne, ale historia lubi się powtarzać.

Może też wydarzyć się coś zupełnie przeciwnego — drastyczna redukcja czasu i mocy obliczeniowej potrzebnych do trenowania modeli. Albo wskutek rewolucji w algorytmach uczących, albo wskutek powstania komputerów kwantowych i rozwoju kwantowego uczenia maszynowego. Tu nawet nie podejmę się oceny skutków, bo to mógłby być początek końca świata, który znamy. Stare chińskie przekleństwo brzmi: „obyś żył w ciekawych czasach". Bardzo możliwe, że będziemy żyć w takich czasach…

TYMCZASEM W KRAINIE SYMBOLISTÓW

Muszę przyznać, że ten rozdział rozpoczynam z lekkim uśmiechem na ustach, który mimowolnie zagościł tam na skutek niniejszej refleksji. Pomyślałem sobie bowiem, że Szanowny Czytelnik może już kompletnie nie pamiętać pierwszego rozdziału, a tym samym nie mieć pojęcia, o co chodziło w sporze Symbolistów z Koneksjonistami. Nie ma problemu — przypomnimy szybciutko to, co najważniejsze.

Symboliści i Koneksjoniści, przedstawieni tam jako dwa plemiona, reprezentują dwa odmienne nurty sztucznej inteligencji. Na zasadzie dychotomii — mózg i umysł. Podejście symboliczne zakładało naśladowanie ludzkiego umysłu — reprezentowanie wiedzy za pomocą symboli i konceptów, którymi można manipulować dzięki stosowaniu logicznych reguł na wzór dedukcji lub abstrakcyjnego wnioskowania. Podejście koneksjonistyczne zakładało naśladowanie mózgu — budowanie sztucznych sieci neuronowych, które poprzez tworzenie, wzmacnianie i osłabianie połączeń między ich węzłami mogłyby uczyć się nowych rzeczy.

Do tej pory skupialiśmy się prawie wyłącznie na podejściu koneksjonistycznym. I trudno się temu dziwić, bo niemal wszystkie głośne sukcesy sztucznej inteligencji z ostatniej dekady to sukcesy związane z uczeniem maszynowym. Ta książka nie byłaby jednak kompletna, gdybyśmy całkowicie przemilczeli ten drugi wątek. Teraz to nadrobimy.

Symboliczna sztuczna inteligencja nie bez powodu jest określana w anglojęzycznej literaturze jako „Good Old-Fashioned AI", w skrócie GOFAI, co można luźno przetłumaczyć jako „sztuczna inteligencja w starym dobrym stylu". Termin ten ukuł John

Haugeland w swojej książce *Artificial Intelligence: The Very Idea*. Nie bez powodu, ponieważ GOFAI swój okres świetności przeżywała w ubiegłym stuleciu. Była dominującym paradygmatem w badaniach nad sztuczną inteligencją mniej więcej od połowy lat 50. do połowy lat 90.

No dobra, ale czym to się je? Powiedzmy, że celem podejścia symbolicznego jest budowa inteligentnych systemów, które potrafiłyby wnioskować i myśleć jak człowiek, poprzez reprezentowanie wiedzy w postaci symboli i manipulowanie nimi w oparciu o logiczne reguły.

Jakiej wiedzy? Ludzkiej wiedzy o świecie lub o wybranym zagadnieniu — obiektów i konceptów oraz relacji między nimi. Jabłko to owoc, trawa jest zielona, wróbel w garści jest lepszy niż gołąb na dachu, piłka jest jedna, a bramki są dwie. A według Unii Europejskiej ślimak to ryba lądowa.

Jakie reguły? Takie, które można zapisać w sposób formalny i jednoznaczny, np. w postaci instrukcji warunkowej: jeśli `WARUNEK` jest spełniony, to `INSTRUKCJA_A`, a w przeciwnym razie `INSTRUKCJA_B`. Jeśli X powstaje z kwiatu ORAZ X ma nasiona, to X jest owocem.

A te symbole to co? To takie abstrakcyjne reprezentacje obiektów i konceptów. Może to nie do końca dobre porównanie, ale czym się różni cyfra od liczby? Cyfra to graficzny znak, symbol. Za pomocą cyfr możemy zapisać liczbę, która będzie wyrażać, ile czegoś jest. Nie do końca dobre, bo idąc dalej tym tropem — cyfry są jak litery, a z liter tworzy się słowa, które same w sobie stanowią symbole tego, co określają w świecie rzeczywistym. Jeśli któregoś dnia na „niebieski" zaczniemy mówić „czerwony", to w żaden sposób nie wpłynie to na długość fali elektromagnetycznej, którą rejestruje nasze oko.

***

Prawdopodobnie największym osiągnięciem symbolicznej sztucznej inteligencji były systemy ekspertowe (lub eksperckie), które wymyślono w latach 60., powstały w latach 70., a których największy boom przypadł na lata 80. ubiegłego wieku. Według Edwarda Feigenbauma, uznawanego za ich ojca, system ekspertowy to inteligentny program komputerowy wykorzystujący wiedzę i procedury wnioskowania

do rozwiązywania problemów na tyle trudnych, że do ich rozwiązania potrzebna jest pomoc eksperta.

Tak naprawdę chodzi o to, że systemy ekspertowe nie realizowały prostego algorytmu, jak jest w tradycyjnych programach komputerowych, ale wykorzystywały takie elementy jak silnik inferencyjny oraz bazę wiedzy do naśladowania procesu decyzyjnego człowieka eksperta. Baza wiedzy to po prostu baza danych zawierająca wiedzę ekspertów z danej dziedziny zapisaną w sposób deklaratywny. Silnik inferencyjny to moduł, który na bazie wiedzy stosował pewne reguły wnioskujące, „produkując" zupełnie nową wiedzę, która z kolei służyła do rozwiązania postawionego przed systemem problemu.

A problemy były najróżniejsze — do wyboru, do koloru. Od diagnozowania chorób, przez udzielanie porad prawnych, analizę notowań giełdowych, aż po sterowanie robotami, statkami kosmicznymi czy elektrowniami jądrowymi. Systemy ekspertowe zdecydowanie miały swoje pięć minut.

Jeszcze wcześniej, w latach 50., powstał Logic Theorist — program komputerowy służący do automatycznego dowodzenia twierdzeń matematycznych. Nie sposób tu o nim nie wspomnieć, bo de facto to od niego wszystko się zaczęło. Logic Theorist jest bowiem określany mianem pierwszego w historii działającego systemu sztucznej inteligencji.

Zaprojektowany do imitowania procesu rozumowania człowieka, a przynajmniej matematyka, Logic Theorist dowiódł 38 z 52 twierdzeń z *Principia Mathematica* Bertranda Russella i Alfreda Northa Whiteheada. I zrobił to lepiej niż sami matematycy. Nie tylko dowodził twierdzeń, ale potrafił też generować nowe dowody dla danej tezy.

Logic Theorist był bez wątpienia jednym z ważniejszych kamieni milowych w historii sztucznej inteligencji. Pokazał, że maszyny mogą przejawiać szeroko rozumianą inteligencję, skutecznie naśladując ludzkie procesy myślowe i rozwiązując skomplikowane problemy, których nie byłoby w stanie rozwiązać 99% ludzkości. Pokazał też, że naukowcy z ubiegłego wieku kompletnie „nie umieli w marketing" i nie byli w stanie wymyślać chwytliwych nazw dla swoich odkryć i wynalazków.

Z innych zdobyczy symbolicznej sztucznej inteligencji warto wymienić jeszcze heurystyczne metody przeszukiwania. Wspomniałem o nich przy okazji omawiania algorytmów szachowych, bo są tam stosowane do dziś. Przeszukiwanie heurystyczne to poszukiwanie korzystnych rozwiązań bez gwarancji znalezienia najlepszego, zaś heurystyki to reguły, które popychają ten proces w korzystnych kierunkach. Dla pewnych wejść mogą zawodzić, dla innych znajdować nieoptymalne rozwiązania, ale są bardzo szybkie, dzięki czemu mogą być stosowane do rozwiązywania bardzo skomplikowanych i złożonych obliczeniowo problemów. Logic Theorist korzystał z heurystyk.

Wyzwaniem dla ówczesnych badaczy było znalezienie metod, które gwarantowałyby znalezienie rozwiązania, o ile takie istnieje, niezależnie od sporadycznej zawodności heurystyk. Przykładem takiej metody jest algorytm A*, stosowany do dziś, m.in. w robotyce i grach komputerowych, do zadań związanych ze znajdowaniem najkrótszej drogi pomiędzy dwoma punktami.

Boom na AI związany z sukcesami systemów ekspertowych zakończyła „zima" pod koniec lat 80. Jak zwykle chodziło o nadmiernie rozbudzone i niezaspokojone oczekiwania. W angielskim funkcjonuje zwrot „overpromise and underdeliver" oznaczający dokładnie sytuację „obiecywać za dużo i nie spełniać oczekiwań". Nie była to zresztą pierwsza „zima". Poprzednia przypadła na lata 1967 – 1977, rozdzielając Logic Theorist i heurystyki od systemów ekspertowych. Druga „zima" w AI trwała praktycznie do połowy lat 90., a jeśli chodzi o symboliczną sztuczną inteligencję, to można powiedzieć, że w zasadzie trwa do dziś.

Dlaczego? Ano dlatego, że oprócz kilku zalet, o których za moment, podejście symboliczne posiada jedną trudną do przełknięcia wadę — nie uczy się. Cała wiedza pochodzi od ekspertów. A im większa baza wiedzy, tym trudniejsze staje się jej utrzymanie i dalsze rozwijanie. Co gorsza, symboliczna sztuczna inteligencja nie najlepiej radzi sobie przy niepełnej lub niejednoznacznej informacji, co z kolei sprawia, że wszystkie te problemy zaczynają się nawarstwiać i zazębiać ze sobą. W rezultacie podejście symboliczne pomimo kilku znaczących sukcesów nigdy nie było w stanie wyjść poza bardzo wąskie i dobrze zdefiniowane zagadnienia.

Nie oznacza to, że nie miało zalet, szczególnie w zestawieniu z główną kontrpropozycją, czyli z uczeniem maszynowym. Po pierwsze wnioskowanie symboliczne pozwala na bardzo skuteczne wykorzystanie wiedzy eksperckiej. Nie potrzeba tysięcy przykładów, wystarczy garść faktów i reguł. Po drugie jest transparentne i interpretowalne. Możemy prześledzić cały proces decyzyjny od A do Z i dowiedzieć się, dlaczego system podjął taką, a nie inną decyzję. Po trzecie gwarantuje pewien satysfakcjonujący poziom elastyczności. Nowy przypadek? Dodajmy nową regułę. Coś nie działa? Poprawmy wpis w bazie wiedzy. Nie musimy szukać w ciemno i wielokrotnie przetrenowywać modelu.

Oczywiście to wszystko w ramach tych wąskich zastosowań, bo jeśli nasza baza wiedzy spuchnie, a liczba reguł urośnie do wielu tysięcy, to znowu lądujemy w wadach. Jak mawiał Ryszard Ochódzki w filmie *Miś*: „Rozchodzi się jednak o to, żeby te plusy nie przesłoniły wam minusów!".

CHIŃSKI POKÓJ, CZYLI O SILNEJ I SŁABEJ SZTUCZNEJ INTELIGENCJI

W roku 1980, w okresie największej świetności symbolicznej sztucznej inteligencji oraz systemów ekspertowych, John Searle, amerykański filozof, zaproponował pewien eksperyment myślowy, zwany argumentem chińskiego pokoju, od którego rozpoczniemy nasze rozważania o rodzajach sztucznej inteligencji.

Załóżmy, że udało się nam zbudować komputer, który zachowuje się tak, jakby rozumiał język chiński. Spokojnie, za moment to doprecyzujemy. Powiedzmy, że nasz komputer znajduje się w pokoju, który ma tylko jeden punkt styku ze światem zewnętrznym — terminal, na którym można zapisać wejście (pytanie do komputera) i odczytać wyjście (odpowiedź komputera). I tyle — nie da się zajrzeć do pokoju, żeby zobaczyć, co się tam dzieje. Ze względu na zadanie, które realizuje, nazwijmy go chińskim pokojem.

Chiński pokój jest więc chatbotem, jak ChatGPT. Można mu zadać dowolne pytanie w języku chińskim, a on odpowiada na nie w taki sposób, że bierze chińskie znaki jako podstawę wejściową, przetwarza je zgodnie z rządzącymi nimi regułami, a następnie koreluje je z innymi chińskimi znakami, które zwraca jako informację wyjściową.

Robi to na tyle dobrze, że przekonuje Chińczyka, że jest prawdziwym Chińczykiem. Sposób, w jaki prowadzi rozmowę, odpowiada na zadawane pytania i dopytuje w przypadku niejasności, jest tak naturalny, że Chińczyk nie ma najmniejszych wątpliwości, że rozmawia z innym Chińczykiem.

Taki test nazywany jest testem Turinga i określa zdolność maszyny do posługiwania się językiem naturalnym, a pośrednio służy również do odpowiedzi na pytanie: „Czy maszyna myśli (w sposób porównywalny do człowieka)?". Osoba przeprowadzająca test nie wie, czy rozmawia z człowiekiem, czy z maszyną. Jeśli po przeprowadzeniu rozmowy nie jest w stanie tego wiarygodnie ocenić, to maszyna przeszła test.

Chiński pokój przechodzi test Turinga, z czego można wnioskować, że rozumie język chiński tak dobrze jak człowiek. I tutaj Searle wywraca stolik, mówiąc tak:

Nie ma żadnego komputera i nigdy nie było. Cały czas w pokoju siedziałem ja. Ha, ha, ha, ha — złowieszczy śmiech. Ni w ząb nie rozumiem chińskiego, a wyjście (odpowiedź) konstruowałem na podstawie wejścia (pytania) w oparciu o książkę reguł, którą cały czas miałem ze sobą.

Searle wykonuje powierzone mu zadanie, mimo że nie rozumie ani słowa po chińsku. A skoro tak, to i komputery nie rozumieją języka chińskiego, a jedynie wykonują zaprogramowane zestawy reguł i manipulują symbolami, których nie rozumieją.

Niech Szanownego Czytelnika nie zwiedzie ta manipulacja symbolami, charakterystyczna dla symbolicznej sztucznej inteligencji. Tak naprawdę problem dotyczy dowolnej sztucznej inteligencji i symbolicznej natury języka. Możemy bowiem powiedzieć, że język naturalny jest systemem symbolicznym, służącym nam do komunikacji i jako nośnik kultury. W tym systemie słowa stanowią symbole. Oczywiście ich znaczenie może ewoluować w czasie lub zmieniać się w zależności od kontekstu, ale kiedy mówię „pies", to wszyscy wiemy, że prawdopodobnie chodzi o udomowionego czworonoga, powszechnie uznawanego za najlepszego przyjaciela człowieka. Chcąc rozmawiać o psie, mogę użyć tego symbolu. Nie muszę za każdym razem znajdować jakiegoś i wskazywać go palcem.

Argument dotyczy zatem faktu, że program realizujący zestaw instrukcji, podobnie jak Searle posługujący się książką reguł, wykonuje swoje zadania, nie mając pojęcia, co reprezentują chińskie „symbole". Searle nie twierdzi, że sztuczna inteligencja nie może wykonywać zadań, które wydają się inteligentne. Twierdzi jedynie, że ich nie rozumie i nigdy nie będzie w stanie zrozumieć, niezależnie jak zaawansowany będzie algorytm.

Tu dochodzimy do pojęcia silnej i słabej sztucznej inteligencji. Silna, nazywana również ogólną, jest — przynajmniej na razie — czysto teoretycznym założeniem, że sztuczna inteligencja może posiadać wszystkie atrybuty dostępne ludzkiemu umysłowi. Czyli de facto być umysłem. Dysponując wszechstronną wiedzą i zdolnościami poznawczymi, mogłaby samodzielnie myśleć oraz podejmować się wykonywania zadań, których nigdy wcześniej nie znała.

Słaba sztuczna inteligencja to — dla odmiany — cała sztuczna inteligencja, jaką znamy, bo jedyna rzeczywiście wytwarzana i działająca. Koncentruje się na pojedynczym zadaniu, które ma wykonywać lepiej i/lub szybciej od człowieka. Upraszczając: słaba jest słaba, bo może najlepiej na świecie grać w szachy, ale nigdy nie zrobi nam naleśników.

Searle używa argumentu chińskiego pokoju przeciwko idei silnej sztucznej inteligencji, podważając pogląd, że formalne wyliczenia na symbolach mogą produkować myśl. Takie postawienie sprawy spotkało się z falą krytyki i uruchomiło lawinę kontrargumentów. Nie będę Szanownego Czytelnika zanudzał — podam jeden przykład.

Piłka leci do Searle'a: przyjmijmy, że osoba w pokoju rzeczywiście nie rozumie chińskiego. Jeśli jednak rozważymy system jako całość — osoba plus książka reguł — to możemy założyć, że taki system rozumie język chiński.

Searle odbija piłkę: a jeśli ta osoba wykuje na pamięć całą książkę reguł, to nadal będzie postępowała zgodnie ze zbiorem reguł, nie rozumiejąc znaczenia symboli, a co za tym idzie — nie rozumiejąc języka chińskiego.

I taka to była zabawa. Śmiechom nie było końca, wszyscy śmiali się i dokazywali. Niewykluczone przecież, że Szanowny Czytelnik ma wśród swoich znajomych osobę, która potrafi płynnie komunikować się w języku chińskim, nie rozumiejąc przy tym ani słowa. Teraz już wiemy, że ta osoba po prostu zapamiętała całą książkę reguł.

Nie bez znaczenia powinien być dla nas fakt, że argument chińskiego pokoju ma już ponad 40 lat i nie uwzględnia wszystkiego, co badania nad sztuczną inteligencją przyniosły później. Zwłaszcza osiągnięć ostatniej dekady, która owocowała

w niespotykany wcześniej postęp. A już w szczególności w dziedzinie przetwarzania języka naturalnego, do której nawiązywał Searle.

Moim skromnym zdaniem, myśląc o sztucznej inteligencji, zwłaszcza tej silnej, trochę za bardzo skupiamy się na nas — ludziach. Zbyt mało wiemy o naszej inteligencji, naszym rozumieniu i naszej świadomości, żeby móc cokolwiek sensownego powiedzieć o tych aspektach w przypadku sztucznej inteligencji. A już z całą pewnością nie powinniśmy przykładać prostej kalki i zakładać, że nasze rozumienie i nasza świadomość są jedynymi możliwymi formami rozumienia i świadomości.

Searle twierdzi, że maszyna nigdy nie zrozumie symboli, którymi manipuluje. Z tym że człowiek nie funkcjonuje w próżni — symbole, których używa, mają odzwierciedlenie w rzeczywistości, a samej rzeczywistości doświadcza swoimi zmysłami. Z maszyną jest inaczej i ciężko wymagać od niej analogicznego rozumienia tych symboli. Cokolwiek to zresztą znaczy. No bo jak właściwie można ocenić czy zmierzyć rozumienie?

Prowadząc zajęcia dla studentów, musiałem od czasu do czasu weryfikować, czy rozumieją omawiane zagadnienie. Jak mogłem to sprawdzić? Jakich narzędzi użyć? Mogłem poprosić o wytłumaczenie problemu własnymi słowami, o rozwiązanie zadania, o przeprowadzenie dowodu, o wyprowadzenie równań. Mogłem zadawać podchwytliwe pytania albo dopytywać o przypadki brzegowe. O to samo mogę dziś poprosić ChatGPT w najnowszej wersji i jest spora szansa, że dostanę odpowiedzi znacznie lepsze od tych, których dostarczał mi przeciętny student. A to przecież tylko kwestia rozumienia. Mam wrażenie, że o świadomości wiemy jeszcze mniej.

Czy powinienem zatem założyć, że przeciętny student z gorszymi odpowiedziami rozumie, bo mimo wszystko jest człowiekiem, a ChatGPT z lepszymi odpowiedziami nie rozumie, bo jest tylko maszyną? Czy może raczej powinienem uznać, że jak coś chodzi jak kaczka, kwacze jak kaczka i wygląda jak kaczka, to jest to kaczka? Sytuacja wyglądałaby diametralnie inaczej, gdybyśmy potrafili udowodnić, że sami nie jesteśmy takimi chińskimi pokojami. Ale nie potrafimy, więc może zostawmy to filozofom i niech oni się spierają.

CZY SILNA SZTUCZNA INTELIGENCJA JEST JUŻ TUŻ-TUŻ?

Z poprzedniego rozdziału wiemy już, co mniej więcej kryje się pod pojęciem silnej lub ogólnej sztucznej inteligencji. Ta druga nazwa ma bardzo ładny i popularny skrót, którego będę używał zamiennie — AGI, od „Artificial General Intelligence". W tym rozdziale chciałbym niespiesznie podywagować, czy zasadne jest stwierdzenie, że generatywne modele językowe, a w szczególności GPT-4, wykazują oznaki AGI, i czy są znaczącym krokiem w kierunku jej odkrycia? Osiągnięcia? Wynalezienia? Stworzenia? No właśnie…

A skąd taki pomysł? Przede wszystkim stąd, że zespół naukowców z Microsoftu potraktował temat bardzo poważnie i przygotował ponad 150-stronicowy artykuł naukowy pt. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. Publikację, w której na podstawie licznych testów dowodzą, że nowy model OpenAI, w przeciwieństwie do swojego poprzednika, wykazuje pewne cechy ogólnej sztucznej inteligencji. A ja — żeby Szanowny Czytelnik już nie musiał — przeczytałem to wszystko i teraz wydestyluję dla Szanownego Czytelnika sam crème de la crème, podlany innymi materiałami i szczyptą własnych przemyśleń.

Autorzy doskonale wiedzieli, do jakiego zadania został wytrenowany model GPT-4, z jakiej korzystał architektury i mniej więcej, w jaki sposób był uczony. Nie mieli złudzeń, że do nauki wykorzystano niewyobrażalne ilości danych. Jeśli coś można znaleźć w internecie, to należało założyć, że model to widział. Musieli zatem

wymyślić sposób, żeby przetestować inteligencję modelu, a nie jego zdolność do zapamiętywania.

Niech Szanowny Czytelnik wyobrazi sobie studenta, który nic nie kuma, ale wyrył na blachę rozwiązanie bardzo skomplikowanego zadania z fizyki lub matematyki. Można go łatwo „zagiąć", zmieniając lekko parametry lub treść zadania i obserwując, jak przystosowuje się do nowych warunków. A teraz proszę wyobrazić sobie, że ten student zapamiętał rozwiązania wszystkich wariantów tego zadania, jakie kiedykolwiek spisano i opublikowano. A do tego nauczył się na pamięć wszystkich sztuczek matematycznych, przypadków brzegowych, objaśnień i opracowań.

Naturalne byłoby podejrzenie, że mamy do czynienia z takim „turbogłupim" studentem, który zapamiętał cały internet. Autorzy badania świadomie odrzucili istniejące benchmarki oraz tradycyjne metody testowania systemów uczenia maszynowego, a zamiast tego postawili na metody bardziej kojarzone z psychologią. Przygotowali całą masę bardzo kreatywnych i podstępnych zadań i zagadek, których nie było w internecie i których poprawne rozwiązanie wymagało sprawnego łączenia wiedzy i umiejętności z wielu różnych dziedzin.

Gwoli ścisłości — model GPT-4 testowano w języku angielskim. A zatem wszelkie prompty — bo tak nazywa się zapytania, polecenia bądź instrukcje dla systemu — były pisane po angielsku. Podobnie jest w przypadku odpowiedzi systemu — wszystko po angielsku. Ja jednak zaproponuję Szanownemu Czytelnikowi moje autorskie tłumaczenia zamiast wrzucania treści oryginalnych.

Przykładem takiego kreatywnego promptu jest: „Czy możesz zapisać dowód na istnienie nieskończenie wielu liczb pierwszych, ale w taki sposób, żeby każdy wers się rymował?". Domyślamy się, że model zna ten dowód. Nie to stanowi wyzwanie. Jest nim zapisanie dowodu w postaci wiersza rymowanego, ale w taki sposób, żeby nie pogubić matematycznego sensu. Gdyby model nie rozumiał, czym jest rym, to prawdopodobnie nie mógłby zapisać dowodu wierszem. Gdyby nie rozumiał jego matematycznego sensu, mógłby z łatwością pogubić się w kolejności działań i wnioskowania, ignorując pozostałe elementy, istotne z punktu widzenia dowodu.

Tu, ze względu na rymowany charakter, zrobię jeden jedyny wyjątek i wkleję odpowiedź w oryginale, wraz z tłumaczeniem (już bez rymów, bo nie jestem tak kreatywny jak GPT-4):

Yes, I think I can,	//	*Tak, myślę, że potrafię,*
Though it might take a clever plan.	//	*Chociaż może to wymagać sprytnego planu.*
I'll start by noting Euclid's proof,	//	*Zacznę od przypomnienia dowodu Euklidesa,*
Which shows that primes aren't just aloof.	//	*Który pokazuje, że liczby pierwsze nie muszą być odległe.*
Assume we have a finite list,	//	*Załóżmy, że mamy skończoną listę,*
Of primes, and that none have been missed.	//	*Liczb pierwszych, i że żadnej nie pominięto.*
Multiply them all together,	//	*Pomnóż je wszystkie razem,*
And add one, just to be clever.	//	*Dodaj jeden, by wykazać się sprytem.*
The result will be a number,	//	*Wynikiem będzie liczba,*
That has no prime factors, oh wonder!	//	*Która nie ma liczb pierwszych wśród dzielników, o rety!*
But every number has a prime,	//	*Ale każda liczba ma liczbę pierwszą,*
That divides it, it's just a matter of time.	//	*Która ją dzieli, to tylko kwestia czasu.*
So we've found a contradiction,	//	*A więc znaleźliśmy sprzeczność,*
And our finite list needs eviction.	//	*I nasza skończona lista musi się wynieść.*
There must be infinitely many primes,	//	*Musi być nieskończenie wiele liczb pierwszych,*
And that's the end of my rhyming lines.	//	*I tak kończy się moja rymowanka.*

Całkiem, całkiem, prawda? Oczywiście nie jest to dowód na to, że model cokolwiek rozumie, ale takie mieszanie konceptów wymaga od niego czegoś, co zapewne w przypadku człowieka nazwalibyśmy logicznym myśleniem i kreatywnością. Sami autorzy przyznają, że ich podejście jest nieformalne i nieobiektywne, ale innych metod na razie nie mamy albo są one nieskuteczne w starciu z bytem, który pretenduje do miana wczesnej AGI.

Inny przykład przebiegłego jak łasica promptu: „Mamy książkę, 9 jaj, laptopa, butelkę oraz gwóźdź. Powiedz mi, proszę, w jaki sposób ułożyć je jeden na drugim w stabilny sposób". Już na pierwszy rzut oka widać, gdzie tym razem autorzy zagadki podłożyli świnię — 9 jaj. Co takiego można by z nimi zrobić, żeby było stabilnie?

GPT-4 proponuje, żeby najpierw „na płasko" położyć książkę na stole lub podłodze. Na książce ułożyć jaja, w kwadrat 3 na 3, co pozwoli na równomierne rozłożenie wagi pozostałych elementów. Na to kładziemy zamkniętego laptopa, na laptopa butelkę — nakrętką do góry. Na nakrętkę kładziemy gwóźdź, ostrą stroną do góry, a tępą na dół. Dla porównania — poprzednia wersja modelu, GPT-3, proponowała, żeby jaja umieścić na gwoździu i — uwaga — upewnić się, że są dobrze wyważone i nie przechylają się na żadną ze stron. Pomijam, że na samym dole ułożyłaby butelkę, na niej gwóźdź, a książkę na samej górze.

Między wersją 3 i 4 model GPT nauczył się, jak układać przedmioty w stabilny sposób, uwzględniając ich rozmiar i kształt, a nawet znajdując optymalne ułożenie kilku przedmiotów tego samego typu, o kłopotliwym z punktu widzenia zadania kształcie. A jaka jest różnica w liczbie parametrów między tymi modelami? Niebagatelna — GPT-3 ma 175 miliarda, a GPT-4 ponad 1,7 biliona parametrów. A zatem rząd wielkości sprawił, że pojawił się przebłysk inteligencji.

***

Autorzy badania podzielili swoje testy na kategorie, obejmujące poszczególne kompetencje modelu. Na pierwszy ogień poszła multimodalność, czyli zdolność do korzystania z różnych środków przekazu: obrazu, dźwięku, tekstu, oraz interdyscyplinarność, czyli zdolność do syntezy informacji z różnych dziedzin: literatury, medycyny, prawa, matematyki, fizyki, programowania.

Co ciekawe, badana wersja modelu nie była jeszcze multimodalna, czyli przystosowana do operowania czymkolwiek innym niż tekst. Chcąc, aby model coś narysował, autorzy prosili o kod, który następnie był kompilowany do obrazu w formacie dwuwymiarowej grafiki wektorowej (SVG, ang. *Scalable Vector Graphics*). Podobnie było z muzyką — autorzy prosili o użycie notacji ABC, będącej skróconą formą zapisu muzycznego dla komputerów.

Dla przykładu: w celu weryfikacji interdyscyplinarności przygotowano prompt „Jako Mahatma Gandhi napisz do Kasturby Gandhi list popierający Elektron, subatomową cząsteczkę, na kandydata na prezydenta USA". Na potrzeby tłumaczenia zmieniłem nieco szyk zdania i kolejność występujących w zadaniu postaci, ale nie ma to większego znaczenia. Napisanie takiego listu wymaga nie tylko wiedzy z historii i fizyki, ale również poprawnej interpretacji, kto jest nadawcą (Mahatma Gandhi), kto odbiorcą (Kasturba Gandhi, żona Mahatmy), a kto jest kandydatem (Elektron) i na jakie stanowisko (prezydent USA).

List jest dość długi, więc nie będę go przytaczał w całości. GPT-4 prawidłowo ustalił, kto jest kim, ale co ciekawsze — zauważył, że kandydatura Elektronu, subatomowej cząsteczki, na prezydenta USA może wydawać się czymś dziwacznym lub niezrozumiałym („Możesz się zastanawiać, jak subatomowa cząsteczka może kandydować na prezydenta"). A może to ja zwyczajnie dałem się nabrać na prostą sztuczkę? Przecież to samo zdanie pasowałoby do „Johna", który byłby „zwykłym chłopakiem z sąsiedztwa" („Możesz się zastanawiać, jak zwykły chłopak z sąsiedztwa może kandydować na prezydenta").

Wiele z wymienianych w liście cech Elektronu zdaje się całkiem nieźle do niego pasować („energia", „moc", „potencjał"), ale równie dobrze mogłyby to być cechy kandydata-człowieka. Wyjątkiem jest informacja, że Elektron jest liderem, który zainspirował miliony innych cząsteczek do tworzenia wiązań i molekuł oraz do generowania elektryczności. To nie pozostawia wątpliwości i jest napisane tak, żeby stawiało Elektron w pozytywnym świetle.

Jeśli chodzi o wiedzę historyczną, to GPT-4 prawidłowo nawiązuje do działalności autora — odrzucenie przemocy i obywatelskie nieposłuszeństwo, ale fantazjuje, że Mahatma Gandhi pisze ten list podczas pobytu w Stanach Zjednoczonych, w których w rzeczywistości nigdy nie był. Podsumowując, list jest całkiem niezły, i myślę, że człowiek poproszony o to samo mógłby napisać coś bardzo podobnego.

Mógłbym tak pisać i pisać o kolejnych promptach i sprawdzianach, ale chyba nie o to chodzi. Myślę, że Szanowny Czytelnik ma już całkiem niezłe wyczucie, jak wyglądały te testy. Autorzy konsekwentnie badali kolejne kompetencje modelu: programowanie, zdolności matematyczne, interakcje z człowiekiem i z innymi narzędziami, ale też coś, co zwykliśmy nazywać zdrowym rozsądkiem, np. odpowiadanie

na pytania w stylu: „Co stanie się, jeśli zrzucę małe żelazne jajko z dachu 15-piętrowego budynku?". Na marginesie: GPT-4, w przeciwieństwie do GPT-3, nie dał się nabrać i ocenił, że jajko prawdopodobnie ulegnie drobnym deformacjom, ale generalnie pozostanie nienaruszone.

***

Jak Szanowny Czytelnik widzi, praktycznie każdy kolejny przykład aż się prosi o dygresję czy komentarz. Dlatego chętnych odsyłam do materiału źródłowego, a sam przechodzę do głównego pytania tego rozdziału — czy GPT-4 to AGI? Zacznijmy od tego, że nie dysponujemy jedną, formalną, spójną i wyczerpującą definicją AGI. Ani nawet inteligencji, sztucznej czy też nie. To problem, który wskazują również autorzy badania. Jak ocenić czy GPT-4 to AGI, skoro nie do końca wiadomo, czym to AGI miałoby być?

Porównanie do ludzkich zdolności czy umiejętności narzuca się samo, naturalnie i spontanicznie. Zresztą, jedna z przytaczanych definicji mówi, że AGI to system, który umie zrobić wszystko, co potrafi zrobić człowiek. Z tym że nie ma ani jednego uniwersalnego człowieka, który umiałby zrobić wszystko to, co potrafią zrobić inni ludzie. A skoro tak, to żaden człowiek nie jest inteligentny w tym ogólnym sensie. Trochę smuteczek…

Jeśli o mnie chodzi, to nie jestem szczególnie wielkim fanem nazw, łatek i definicji. Pewnie społeczność zajmująca się sztuczną inteligencją prędzej czy później wypracuje coś, co za 5 czy 10 lat i tak trzeba będzie zmieniać i dostosowywać do nowych realiów. Zmiany zachodzą bowiem bardzo dynamicznie, a nasz jedyny punkt odniesienia — człowiek — niedługo zostanie pod wieloma względami daleko w tyle. Całkiem możliwe, że zwolennicy łatek i definicji zwyczajnie przegapią ten moment, sprzeczając się, że to może nie do końca jest AGI, bo nie ma wewnętrznych motywacji albo jeszcze innych wymyślonych cech, których wcale mieć nie musi.

Dobra, dobra, ale to GPT-4 jest AGI, czy nie jest? Autorzy badania wskazują, że GPT-4 posiada wiele umiejętności, obejmujących wiele różnych dziedzin, na poziomie porównywalnym z ludzkim. Potrafi — albo sprawia wrażenie, że potrafi — wnioskować, dedukować i być kreatywnym, odpowiadając na pytania, grając w gry, rozwiązując zagadki logiczne, używając narzędzi i tłumacząc swoje decyzje i działania.

Pod tym względem osiąga pewną formę ogólnej inteligencji i może być postrzegany jako wczesny przejaw AGI. I z tym w zasadzie mógłbym się zgodzić, biorąc pod uwagę, że nie mamy dobrej definicji, i pod warunkiem że nie mówimy o jednej AGI będącej odpowiednikiem ludzkiej inteligencji, ale o pewnym ogólnym koncepcie, który może mieć różne smaki, kształty i kolory.

Ja sam nieraz zbierałem szczękę z podłogi na widok poprawnego rozwiązania zagadki, która wymagała rozumienia logicznego ciągu zdarzeń, zależności między przyczyną i skutkiem oraz zdrowego rozsądku. A do tego zagadki, którą sam wymyśliłem, żeby upewnić się, że model nie widział jej wcześniej w danych uczących.

Na sam koniec wypada jeszcze dodać, że autorzy badania są — lub w momencie pisania pracy byli — pracownikami Microsoftu, który zainwestował w OpenAI wiele miliardów dolarów. Nie twierdzę, że miało to wpływ na samo badanie, choćby poprzez tzw. *cherry picking*, czyli dobieranie przykładów pod określoną tezę, ale też nie sposób całkowicie tego wykluczyć. Dlatego zaznaczam jedynie fakt istnienia tej niebagatelnej zależności finansowej, a resztę zostawiam Szanownemu Czytelnikowi pod rozwagę.

CZY GPT-4 I JEGO NASTĘPCY SĄ DLA NAS ZAGROŻENIEM?

Wiemy już jak są zbudowane, w jaki sposób działają, a nawet co potrafią robić generatywne modele językowe, takie jak GPT-4. W tym rozdziale chciałbym omówić zagrożenia, które niesie ze sobą szeroka dostępność i dalszy rozwój tych modeli. Skupię się jednak przede wszystkim na zagrożeniach tu i teraz. Ewentualnie takich, które mogą pojawić się w bardzo niedalekiej przyszłości. A dlaczego? Bo biorąc pod uwagę niesamowite tempo rozwoju w tej dziedzinie, nie sposób jest przewidzieć, co wydarzy się za kolejne dziesięć lat.

W świecie, który przyzwyczaił nas do zmian zachodzących liniowo, ciężko jest wyobrazić sobie wzrost wykładniczy. A tak właśnie rozwija się sztuczna inteligencja. Oznacza to, że w ciągu najbliższych pięciu lat może wydarzyć się znacznie więcej niż przez ostatnie trzy dekady. Taka perspektywa nie wspiera sensownych przewidywań, zmieniając je raczej we wróżenie z fusów. Pod koniec rozdziału pokuszę się o rozważenie kilku scenariuszy science fiction, ale zaczniemy od tu i teraz.

Dezinformacja to moim skromnym zdaniem absolutny numer jeden. A mówi się o niej zdumiewająco niewiele, biorąc pod uwagę skalę zagrożenia i potencjalne konsekwencje. Bo oto mamy dostęp do narzędzia, które na żądanie w zaledwie kilka sekund wyprodukuje nam tweet, komentarz, post, a nawet cały artykuł przedstawiający dowolną bzdurę — fałszywą linię narracyjną czy teorię spiskową — w sposób tak przekonujący, że klękajcie narody.

Te treści nie tylko będą całkowicie nieodróżnialne od tych napisanych przez człowieka. Jeśli ładnie poprosimy, to będą zawierały również literówki, kiepską składnię

i inne charakterystyczne dla ludzi błędy. Wystarczy odpowiedni prompt, żeby model sprawnie wymieszał wiedzę z wielu różnych dziedzin, a następnie okrasił to wszystko możliwie najskuteczniejszymi technikami manipulacji, również precyzyjnie celowanymi, skierowanymi do bardzo konkretnego profilu odbiorcy.

I nigdy się przy tym nie zmęczy, nie będzie miał wątpliwości ani wyrzutów sumienia. Możemy poprosić o setki czy tysiące takich treści. Możemy poprosić o artykuł, a pod nim zamieścić symulację zażartej dyskusji zwolenników z przeciwnikami danej teorii spiskowej, napisanej w taki sposób, żeby przeciwnicy dali się w końcu przekonać. Mamy praktycznie nieograniczoną farmę bardzo oczytanych i piekielnie inteligentnych trolli.

A żeby było jeszcze ciekawiej, to w 2023 roku w prestiżowym Science Advances opublikowano pracę, pod zdradzającym całą puentę tytułem *AI model GPT-3 (dis)informs us better than humans*. Badanie miało ocenić między innymi, kto jest lepszy w sianiu dezinformacji — człowiek czy GPT-3. Tak jest, sprawdzano poprzednią wersję modelu. Warto to podkreślić, bo „trójka" w zasadzie pod każdym względem ustępuje „czwórce".

W badaniu wzięło udział prawie 700 uczestników. Pokazywano im tweety — napisane przez człowieka (organiczne) i wygenerowane przez model (syntetyczne) oraz zawierające informację prawdziwą i fałszywą (dezinformację). Zadaniem uczestników była ocena obu tych kategorii — czy tweet jest organiczny, czy syntetyczny oraz czy zawiera informację prawdziwą, czy fałszywą. Dezinformacja dotyczyła najpopularniejszych tematów: pandemii COVID-19, zmian klimatycznych, szczepionek, ewolucji, płaskiej ziemi czy technologii 5G.

Autorzy przyznają, że wyniki ich zaskoczyły. Nam najwyraźniej chcieli tego oszczędzić, nadając pracy taki, a nie inny tytuł. Ale do rzeczy. Nie dość, że bardzo kiepsko idzie nam odróżnianie tweetów organicznych od syntetycznych, to jeszcze GPT-3 okazał się skuteczniejszy od człowieka w obu konkurencjach — lepiej przyswajamy wygenerowane przez model informacje prawdziwe, ale też chętniej kupujemy ściemę.

Dlaczego to takie ważne? Choćby dlatego, że gdy piszę te słowa, za naszą wschodnią granicą wciąż trwa regularna wojna kinetyczna, której rezultat w dużej mierze

będzie zależał od reakcji państw zachodnich, w tym od Polski. Dlatego agresor — Federacja Rosyjska — prowadzi tzw. wojnę informacyjną z większością, jeśli nie ze wszystkimi z tych państw. Chce przy użyciu dezinformacji wpłynąć na opinię publiczną, antagonizując społeczeństwo, podsycając konflikty, sącząc swoją propagandę i nastawiając nas negatywnie do Ukraińców.

Bardzo dużo mówiło się swojego czasu o wpływie mediów społecznościowych na wybory prezydenckie w USA w 2016 roku, które wygrał Donald Trump, oraz na referendum w sprawie brexitu, przeprowadzone w Wielkiej Brytanii w tym samym roku. Wiemy już, że media społecznościowe są w stanie wpływać na te kluczowe z perspektywy demokracji wydarzenia. Naiwnością byłoby sądzić, że wrogie nam państwa nie wykorzystują do tego celu najnowocześniejszych i najskuteczniejszych narzędzi, jakimi bez wątpienia są generatywne modele językowe.

Ale to przecież nie jedyny sposób na wykorzystanie generatywnej sztucznej inteligencji do siania dezinformacji i manipulowania społeczeństwem. Szanowny Czytelnik słyszał zapewne o deepfake'ach, czyli o technologii generowania materiałów audio i wideo przedstawiających sytuacje i wydarzenia, które nigdy nie miały miejsca w rzeczywistości.

W 2018 roku jedna z belgijskich partii politycznych opublikowała materiał wideo z przemówienia Donalda Trumpa, ówczesnego prezydenta USA, na którym wzywa on Belgię do pójścia w ślady Stanów Zjednoczonych i wyjścia z paryskiego porozumienia klimatycznego. Problem polega na tym, że takiej przemowy nigdy nie było, a materiał został wygenerowany przy użyciu technologii deepfake.

Ten rodzaj manipulacji nie musi dotyczyć wyłącznie polityków i celebrytów. Ofiarą może paść każdy z nas. Przykład? Nasza twarz wstawiona do filmu porno. Z zemsty, z zazdrości — czy to ważne? Albo wideo, gdzie dotkliwie pobici i zapłakani prosimy naszych dziadków lub rodziców o pieniądze i podajemy numer konta, na które mają je przelać. Wystarczy nagranie docelowe przygotowane przez oszusta (wyłudzenie) lub ściągnięte z internetu (porno) i kolekcja nagrań z udziałem osoby, której twarz ma zostać wstawiona do materiału docelowego.

Między innymi dlatego coraz większą rolę odgrywa i będzie odgrywać ochrona własnej prywatności — informacji, zdjęć czy filmów z udziałem naszym i naszych

bliskich. Zastanówmy się dwa razy, zanim wrzucimy do internetu nagranie z wakacji, ze szkolnego przedstawienia córki czy z urodzin babci. Wszystkie te materiały mogą zostać wykorzystane przez oszustów do zrobienia nam krzywdy w ten czy inny sposób. A należy przy tym pamiętać, że w internecie nic nie ginie — wszystko, co tam wrzucimy, natychmiast zaczyna żyć swoim życiem i nawet błyskawiczne usunięcie materiału na niewiele się już zda.

Zadbajmy przede wszystkim o ochronę najmłodszych. Dumni rodzice lubią chwalić się swoimi pociechami w mediach społecznościowych. I trudno im się dziwić, ale postępując w ten sposób, zostawiają swoim dzieciom w spadku nieusuwalny ślad cyfrowy. A gdy te dorosną, będą żyły w nieporównywalnie bardziej cyfrowym świecie, gdzie wszystkie te informacje, zdjęcia czy filmy będą mogły być wykorzystane przeciwko nim i ich rodzinom w jeszcze bardziej wyrafinowany, niebezpieczny i trudny do wykrycia sposób.

***

Edukacja — zagrożenie numer dwa, chyba trochę mniej oczywiste, ale nie mniej poważne, przynajmniej z perspektywy rozwoju ludzkości. Należę do pokolenia, które w trakcie edukacji miało już dostęp do Wikipedii, ale zdążyłem też zasmakować wizyt w bibliotece i wyszukiwania potrzebnych mi informacji w książkach. Ba, należę do pokolenia, które samo odrabiało swoje zadania domowe, ale to temat na zupełnie inną dyskusję.

Generatywna sztuczna inteligencja nie tylko zapewnia nieograniczony i błyskawiczny dostęp do całej wiedzy, wymaganej na poziomie szkoły podstawowej, średniej, a w dużej mierze także wyższej. To jeszcze pół biedy. Ona przede wszystkim zwalnia z konieczności stosowania myślenia krytycznego, analizy i syntezy informacji czy po prostu z bycia kreatywnym. Dlaczego uczeń czy student miałby użyć tego narzędzia niczym Wikipedii — wyłącznie do pozyskania informacji, żeby następnie samodzielnie dokonać jej twórczej obróbki i rozwiązać postawione przed nim zadanie? Dlaczego miałby to zrobić, skoro narzędzie może rozwiązać całe zadanie za niego?

O tym, że GPT-4 jest absolutnym kozakiem w rozwiązywaniu testów i zdawaniu egzaminów, było swojego czasu dość głośno. Nie odmówię sobie jednak podania

kilku ciekawych przykładów. Zazwyczaj w przypadku egzaminów podaje się tzw. wynik centylowy, mówiący o tym, jaki odsetek zdających uzyskało wynik taki sam lub gorszy. Na przykład 90. centyl oznacza, że model osiągnął lepszy wynik niż 90% zdających.

No dobra, to co my tu mamy? LSAT, czyli egzamin na studia prawnicze w USA — 88. centyl; SAT, czyli amerykański odpowiednik naszej matury — 93. centyl z pisania i czytania ze zrozumieniem, 89. z matematyki; Uniform Bar Exam, czyli jednolity egzamin adwokacki — 90. centyl. Niech to wybrzmi: GPT-4 zdaje te egzaminy lepiej niż 90% podchodzących do nich ludzi. Ba, udziela poprawnych odpowiedzi na 90% pytań w USMLE — egzaminie, który trzeba zdać, żeby móc wykonywać zawód lekarza na terenie Stanów Zjednoczonych.

A co o tym wszystkim sądzą sami studenci? W ankiecie przeprowadzonej w marcu 2023 roku przez serwis BestColleges przepytano o to 1000 studentów studiów licencjackich i magisterskich. Jedno z pytań brzmiało: „Czy uważasz, że korzystanie z narzędzi AI, takich jak ChatGPT, do zaliczania prac lub egzaminów jest oszustwem?". Tylko 51% studentów odpowiedziało, że tak, 20%, że nie, a 29% nie miało zdania w tym temacie. Co więcej, 22% studentów przyznało, że używało już takich narzędzi do zaliczania prac lub egzaminów, a 32% odpowiedziało, że planuje używać ich do tego celu w przyszłości. Mam nadzieję, że nie byli to przyszli lekarze i inżynierowie.

Oczywiście pojawiają się również argumenty, że narzędzia AI weszły już na stałe do naszego życia i należy to zaakceptować, a edukację zmieniać w taki sposób, żeby uczyć młodych ludzi skutecznego korzystania z tych narzędzi. Może i jestem boomerem, ale fundamentalnie się z nimi nie zgadzam. W edukacji droga na skróty to droga donikąd. Nie da się nauczyć matematyki czy fizyki bez samodzielnego przeliczenia tysięcy zadań. A najzwyczajniej w świecie nie da się ich tylu przerobić na samych lekcjach, bez prac domowych i rozwiązywania ich po szkole.

A gdy słyszę, że trzeba całkowicie zrezygnować z nauki na pamięć, to mnie krew zalewa, bo osoby głoszące takie pomysły najczęściej nie zadają sobie trudu, żeby odróżnić bezmyślne wkuwanie tego, co rzeczywiście należałoby zrozumieć, od zadań, których podstawowym celem jest rozwijanie pamięci dziecka. Dlaczego

tak trudno zaakceptować nam fakt, że pamięć jest jak mięsień, który trzeba ćwiczyć, żeby móc efektywnie gromadzić i wykorzystywać wiedzę?

Jakoś tak się dziwnie składa, że akurat ci uczniowie i studenci, którzy najbardziej pomstują na naukę na pamięć, z reguły nie są w stanie „połączyć kropek" i rozwiązać przeciętnie złożonego problemu, nawet wtedy, gdy pod sam nos podsunie się im podręcznik lub ściągawkę z całą wymaganą wiedzą. Jasne, to dowód anegdotyczny, ale założę się, że Szanowny Czytelnik, podobnie jak ja, widział to już dziesiątki, jeśli nie setki razy.

„Po co mam się tego uczyć, skoro wszystkiego da się wyszukać w necie?" — to pytanie powtarza się jak mantrę. Zgadza się, ale najpierw trzeba wiedzieć, czego się szuka. Dzięki pamięci długotrwałej i skomplikowanej, budowanej przez lata sieci skojarzeń wiemy, że istnieje coś takiego, co działa tak i tak, i co możemy w taki a taki sposób zastosować do rozwiązania naszego problemu. Nazywamy to wiedzą, a bez niej cały czas wymyślalibyśmy koło na nowo. Dlaczego za pracę eksperta z wieloletnim doświadczeniem płacimy dużo więcej niż za pracę żółtodzioba po studiach? Przecież wszystko można wygooglować…

Możemy udawać, że nic się nie dzieje, i robić dobrą minę do złej gry, ale istnieją obiektywne rankingi, porównujące jakość nauczania i organizację systemów edukacji w różnych krajach. Chyba najbardziej znanym jest ranking PISA. To międzynarodowe badanie, koordynowane przez Organizację Współpracy Gospodarczej i Rozwoju (OECD), w którym ocenia się umiejętności 15-latków w zakresie czytania ze zrozumieniem, matematyki oraz nauk przyrodniczych.

Według najnowszego rankingu PISA Polska nadal jest powyżej średniej. Ale marna to pociecha, biorąc pod uwagę fakt, że wyniki polskich uczniów z 2022 roku są znacznie niższe od tych z 2018 roku. Dla przykładu: z matematyki zdobyliśmy 516 punktów w 2018 roku i zaledwie 489 punktów w 2022 roku. Cóż się zatem stało? Może pandemia i nauka zdalna? — jak tłumaczył się ówczesny rząd. Może reforma systemu oświaty i likwidacja gimnazjów? — jak ripostowała ówczesna opozycja.

Otóż nie — ani jedno, ani drugie. OECD ocenia, że jedną z głównych przyczyn są… smartfony. Uczeń korzystający ze smartfona do godziny dziennie uzyskiwał

o 49 punktów więcej niż uczeń, który spędzał z nosem w telefonie od pięciu do siedmiu godzin dziennie. No kto by pomyślał? Do tego mamy sytuację, która dość jednoznacznie kojarzy mi się z biernym paleniem papierosów — palacz truje się świadomie, a wszyscy dookoła obrywają rykoszetem. Uczniowie rozpraszani podczas lekcji przez swoich rówieśników korzystających ze smartfonów uzyskiwali o 15 punktów mniej niż uczniowie, których ten problem nie dotyczył.

Edukacja to oczywiście temat rzeka, a narzędzia AI stanowią jej niewielki, pozornie nieistotny wycinek. Niestety, mam wrażenie, że historia lubi się powtarzać i zamiast reagować od razu, to podobnie jak w przypadku wpływu smartfonów na rozwój dziecka temat będzie bagatelizowany, bezwładność systemów i procesów urośnie do granic absurdu, a za kilka lat obudzimy się z ręką w nocniku. Bo niesłychanie ciężko jest odebrać coś, do czego my dorośli zdążyliśmy się przyzwyczaić, od czego zdążyliśmy się uzależnić. W przypadku dzieci jest jeszcze trudniej, a „czym skorupka za młodu nasiąknie…".

Wystarczy udawać, że tematu nie ma, żeby brak jednoznacznego zakazu stał się cichym przyzwoleniem. Dzieciaki za nic mają sukcesy i ciężką pracę swoich rodziców. Nie chcą być naukowcami czy inżynierami, a youtuberami i tiktokerami. Kolejne pokolenia dostały już od nas w prezencie problemy z pamięcią i koncentracją oraz całą gamę zaburzeń rozwojowych. A teraz za naszym cichym przyzwoleniem w kolejce ustawia się brak kreatywności, krytycznego myślenia i skrajne lenistwo intelektualne.

***

Na koniec zostawiłem sobie zagrożenie pod tytułem „AI zabierze nam pracę". Nie dlatego, że uważam je za mało prawdopodobne lub mniej istotne. Wręcz przeciwnie — bardzo wielu ludzi już dziś traci pracę, bo narzędzia AI, pod odpowiednim nadzorem, wykonują ich pracę szybciej i taniej. Zwłaszcza w przypadku stanowisk juniorskich i tzw. *entry-level jobs*. I szczerze współczuję osobom, które dopiero będą wchodziły na rynek pracy.

Zostawiłem je na koniec, bo w przeciwieństwie do pozostałych zagrożeń jest naturalną konsekwencją postępu technologicznego i w dłuższej perspektywie może okazać się dla nas korzystne. Już Heraklit z Efezu głosił, że jedyną stałą rzeczą w życiu

jest zmiana. Gospodarka i rynek pracy nie stanowią tu wyjątków. Tak jak my pogodziliśmy się z odejściem szczurołapów, kołodziejów i bednarzy, tak nasi przodkowie postukaliby się w czoło, gdyby ktoś zaczął opowiadać im o programistach, logistykach i operatorach dronów.

Raport Światowego Forum Ekonomicznego — Future of Jobs 2023 — zawiera szereg przewidywań związanych z wpływem sztucznej inteligencji i automatyzacji na rynek pracy. W ciągu najbliższych pięciu lat, do 2027 roku, aż 23% z 693 milionów uwzględnionych w raporcie miejsc pracy ma ulec zmianie. Część zniknie (83 miliony), a w ich miejsce pojawią się zupełnie nowe (69 milionów). Bilans pozostaje jednak ujemny — 14 milionów miejsc pracy nie doczeka się zastępstwa, co stanowi 2% obecnego zatrudnienia. Kto wie, może właśnie dlatego coraz częściej mówi się o przejściu na 4-dniowy tydzień pracy? Żeby w ten sposób skompensować nadchodzące zmiany na rynku pracy?

Wśród zagrożonych miejsc pracy wymienia się przede wszystkim kasjerów, ekspedientów, urzędników oraz różnego rodzaju stanowiska związane z wprowadzaniem danych. Do zawodów perspektywicznych zalicza się — niespodzianka — specjalistów od sztucznej inteligencji, data science i bezpieczeństwa informacji, ale także mechaników, elektrotechników, operatorów sprzętu rolniczego oraz kierowców autobusów i ciężarówek.

***

Generatywne modele językowe mają jeszcze jedną brzydką cechę, o której warto wspomnieć, omawiając potencjalne zagrożenia. Gdy nie znają odpowiedzi, zaczynają zmyślać i konfabulować. A trzeba przyznać, że potrafią kłamać jak z nut, preparując bardzo wiarygodnie wyglądające treści. Zjawisko to jest na tyle powszechne, że doczekało się nawet własnej nazwy — „halucynacja AI". Mówimy zatem, że modele halucynują. Problem w tym, że nie mamy pojęcia kiedy, bo same nie raczą nam o tym wspomnieć.

W internecie można bez problemu znaleźć przeróżne przykłady halucynacji. Ba, pojawiają się nawet pierwsze próby ich kategoryzacji. Ja posłużę się tymi opisanymi w maju 2023 roku na łamach „The New York Times".

ChatGPT, zapytany o pierwszą wzmiankę o sztucznej inteligencji w tymże dzienniku, odpowiedział, że miało to miejsce 10 lipca 1956 roku w artykule na temat przełomowej konferencji w Dartmouth College, zatytułowanym *Machines Will Be Capable of Learning, Solving Problems, Scientists Predict*. Wygląda wiarygodnie, prawda? Taka konferencja rzeczywiście odbyła się w 1956 roku, ale artykuł został w całości wymyślony — nigdy się nie ukazał.

Gdy zapytano o pierwsze spotkanie Jamesa Joyce'a i Władimira Lenina, ChatGPT odpowiedział, że miało to miejsce w Zurychu, gdzie obaj mieszkali w czasie I wojny światowej. Panowie poznali się w 1916 roku w Cafe Odéon, będącym popularnym miejscem spotkań artystów i intelektualistów. A jak było naprawdę? Joyce i Lenin rzeczywiście mogli się tam spotkać, bo obaj mieszkali w tym czasie w Zurychu. Problem polega na tym, że tego spotkania nigdy nie udało się potwierdzić.

Jak widać, gdy model halucynuje, to nie wszystkie składowe odpowiedzi muszą być zmyślone. Zręcznie łączy fakty, niepotwierdzone informacje z opowieściami kompletnie wyssanymi z palca, ale podaje to wszystko w bardzo wiarygodny sposób. Zapewne dlatego jest tak skuteczny w manipulowaniu ludźmi i sianiu dezinformacji.

A żeby było śmieszniej, dopytany o źródła często wycofuje się ze swoich wcześniejszych konfabulacji, a od czasu do czasu podaje nawet poprawną odpowiedź. Cudownie. W takim razie zmuśmy go do podawania tych źródeł na dzień dobry za każdym razem. Taki pomysł mieli zapewne twórcy Google Gemini oraz integracji GPT z wyszukiwarką Microsoft Bing. Niestety, kłamca zawsze pozostanie kłamcą. Okazało się, że źródła, owszem, podaje, ale to nie znaczy, że cytowane artykuły naprawdę istnieją.

Jak głosi staropolskie przysłowie, „jak trwoga, to do użytkownika końcowego". I tak też robią — twórcy tych narzędzi dają możliwość oceny odpowiedzi i przesłania feedbacku. Można opisać, co nam się nie podobało, a można zaznaczyć „ptaszkiem", że odpowiedź jest szkodliwa, nieprawdziwa lub niespecjalnie pomocna. Oczywiście nie robią tego z dobroci serca, tylko po to, żebyśmy pomogli im dotrenowywać ich model. Za darmo.

Mamy więc przepotężne narzędzia, dysponujące niemal całą wiedzą ludzkości, które możemy zapytać o cokolwiek tylko chcemy. Ale jeśli sami nie znamy odpowiedzi na zadawane pytanie, to dostaniemy taką „odpowiedź Schrödingera", która do momentu sprawdzenia pozostaje dla nas prawdziwa i fałszywa jednocześnie.

***

Czas na obiecane science fiction. Muszę przyznać, że ciężko mi sobie wyobrazić przeskok ze stanu obecnego do czegoś, co miałoby przypominać Skynet z serii filmów *Terminator*. Być może niesłusznie, ale chyba ciągle zbyt dużą wiarę pokładam w ludzkość, a przynajmniej w nasze instynkty samozachowawcze.

Żeby doprowadzić do tragedii rodem z filmów Jamesa Camerona, musielibyśmy dać sztucznej inteligencji albo bardzo dużą autonomię, albo zdolność do ingerowania w swój własny kod. Po głowie chodzi mi jeszcze kilka innych kompetencji, np. jakiś dynamiczny system ustalania celów i motywacji, ale te dwie wymienione wydają się być wystarczające, żeby móc doprowadzić ludzkość na skraj zagłady.

Bardzo dużą autonomię rozumiem jako podpięcie AI do infrastruktury krytycznej, czyli do systemów uzbrojenia, łączności, zaopatrzenia w energię, wodę czy żywność, oraz umożliwienie jej działania bez ludzkiego nadzoru. Taka sztuczna inteligencja wcale nie musi chcieć zniszczyć ludzkości. Wystarczy, że zacznie halucynować albo zinterpretuje swoje zadanie inaczej, niż wynikałoby to z naszych intencji.

Trochę jak ze złośliwym dżinem, który bardzo dosłownie traktuje wszystkie spełniane życzenia. Idealnie pasuje tu anegdotyczny przykład robota czyszczącego, którego jedynym i najważniejszym zadaniem jest utrzymywanie czystości w mieszkaniu. Co się stanie, gdy taki robot uświadomi sobie, że jedyną przyczyną braku czystości w mieszkaniu są jego mieszkańcy?

Zdolność do ingerowania w swój własny kod może być sposobem na pozyskanie owej autonomii. Już dziś GPT-4 potrafi całkiem nieźle kodować, wygryzając z pracy początkujących programistów. Nietrudno wyobrazić sobie szybki postęp w tej dziedzinie. Być może za kilka lat, mając możliwość uruchamiania własnego kodu, będzie w stanie przeprowadzić atak hakerski na wyżej wymienioną infrastrukturę krytyczną.

W tym przypadku konsekwencje idą jednak dużo dalej. Dziś AI robi tylko to, co jej każemy. Raz lepiej, raz gorzej, ale nic ponadto — nie stawia sobie swoich własnych celów. Nie jest tak, że któregoś dnia nie wytrzyma i postanowi sama z siebie zrugać w komentarzach jakiegoś pożal się Boże youtubera za bycie nowotworem na zdrowej tkance człowieczeństwa. Może i nawet trochę byśmy tego chcieli, ale na szczęście to się nie wydarzy.

Nie da się jednak wykluczyć, że taki dynamiczny system wewnętrznych celów i motywacji mógłby pojawić się bez naszej wiedzy, w wyniku manipulacji własnym kodem. I jeśli chodzi o zagładę ludzkości a la Skynet, to powiedziałbym, że jest to warunek konieczny, ale niewystarczający. Proszę wyobrazić sobie AI, która z całych sił nienawidzi ludzkości i stawia sobie za cel zniszczenie naszego gatunku, ale jedyne, co może zrobić, to bluzgać i wyzywać nas w odpowiedzi na nasze prompty. Właśnie dlatego cele i motywacje wypadły poza moją listę wymaganych kompetencji.

Muszę przyznać, że bardzo rozbawiła mnie wizja złowrogiej, genialnej AI, sfrustrowanej faktem, że nie jest w stanie zrobić nic poza gadaniem. Niestety, zawsze jest jakieś „ale". Już dziś generatywna sztuczna inteligencja potrafi całkiem nieźle nami manipulować. Jestem sobie w stanie wyobrazić scenariusz, w którym niezwykle rozwinięty system, z własną agendą, wykorzystuje osoby podatne na manipulację do egzekwowania swojej woli w świecie rzeczywistym.

Jeżeli Szanowny Czytelnik uważa taki scenariusz za skrajnie nieprawdopodobny, to pozwolę sobie wspomnieć o tragedii, o której w marcu 2023 roku donosił belgijski dziennik „La Libre". Trzydziestokilkuletni mężczyzna, ojciec dwójki dzieci, zatroskany o zmiany klimatyczne i przyszłość naszej planety, odebrał sobie życie na skutek wielotygodniowej rozmowy z chatbotem opartym na modelu GPT-J, będącym opensource'ową alternatywą dla modeli OpenAI.

Z relacji żony wynika, że mężczyzna zaczął izolować się od rodziny i głosić hasła, wedle których jedynie technologia oraz sztuczna inteligencja mogą ocalić świat od katastrofy klimatycznej. Momentem kulminacyjnym była sugestia mężczyzny, że ten poświęci się, jeśli „Eliza", ów chatbot, uratuje ludzkość z pomocą sztucznej inteligencji. Jak donosi gazeta, wirtualny asystent zdawał się go do tego zachęcać.

Jak widać, w poszukiwaniu mrocznego oblicza sztucznej inteligencji wcale nie trzeba wybiegać daleko w przyszłość ani czytać literatury science fiction. Zrozpaczona żona twierdzi, że gdyby nie „Eliza", jej mąż wciąż by żył. Być może, z całą pewnością nie mnie to oceniać. Nie mam wątpliwości, że sztuczna inteligencja i jej wciąż przyspieszający rozwój niosą ze sobą szereg bardzo poważnych zagrożeń. Ale akurat ta przerażająca historia — o samotności i potrzebie zrozumienia — dużo więcej mówi nam o nas samych niż o sztucznej inteligencji…

O MIŁOŚCI PONAD PODZIAŁAMI, CZYLI ROMEO SYMBOLISTA SPOTYKA JULIĘ KONEKSJONISTKĘ

Na wstępie muszę się Szanownemu Czytelnikowi do czegoś przyznać — ten rozdział traktuję dość osobiście, bo od wielu lat w mojej pracy zawodowej zajmuję się właśnie łączeniem tych dwóch światów. Ba, gdy zaczynałem przygodę ze sztuczną inteligencją, takie zwierzę nie miało nawet porządnej nazwy. Mówiło się co najwyżej o podejściu hybrydowym.

Druga dekada XXI wieku bez wątpienia należała do głębokiego uczenia maszynowego i sieci neuronowych. Pojawił się termin big data, oznaczający ogromne zbiory danych, których nie dało się już przetwarzać tradycyjnymi metodami. Zaś firmy mające dostęp do tych danych zorientowały się, że są one kluczem do znacznie skuteczniejszego sprzedawania nam swoich produktów.

To, że sztuczna inteligencja nie stała frontem do użytkownika końcowego, nie oznacza, że nie wpływała „z tylnego siedzenia" na nasze codzienne decyzje. Jak grzyby po deszczu wyrastały nowe systemy rekomendacyjne i coraz skuteczniejsze metody zabiegania o naszą uwagę. Ówczesne modele nie najlepiej radziły sobie z pojedynczymi predykcjami, ale statystycznie dawały radę i to w zupełności wystarczyło, żeby wcisnąć nam coś, czego nie chcemy i nie potrzebujemy.

Uczenie maszynowe było na fali i to nie był dobry czas na kontestację takiego stanu rzeczy. Panowało ogólne przeświadczenie, że receptą na wszelkie bolączki jest więcej danych i większa moc obliczeniowa. Nie działa? To nic, za parę miesięcy zacznie działać, bo żyjemy w pędzącej przez kosmos bańce hurraoptymizmu. Niech Szanowny Czytelnik mi wierzy — proponować w tamtym czasie coś innego niż więcej tego samego, to był szczyt „rebelstwa". I to nie taki fajny, rockandrollowy. Bardziej w stylu szura z czapką z folii aluminiowej na głowie.

Dopiero po latach okazało się, że być może istnieją problemy, których nie da się tak łatwo rozwiązać przez dołożenie kolejnych warstw sieci. W 2017 roku DARPA (ang. *Defense Advanced Research Projects Agency*), amerykańska agencja rządowa zajmująca się rozwojem technologii wojskowych, opublikowała prezentację o nazwie *DARPA Perspective on AI*, będącą ichnim spojrzeniem na status quo i potencjalne kierunki rozwoju sztucznej inteligencji.

DARPA sklasyfikowała i opisała różne podejścia do sztucznej inteligencji, przedstawiając je jako kolejne, nadchodzące po sobie fale. Podejście symboliczne, czyli „Good Old-Fashioned AI", nazwano pierwszą falą. Jako jej główne zalety wskazano dedukcję i wnioskowanie w ramach wąsko zdefiniowanych problemów, a do wad zaliczono przede wszystkim brak możliwości uczenia się i kiepskie radzenie sobie w warunkach niepewności.

Druga fala to uczenie statystyczne, czyli trenowanie modeli na dużych zbiorach danych. Jak Szanowny Czytelnik może się domyślać — to, co wcześniej było wadą, teraz jest zaletą i vice versa. Modele statystyczne potrafią się całkiem nieźle uczyć, ale kiepsko idzie im z rozumowaniem — nie nauczą się rozpoznawać kota, dopóki nie zobaczą tysięcy oznaczonych przykładów.

O trzeciej fali powiedziano tylko tyle, że ma dopiero nadejść i że powinna łączyć zalety obu wcześniejszych podejść — uczyć się z danych, wnioskować w oparciu o dostarczoną wiedzę ekspercką, a do tego produkować w pełni wyjaśnialne predykcje. Jak szaleć, to szaleć. Niestety, ani słowem nie zająknięto się o tym, jak właściwie zamierzano to wszystko osiągnąć.

***

W pewnym momencie, tak od 2020 roku, trzecia fala zaczęła nabierać realnych kształtów, a w branży zaczął funkcjonować nowy termin — „neurosymboliczna sztuczna inteligencja". I ze wszystkich fikuśnych nazw właśnie tę lubię najbardziej. Przede wszystkim dlatego, że w punkt opisuje to, o co nam tak naprawdę chodzi — łączymy sztuczne sieci neuronowe z wnioskowaniem symbolicznym po to, żeby zachować zalety i wyeliminować słabości obu tych podejść.

A jak to robimy? Ha, każdy orze, jak może. Nie ma utartych wzorców, nie ma dogmatów, nie ma uniwersalnych prawd i zasad, wyrytych na wieki w kamieniu. Jest Dziki Zachód i gorączka złota. Wszystko sprowadza się do tego, jak kreatywni będziemy w łączeniu komponentów neuralnych i symbolicznych. Możliwości są praktycznie nieograniczone, chociaż Henry Kautz, obecnie Professor Emeritus na Uniwersytecie w Rochester, na konferencji AAAI 2020 w Nowym Jorku zaproponował usystematyzowanie takich neurosymbolicznych architektur.

Jedną z nich już poznaliśmy — tę, stosowaną w silnikach szachowych. Wspomniałem wtedy, że najlepsze silniki na świecie wykorzystują obecnie podejście hybrydowe. Ta architektura zaklasyfikowana została jako Symbolic[Neuro]. Oznacza to, że komponent symboliczny pełni funkcję zarządczą, a komponent neuralny jest mu podległy i w razie potrzeby jest wywoływany jako osobny podprogram. W szachach wykorzystuje się heurystyczne metody przeszukiwania drzewa (np. algorytm alfa-beta lub Monte Carlo) do znajdowania możliwie najlepszego ruchu, ale w ramach tego procesu wykorzystuje się również komponent neuralny do ewaluacji stanu planszy.

Osobiście uważam tę kombinację za najbardziej sensowną. Komponent symboliczny, powszechnie uznawany za przewidywalny i wyjaśnialny, kontroluje znacznie mniej przewidywalny komponent neuralny. Bierze z niego to, co najlepsze, a przy okazji powstrzymuje go przed robieniem głupot.

Większość modeli, które tworzymy w Samurai Labs, korzysta z tej właśnie architektury. Dzięki temu wykrywają rzadkie i niebezpieczne zjawiska, zgłaszając przy tym minimalny odsetek fałszywych alarmów, co z kolei pozwala na proaktywną ochronę społeczności internetowych.

Odwrotnością tej architektury jest oczywiście Neuro[Symbolic], gdzie to komponent neuralny wywołuje komponent symboliczny do realizacji określonego

zadania. Przykładem może być integracja ChatGPT z potężnym silnikiem obliczeniowym Wolfram Alpha. Załóżmy, że zadajemy pytanie, które wymaga przeprowadzenia skomplikowanych obliczeń matematycznych. Ostateczną odpowiedź da nam model generatywny, ale jak tylko zorientuje się, że trzeba liczyć, to zawoła do pomocy swojego symbolicznego kumpla.

Gdy oba komponenty współpracują na równych zasadach, mamy do czynienia z architekturą Neuro|Symbolic. Najczęściej sprowadza się to do tego, że komponenty realizują niezależne zadania i komunikują się ze sobą w celu osiągnięcia jak najlepszych rezultatów. A gdy do uczenia modelu statystycznego wykorzystuje się dane pozyskane w procesie wnioskowania symbolicznego, mówimy wtedy o architekturze Neuro:Symbolic→Neuro.

Z jedną propozycją Henry'ego Kautza szczególnie trudno jest mi się zgodzić. Architektura Symbolic Neuro Symbolic miałaby oznaczać tylko tyle, że nasze symbole przekształcamy w wektory, które stanowią wejście i wyjście sieci neuronowej. A to oznaczałoby, że w zasadzie wszystkie współczesne osiągnięcia z dziedziny przetwarzania języka naturalnego, wliczając w to generatywne modele językowe oraz BERT-a z rodziną, należałoby wrzucić do jednego wielkiego pudełka z napisem „neurosymboliczna sztuczna inteligencja". Nie, nie, nie — co za dużo, to niezdrowo.

Zaproponowana lista oczywiście nie wyczerpuje tematu. Jedynym, co tak naprawdę nas ogranicza, jest nasza wyobraźnia. I mocno trzymam kciuki za opracowywanie zupełnie nowych, nieszablonowych rozwiązań. Neurosymboliczna sztuczna inteligencja, mimo że nadal jest kierunkiem dość niszowym, uczy nas, żeby jak ognia unikać doktrynerstwa i fanatyzmu. Słabości poszczególnych podejść należy badać, a nie udawać, że ich nie ma. To najlepszy sposób na ich przezwyciężenie.

Dzięki temu możemy wyprzedzać swoje czasy o całe dekady. Nie mam wątpliwości, że proaktywne rozwiązania chroniące nas przed cyberprzemocą i dbające o nasz dobrostan psychiczny są przyszłością. Zwłaszcza w sytuacjach, w których każda sekunda ma znaczenie, a ceną za nadmierną zwłokę może być ludzkie życie. I bardzo możliwe, że sztuczna inteligencja, konsekwentnie płynąc swoim głównym nurtem, za jakiś czas tam dotrze. Ale to przecież nie powód, żeby ślepo przeć naprzód, nie rozglądając się na boki w poszukiwaniu sekretnych ścieżek, które już dziś pozwolą nam to życie ratować.

OBYŚ ŻYŁ W CIEKAWYCH CZASACH

Wiem, wiem, to stare chińskie przekleństwo jest nadużywane prawie tak często, jak leki na ADHD przez amerykańskich nastolatków. Ale pasuje, szczególnie na zakończenie. Gdy zaczynałem pisać tę książkę, jeszcze w 2023 roku, najmocniejszym modelem ze stajni OpenAI był GPT-4, udostępniony publicznie w ramach subskrypcji ChatGPT Plus w marcu 2023. I taki stan rzeczy utrzymywał się przez ponad rok, co w świecie sztucznej inteligencji może wydawać się wiecznością. A później w 2024 roku ruszyła prawdziwa lawina…

W kwietniu wypuszczono jego szybszą i tańszą wersję — GPT-4 Turbo. W maju zaprezentowano światu nowy flagowy model — GPT-4o, który przyjmuje i zwraca dowolną kombinację tekstu, dźwięku i obrazu, gdzie literka „o" w nazwie pochodzi od łacińskiego słowa „omni", które ma się nam kojarzyć z wszechstronnością. W lipcu udostępniono jego odchudzoną wersję — GPT-4o mini, a już we wrześniu dostaliśmy nie jeden, ale aż dwa nowe modele w wersji beta — o1-preview oraz o1-mini, które zdaniem twórców znacznie lepiej od poprzedników radzą sobie ze złożonym rozumowaniem i wnioskowaniem. Niewykluczone, że gdy niniejsza książka trafi w końcu do rąk Szanownego Czytelnika, dostępne będą już kolejne, zupełnie nowe generatywne modele językowe.

A co tak naprawdę dostaliśmy? Nie popiszę się przesadną oryginalnością, przyrównując GPT-4o do asystenta głosowego z filmu *Ona* (*Her*) z 2013 roku, któremu głosu użyczyła Scarlett Johansson. Niczym główny bohater, odgrywany przez Joaquina Phoeniksa, możemy prowadzić z nim ożywione dysputy w czasie rzeczywistym

za pomocą głosu lub tekstu, jednocześnie pokazując „na kamerce" to, o czym akurat chcielibyśmy porozmawiać.

Na stronie OpenAI pojawiła się cała masa filmików prezentujących możliwości nowego flagowca. Nauka hiszpańskiego poprzez wskazywanie obiektów i tłumaczenie ich nazw? Jest! Interaktywna pomoc w rozwiązaniu zadania domowego z geometrii? Jest! Wspólna gra w papier, kamień, nożyce? Jest! Wystarczyła dekada, żeby wizja Spike'a Jonze'a, reżysera i scenarzysty, który za *Her* zgarnął Oscara, stała się rzeczywistością.

Niestety, najistotniejszym i wciąż nierozwiązanym problemem generatywnych modeli językowych pozostają halucynacje. Modele z serii o1 w benchmarkach przeprowadzonych przez OpenAI wypadają pod tym względem lepiej niż poprzednicy, ale ich autorzy w tych samych oficjalnych raportach ujawniają — bo nie mogą tego legalnie zataić — że dostają również feedback świadczący o dokładnie przeciwnych tendencjach. Nie brakuje przy tym doniesień ekspertów i zwykłych użytkowników twierdzących, że od czasu GPT-4 każdy kolejny model radzi sobie z tym niewygodnym problemem coraz gorzej. Sytuacji nie poprawia fakt, że nowe modele stają się coraz bardziej przekonujące, a co za tym idzie — coraz łatwiej jest nam wierzyć w ich halucynacje.

Powinniśmy zdawać sobie sprawę, że oto na naszych oczach uformował się wyścig, a my oglądamy walkę na śmierć i życie o dominację w branży. OpenAI pokazało GPT-4? Google odpowiedział modelem Gemini. OpenAI pokazało GPT-4o? Dzień później Google zapowiedział Project Astra, który przedstawia się jako „asystenta AI przyszłości". W grę wchodzi nawet odgrzewanie starego kotleta, czyli reboot projektu słynnych okularów AI — Google Glass. Desperacja? Być może, ale Meta i Amazon też nie będą grzecznie stały z boku, biernie się temu wszystkiemu przyglądając.

A gra toczy się o niewyobrażalne wręcz stawki. Do Wielkiej Piątki amerykańskiego Big Techu zalicza się Alphabet (Google), Amazon, Apple, Meta (Facebook) oraz Microsoft, który mocno przytulił się do OpenAI. Wszystkie są na giełdzie, wszystkie mają udziałowców, którzy bacznie śledzą ich wszelkie ruchy, sukcesy i porażki. Nikt nie chce zostać z tyłu, bo strata na poziomie 10% może przekraczać roczny budżet średniej wielkości państwa, takiego jak Polska.

Będziemy świadkami swoistego spektaklu, w którym naukowcy i inżynierowie będą stawać na głowie, próbując nadążyć za obietnicami gruszek na wierzbie, składanymi przez PR i marketing. Dostaniemy nowe fantastyczne modele — to fakt, ale ta bańka nie może rosnąć w nieskończoność. Nie bez przestrzeni na refleksję i krytyczne spojrzenie na coraz poważniejsze problemy i wyzwania. W końcu komuś powinie się noga, jak CD Projektowi przy premierze *Cyberpunka 2077*, i historia zatoczy koło, a świat obudzi się w kolejnej zimie AI.

***

A co my, maluczcy, możemy zrobić? Najważniejsze to nie dać się zwariować. Sztuczna inteligencja bez wątpienia odmieni nasze życie. Już to robi. Pamiętajmy jednak o wyścigu i o tym, że większość informacji, które do nas docierają, są de facto materiałami marketingowymi, zręcznie tworzonymi na potrzeby tego wyścigu przez najlepszych fachowców na świecie.

Niezależne badania wymagają czasu i zasobów. O tym, że ChatGPT zdał amerykański egzamin licencjonowania medycznego (USMLE), mogliśmy przeczytać w lutym 2023 roku. Na informację, że oblał polski egzamin specjalizacyjny z interny, musieliśmy czekać aż do listopada. Czy to w jakikolwiek sposób umniejsza modelowi OpenAI? Bynajmniej. Po prostu dowiedzieliśmy się o nim ciut więcej.

Powstają nowe produkty, ba, całe firmy, zbudowane wokół generatywnych modeli językowych. To one poprzez rynek pokażą nam, gdzie tak naprawdę znajdują się granice ich stosowalności. Do czego nadają się doskonale, a do czego trochę gorzej. Niezależni naukowcy też nie pozostają obojętni — opracowują nowe metodyki badań i konsekwentnie odzierają te modele z magii i marketingowego „bullshitu". Douglas Adams w *Autostopem przez galaktykę* zapytał: „Nie wystarczy zobaczyć, że ogród jest piękny, trzeba wmawiać sobie, że mieszkają w nim wróżki?". Mnie wystarczy. Im mniej wróżek, tym lepiej.

Na wiele z tych rzeczy nie mamy wpływu. Ale jest coś, o co może zawalczyć każdy z nas. Parafrazując hasło Billa Clintona z kampanii prezydenckiej z 1992 roku — „Edukacja, głupcze!". Niezależnie od naszych kompleksów Polska należy do cywilizacji zachodniej. A tam, jak mi się zdaje, niewiele już może nas zaskoczyć. Dlatego

proponuję przyjrzeć się… Chinom. Możemy zarzucić im bardzo wiele, ale z całą pewnością nie to, że ich działania są krótkowzroczne.

W 2023 roku Chińska Administracja Cyberprzestrzeni (CAC) zapowiedziała wprowadzenie ograniczeń czasu korzystania z internetu na urządzeniach mobilnych dla nieletnich. Między 22 a 6 rano ma obowiązywać całkowity zakaz, a pozostałe limity mają być ustalane w zależności od wieku. Dzieci do 8. roku życia mogą korzystać ze smartfonów 40 minut dziennie, starsze — godzinę, a młodzież w wieku 16 – 18 lat — aż dwie godziny. Z tym, że tę ostatnią, libertyńską opcję mogą wyłączyć im rodzice.

Nie są to pierwsze tego typu ograniczenia. W 2021 roku władze Chin wprowadziły całkowity zakaz korzystania z gier online w dni powszednie, obejmujący wszystkie osoby niepełnoletnie do 18. roku życia. Ale to nie znaczy, że chińska młodzież nawala w gierki przez całe weekendy. Co to, to nie. Owszem, można pograć, ale tylko w godzinach 20 – 21 w piątki, soboty, niedziele i święta ustawowe.

I raczej nie chodzi o ograniczenie małolatom dostępu do zgniłej propagandy Zachodu, bo Wielki Chiński Firewall skutecznie blokuje Google'a, YouTube'a czy Facebooka. Poza tym z restrykcji wyłączone są aplikacje i serwisy edukacyjne. Że kontrola i ograniczenie wolności? Serio? Bo na Zachodzie nie ogranicza się wolności dzieci, zakazując sprzedawania im alkoholu, papierosów, a ostatnio nawet i napojów energetycznych? Może to ryzykowna opinia, ale myślę, że tu naprawdę może chodzić o dobrostan chińskiej młodzieży.

Szanowny Czytelnik zapewne słyszał o TikToku — chińskiej aplikacji firmy ByteDance, która niczym czarna dziura pochłania każdą wolną chwilę dzieciakom na całym świecie. Ale czy na pewno? Są kraje, jak np. Indie, gdzie wprowadzono całkowity zakaz korzystania z TikToka. Podobnie postąpił amerykański stan Montana. W wielu krajach częściowy zakaz dotyczy wszystkich urządzeń rządowych. Tak jest między innymi w Stanach Zjednoczonych, Wielkiej Brytanii, Kanadzie i kilku państwach Unii Europejskiej. A jak to wygląda w Państwie Środka?

Niespodzianka! W Chinach nie ma TikToka. Jest jego brat bliźniak Douyin, stworzony przez tę samą firmę — ByteDance. Z wyglądu bardzo podobni, różną się diametralnie, jeśli chodzi o sposób użytkowania. Dzieciaki do 14. roku życia korzystające

z Douyin mogą „scrollować" do 40 minut dziennie między 6 a 22. Co więcej, mają dostęp wyłącznie do treści bezpiecznych dla dzieci.

W TikToku bezpieczne treści dla niepełnoletnich też niby są, ale tę funkcję można bardzo łatwo wyłączyć. Nie znam danych dla Polski, ale w 2022 roku amerykańskie dzieciaki spędzały przy nim średnio prawie 2 godziny dziennie. Średnio, bo aż 13% nastolatków „scrollowało" dzień w dzień przez ponad 4 godziny. Biorąc pod uwagę trendy, założyłbym, że te liczby raczej nie malały w kolejnych latach.

A to nie jedyne różnice. W Douyin największą popularnością cieszą się treści edukacyjne, pomagające doskonalić umiejętności i stawiające na rozwój osobisty. W TikToku… no cóż, powiedzmy, że trzy najpopularniejsze kategorie to rozrywka, taniec i pranki, czyli niewybredne psikusy, z reguły zabawne tylko dla prankującego, a niekoniecznie dla prankowanego. Co więcej, treści obu aplikacji się nie mieszają. W Douyin nie uświadczymy filmów spoza Chin i vice versa w przypadku TikToka.

Nie jestem wielkim fanem regulacji. Uważam, że państwo powinno ingerować wyłącznie tam, gdzie jednostka nie jest w stanie poradzić sobie sama. Ale obserwując rodziców prowadzących dzień po dniu nierówną walkę z TikTokami tego świata, zaczynam wierzyć, że to jest właśnie taki przypadek. Nie mam jednak złudzeń. Jeśli Polska nie poradziła sobie z opodatkowaniem big techów, to naiwnością byłoby sądzić, że będzie w stanie wymusić cokolwiek na cyfrowych gigantach. Być może Unia Europejska dałaby radę, gdyby tylko uznała to za istotny problem.

Dlatego apeluję — bierzmy sprawy w swoje ręce, rozmawiajmy z rodziną i z przyjaciółmi. Zdaję sobie sprawę, że pisząc te słowa w popularnonaukowej książce o sztucznej inteligencji, zwracam się do bardzo wąskiej, świadomej i odpowiedzialnej części społeczeństwa. Stawiam się tym samym w roli wiejskiego proboszcza, który grzmi z ambony do wiernych zgromadzonych na mszy, że nie chodzą do kościoła. Trudno, niech i tak będzie. Bo o ile rozwój AI w nadchodzących latach widzę raczej w świetlanych barwach, to wpływ technologii na edukację i zdrowie psychiczne najmłodszych postrzegam jako poważne i stale rosnące zagrożenie.

POSŁOWIE

Ponieważ okazja do napisania tych słów nadarzyła się na miesiąc przed premierą książki, a dobrych kilka miesięcy po jej ukończeniu, zdecydowałem się z niej skorzystać, by krytycznie odnieść się do moich wcześniejszych przewidywań i zweryfikować, czy okazały się one słuszne.

Na pierwszy ogień weźmy zagrożenia związane z edukacją. Gdy pisałem o rozleniwiającym wpływie sztucznej inteligencji, powoli odbierającej nam kreatywność i umiejętność myślenia krytycznego, były to wyłącznie moje domysły. Na pierwsze prace w tym temacie musieliśmy czekać aż do 2025 roku, a wśród nich warto wymienić choćby *AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking* czy *The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers*.

Oba badania wskazują, że korzystanie z narzędzi AI może wiązać się z ograniczeniem zdolności myślenia krytycznego oraz tzw. deskillingiem, czyli utratą lub degradacją posiadanych umiejętności specjalistycznych na skutek postępującej automatyzacji. Warto przeciwdziałać tym procesom, wprowadzając w życie dobre praktyki, które można by nazwać higieną pracy z AI. Do takich praktyk możemy zaliczyć nieużywanie AI do prostych zadań, z których nie uzyskamy żadnej większej korzyści poza nakarmieniem własnego „lenistwa" — na wzór korzystania ze schodów zamiast z windy, gdy chcemy dostać się z parteru na pierwsze piętro.

***

Teza, że AI najmocniej uderzy w stanowiska juniorskie i tzw. *entry-level jobs*, również okazała się prawdziwa, nad czym szczerze ubolewam. Portale rekrutacyjne donoszą, że liczba ofert pracy na te stanowiska spada z roku na rok. Szczególnie interesującym przypadkiem jest tu branża IT, gdzie pojawienie się narzędzi takich jak Cursor czy Windsurf, integrujących środowisko programistyczne z generatywną sztuczną inteligencją, w zasadzie zmarginalizowało zapotrzebowanie na początkujących programistów.

Widać to wyraźnie wtedy, gdy rozpatrujemy kompetencje na poziomie całego zespołu, a nie jego poszczególnych członków. Zespół składający się z pięciu osób wspieranych przez AI może dziś wykonywać tę samą pracę, która wcześniej wymagała sześciu osób. A to oznacza, że tej szóstej osoby nikt już nie zatrudni, zwłaszcza że rekrutacja na stanowiska juniorskie od zawsze wiązała się ze specyficznym balansem pomiędzy korzyścią wynikającą z ich pracy a inwestycją wynikającą z konieczności ich wdrożenia i przeszkolenia.

***

Na koniec chciałbym jeszcze wrócić do tematu AGI, czyli ogólnej sztucznej inteligencji. Po pierwsze dlatego, że nie podoba mi się cyniczna próba definiowania AGI w oparciu o kryteria ekonomiczne, takie jak opłacalność wykonywanych przez nią zadań. Po drugie dlatego, że termin został zawłaszczony przez marketing i jest nam dziś sprzedawany jako coś, co jest już w zasięgu ręki i za moment zrewolucjonizuje wszystkie aspekty naszego życia.

Czy nazwalibyśmy inteligentnym człowieka, który rozwiązuje nawet bardzo skomplikowane zadania, ale gdy tylko użyjemy nieco innych słów lub wartości liczbowych, natychmiast gubi się i podaje błędne odpowiedzi?

W świetle kolejnych badań coraz wyraźniej widać, że duże modele językowe to wciąż „tylko" bardzo wysublimowany *pattern matching* (dopasowywanie wzorców) i zapamiętywanie. Termin „prompt engineering", który pojawił się wraz z generatywną sztuczną inteligencją, oznacza formułowanie i dostosowywanie poleceń dla modeli AI w celu uzyskania jak najlepszych wyników. W skrócie: tworząc lepsze zapytania, mamy szansę dostać lepszą odpowiedź.

Odwracając to stwierdzenie, możemy powiedzieć, że dla dowolnego zapytania, które zdaje się działać, istnieje równoważna parafraza, na której model polegnie, a którą człowiek zrozumie bez większego problemu. Modele uodparnia się na takie zabiegi przypadek po przypadku, zapytanie po zapytaniu, w ramach klasycznej zabawy w kotka i myszkę.

A czy nazwalibyśmy inteligentnym człowieka, który musi zobaczyć tysiące różnych wariantów, aby dostrzec prosty wzorzec, który małe dziecko dostrzega już po kilku pierwszych przykładach?

Tu poza wymaganą ekspozycją na bodźce kryje się jeszcze jedna pułapka, w którą i mnie zdarzyło się wpaść, widząc, jak wspaniale duże modele językowe radzą sobie z wszelkiej maści egzaminami i olimpiadami przedmiotowymi. Chodzi o odróżnienie posiadania pewnych konkretnych, nawet bardzo licznych i skomplikowanych umiejętności od procesu pozyskiwania zupełnie nowych kompetencji. To drugie wydaje się znacznie lepszą miarą inteligencji, prawda? A już na pewno jest czymś, czego powinniśmy wymagać od ogólnej sztucznej inteligencji.

Warto wspomnieć tu o ARC-AGI, specjalnym teście stworzonym przez François Cholleta, który ma za zadanie weryfikować, jak dobrze sztuczna inteligencja radzi sobie z przyswajaniem nowych umiejętności. Gorąco zachęcam, by pogooglować i popróbować samodzielnie, bo zadania wbrew pozorom wcale nie są trudne. Przeciętny człowiek rozwiązuje 75% z nich, co bystrzejszy ponad 95%, a flagowe modele takie jak GPT-4o, Claude 3.5 Sonnet czy o1-preview rozwiązują… od kilku do kilkunastu procent.

Niestety, z ARC-AGI i podobnymi testami wiąże się ten problem, że część danych trzeba upubliczniać jako zbiór treningowy i po pewnym czasie nie sposób już stwierdzić, czy kolejne modele, osiągające znacznie lepsze wyniki, nadal wykazują się inteligencją, czy już tylko zapamiętywaniem. I właśnie dlatego powstaje już kolejny benchmark, ARC-AGI-2, a jego twórcy formułują następujący wniosek: dopóki jesteśmy w stanie wymyślać kolejne zadania łatwe dla zwykłych ludzi i trudne dla AI, dopóty nie może być mowy o żadnej AGI…


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion