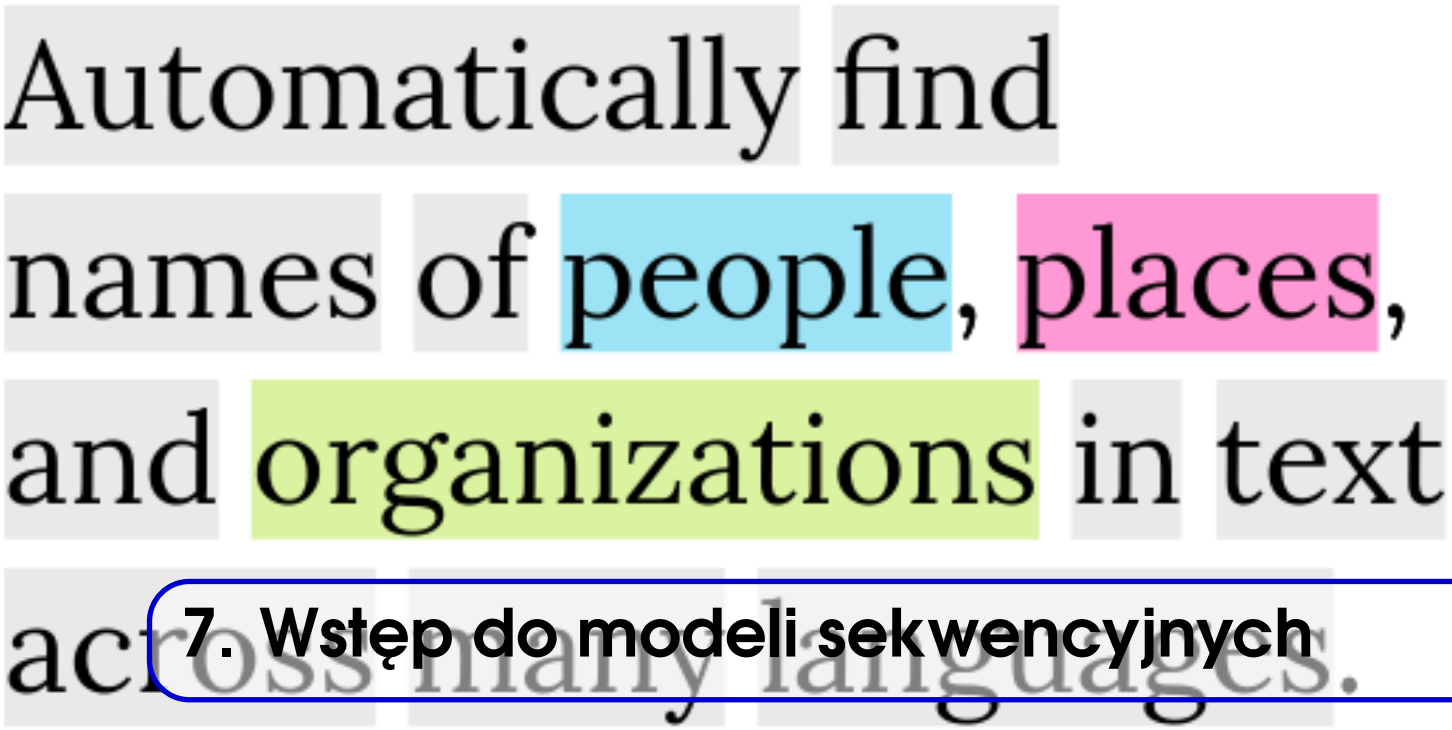


# 7. Wstęp do modeli sekwencyjnych

## 7.1 Zamiast motywacji: przykłady antagonistyczne

Niezwykle interesującym fenomenem, który został początkowo zaobserwowany na modelach głębokich, a potem jego istnienie pokazano również na innych modelach np. regresji logistycznej, jest istnienie przykładów antagonistycznych (ang. *adversalial examples*). Są one swoistymi atakami hakerskim na modele uczenia maszynowego, a aby wyjaśnić ich działanie posłużymy się przykładem. Dany obrazek przedstawiający znak drogowy „STOP" prezentujemy klasyfikatorowi, który prawidłowo przypisuje mu klasę „znak STOPu" z bardzo dużą pewnością. Następnie do obrazka dodajemy niewidoczny dla ludzkiego oka szum, który sprawia, że dany obrazek jest już klasyfikowany jako np. „droga z pierwszeństwem" również z bardzo dużą pewnością.

Nie jest to więc klasyczny błąd klasyfikacji w czasie testowania – system umie klasyfikować ten obrazek prawidłowo o ile nie byłoby – zupełnie niewidocznego dla człowieka – szumu. Pokazano, że tak spreparowany obrazek może zostać z powodzeniem zostać wydrukowany np. na kartce papieru zwykłą drukarką, która przylepiona do znaku drogowego nie robi żadnej różnicy kierowcom-ludziom a powoduje drastyczną różnicę systemom SI. Inna możliwość to wydruk specjalnej koszulki, która sprawia że stajesz się „niewidzialny" dla systemu wykrywającego osoby w czasie rzeczywistym (np. architektura YOLO i pochodne). Nie trzeba chyba nikogo uświadamiać w zakresie potencjalnych zastosowań takiego ataku przez złośliwe trzecie strony – wystarczy wyobraźnia czytelnika.

Powstaje pytanie w jaki sposób takie przykłady są tworzone, wszak dodanie małego losowego szumu do obrazka nie wpływa zwykle na zmianę klasy, co najwyżej na spadek pewności. Najbardziej klasyczną metodą jest szybka metoda znaku gradientu (ang. *fast gradient sign method, FGSM*), która polega na wykonaniu kroku „deoptymalizacji" algorytmu stochastycznego spadku wzdłuż gradientu. W czasie treningu uczymy sieci zgodnie z regułą:

$$\mathsf{w} = \mathsf{w} - \eta \nabla_{\mathsf{w}} Loss(x, y)$$

wykonując krok w kierunku przeciwnym do gradientu (kierunku najszybszego wzrostu funkcji), aby zminimalizować daną funkcję straty. W przypadku gdy chcemy system oszukać to chcemy zmaksymalizować jego funkcję straty, jednak – zakładając brak bezpośredniego włamania i możliwości modyfikacji kodu źródłowego – nie możemy zmienić wag sieci neuronowej w systemie. Jednak nic nie stoi na przeszkodzie aby policzyć gradient nie względem wag, ale względem... samego obrazka. Otrzymujemy więc:

$$\mathrm{x} = \mathrm{x} + \eta\, sign(\nabla_{\mathrm{x}} Loss(x, y))$$

gdzie $sign()$ to funkcja zwracająca $+1$ dla liczb większych od 0 i $-1$ dla liczb ujemnych, która jest stosowana do każdego elementu wektora gradientu. Jej dodanie wynika z większej praktycznej skuteczności (obserwacja empiryczna).

Na pozór użycie tej metody wymaga dostępu do wag sieci w celu policzenia gradientu, wystarczy jednak mieć dostęp do ciągłego wyjścia sieci i wykorzystać numeryczne przybliżenie gradientu formułą różnicową (patrz: Optymalizacja Ciągła). Ponadto, co ciekawe (i trochę frapujące) przykłady antagonistyczne stworzone dla jednej architektury sieci z dużą szansą zadziałają też dla sieci o innej architekturze.

Stosowanie metod atakujących systemy uczące przy pomocy przykładów antagonistycznych jest szczególnie wygodne dla danych wizyjnych, ponieważ mają one (prawie) ciągłą naturę. Możemy więc bez problemu dodać do piksela trochę szumu, odrobinę go pociemniając lub pojaśniając, jednocześnie zachowując niezmieniony odbiór całego obrazu przez człowieka. Tekst ma jednak zupełnie inną, tj. dyskretną, naturę. Oczywiście słowo możemy zastąpić innym losowym słowem, jednak zostanie to natychmiast wychwycone przez człowieka i zmieni sens wypowiedzi. Z kolei wprowadzanie literówek czy też wprowadzanie drobnych modyfikacji w kolejności liter w słowie może zarówno zostać wykryte przez człowieka jak i przez niezawodny mechanizm obronny: korektor pisowni (ang. *spell checker*). Została jednak zaproponowana metoda tworzenia przykładów antagonistycznych poprzez zastępowanie słów ich.. synonimami.

Metoda prawdopodobieństwa ważonego istotnością słów (ang. *probability weighted word saliency, PWWS*) tworzy przykłady antagonistyczne dla klasyfikacji tekstu w trzech krokach. Po pierwsze dla każdego słowa w klasyfikowanym zdaniu/dokumencie tworzy się listę kandydatów. Jeśli słowo jest jednostką nazewniczą (ang. *named entity*) czy np. nazwą miejsca, firmy czy nazwiskiem to kandydatami są inne jednostki nazewnicze o tym samym typie. Jeśli słowo jest zwykłym słowem to synonimu szuka się w słowosieci. Następnie dla każdego słowa wybiera się najlepszego kandydata na zastępnika poprzez sprawdzenie jak duży spadek pewności klasyfikatora spowoduje on dla prawdziwej klasy. Konkretnie, podmienia się dane $i$-te słowo na kandydata, przepuszcza przez klasyfikator i odczytuje się pewność klasyfikacji do prawidłowej klasy:

$$\Delta P_i^* = \underbrace{P(y_{true}|\mathrm{x})}_{\text{klasyfikacja przykładu bez modyfikacji}} - \underbrace{P(y_{true}|\mathrm{x}_\mathrm{i}^*)}_{\text{z podmienionym słowem}}$$

Na końcu tego etapu mamy więc dla każdego słowa wybranego jego najlepszego zastępnika.

W drugim etapie określamy istotność wpływu danego słowa na wynik klasyfikacji poprzez sprawdzenie jak bardzo zmieniłaby się klasyfikacja gdybyśmy to słowo zastąpili tokenem $\boxed{\text{UNK}}$ . Stosujemy ten sam wzór co wyżej, a jego wynik oznaczymy $\Delta S_i$ od

istotności słowa (ang. *saliency*). Następnie wartości istotności $\Delta S_i$ normalizuje się dla całego zdania poprzez zastosowanie funkcji softmax.

W trzecim etapie zastępujemy słowa na ich odpowiedniki tak długo aż klasyfikator się pomyli. Kolejność zastępowania słów jest określona poprzez

$$softmax(\Delta S)_i \cdot \Delta P_i^*$$

czyli różnicy prawdopodobieństwa zastąpienia zastępnika ważonej znormalizowaną istotnością (patrz: nazwa metody). Metoda pomimo swojej prostoty powoduje spadek trafności zarówno modeli opartych na sieciach splotowych jak i rekurencyjnych z 85-90% do nawet 2% na niektórych zbiorach danych [2]. Jednocześnie trafność człowieka na wygenerowanych przykładach pogarsza się jedynie o ok. 5%.

Zaimplementowanie metody w zasadzie nie powinno przysporzyć problemu są jednak dwa „ale":

- metoda potrzebuje kandydatów dla jednostek nazewniczych – w jaki sposób wykryć w tekście jednostki nazewnicze?
- metoda szuka synonimów w słowosieci, jednak słowa w słowosieci mają wiele różnych znaczeń i co za tym idzie różne synonimy. W jaki sposób wybrać właściwe znaczenie? Dla języka angielskiego dużym uproszczeniem byłoby określenie części mowy słowa (np. *book* - książka (rzeczownik) i rezerwować (czasownik)). Znów... jak to zrobić?

Takimi problemami zajmiemy się w tym module.

## 7.2 Znakowanie części mowy

Znakowanie części mowy (ang. *part-of-speech tagging, PoS*) polega na przypisaniu każdemu tokenowi w tekście jednej z etykiet (klas) będącej przypisaniem do części mowy. Zadanie to, choć na pozór bardzo „lingwistyczne" i niepraktyczne, ma bardzo dużo zastosowań i stanowi częsty element potoku przetwarzania tekstu w systemach inżynierii lingwistycznej. Dla języka angielskiego część mowy czasami determinuje umiejscowienie akcentu w wyrazie np. „absent" akcentowane na pierwszą sylabę jest przymiotnikiem a na drugą sylabę czasownikiem, podobnie „rebel", „incline", „discourse" czy... „accent" są rzeczownikami jeśli są akcentowane na pierwszą sylabę i czasownikami jeśli akcent jest na drugiej sylabie. Wykrycie poprawnej części mowy jest więc niezbędne do konstrukcji systemu syntezy mowy (ang. *text-to-speech*). Część mowy (nawet w wersji worka słów) jest też użyteczną cechą dla klasyfikacji tekstu, a szczególnie analizy sentymentu np. słowo-klucz „like" jako czasownik jest pozytywne, a jako przymiotnik i przyimek raczej neutralne. Podobnie „but" jako spójnik wprowadza zdanie przeciwstawne, ale jako przyimek jest neutralne. Wykrycie części mowy stanowi też etap wstępny do analizy syntaktycznej zdania, która pozwala na modelowanie znaczenia zdania (zagnieżdżenia słów modelują znaczenie słowa) i jest używana w systemach ekstrakcji wiedzy.

Ponadto niektórzy praktycy używają części mowy dla angielskiego jako (słabe) ujednoznacznianie angielskich słów, ponieważ systemy dla ujednoznaczniania często są dość skomplikowane i nie oferują znacznych zysków ponad części mowy. Jednocześnie to ostatnie zastosowanie sygnalizuje główną trudność w określaniu części mowy: wymaga ono często poradzenia sobie z niejednoznacznością słów i określenie jego części mowy na

podstawie kontekstu. Rozważając przykładowe zdanie: „The Fed raises interest rates despite pressure from Trump" zauważ że zdecydowana większość słów ma kilka możliwych części mowy: „The [DET] Fed [N/V] raises [V] interest [N/V] rates [N/V] despite [IN] pressure [N/V] from [IN] Trump [N/V]" gdzie IN oznacza przyimek, N rzeczownik, V czasownik, DET rodzajnik. Akurat w podanym zdaniu dwie części można prosto określić na podstawie wielkości liter, jednak często w procesie przetwarzania wstępnego tekstu normalizuje się tekst do małych liter (co jak widać nie jest zawsze dobrym pomysłem). Biorąc pod uwagę cały słownik angielski ok. 85% wyrazów ma tylko jedną część mowy, jednak biorąc pod uwagę częstość występowania słów okazuje się że losowe słowo w angielskim tekście ma jedną część mowy z prawdopodobieństwem $\approx 35\%$ [1].

Najczęściej stosowaną miarą jakości wykrywania części mowy jest trafność klasyfikacji (ang. *accuracy*). Każde słowo/token w zdaniu traktuje się jako przykład testowy i zlicza się liczbę prawidłowych klasyfikacji (przypisań do części mowy). Pomimo tego, że częste słowa mają zwykle kilka części mowy, wyniki osiągane dla tego zadania są stosunkowo wysokie. Dzieje się tak dlatego, że słowa mające wiele części mowy mają najczęściej silną tendencję do występowania w danej części mowy. Rozwiązanie przypisujące słowu jako najczęściej występującą część mowy uzyskuje dla języka angielskiego ok. 90% trafności, choć to zależy m.in. od liczby wykrywanych części mowy. Korpusy bowiem stosują różne zestawy etykiet oznaczających części mowy: niektóre są bardzo małe, a inne rozbudowane. Na przykład wszystkie rzeczowniki możesz opatrzeć tagiem „N" albo możesz je rozdzielić na rzeczowniki pospolite w liczbie pojedynczej „NN", mnogiej „NNS" oraz rzeczowniki-nazwy własne (ang. *proper nouns*) w liczbie pojedynczej „NNP" i mnogiej „NNPS". Liczba etykiet w oczywisty sposób wpływa na trudność zadania i jakość predykcji, dlatego zawsze przed rozpoczęciem pracy nad problemem wykrywania części mowy warto sprawdzić ile wynosi trafność prostego rozwiązania referencyjnego.

Znakowanie części mowy dla języka polskiego jest pod pewnymi względami prostsze niż języka angielskiego. Przede wszystkim w języku polskim możemy dużo wywnioskować o części mowy po prostu z budowy wyrazu (szczególnie po udanej lematyzacji) np. słowa kończące się na „-ować" będą czasownikami, na „-owany" przymiotnikami, a na "-owanie" rzeczownikami. Podobnie do angielskiego, również w języku polskim określenie części mowy może wymagać ujednoznaczniania np. wyraz „lecz" który jest zarówno czasownikiem (Lekarzu lecz się sam!) lub spójnikiem. Wydaje się jednak, że jest to jednak zjawisko dużo rzadsze niż w języku angielskim. Z drugiej strony, znakowanie części mowy dla języka polskiego często odbywa się z bardzo dużym zestawem etykiet (np. rzeczownik, jego przypadek, jego rodzaj, liczba itd.).

**Ćwiczenie 7.1** Wejdź na stronę `https://parts-of-speech.info/` i przetestuj system wykonujący znakowanie części mowy w praktyce. Zwróć uwagę, że kolorami znakowane są tylko ogólne kategorie części mowy, ale po najechaniu na słowo można zobaczyć pełen tag słowa. ∎

## 7.3 Rozpoznawanie jednostek nazewniczych

Zadanie znakowania sekwencji może być wykorzystane także w innych problemach – w szczególności można przetransformować do niego zadanie jednoczesnej segmentacji i znakowania. Jednym z najpopularniejszych tego typu problemów jest rozpoznawanie jednostek nazewniczych/identyfikacyjnych/referencyjnych (ang. *named entity recognition,*

*NER*), które polega na wykryciu w tekście nazw własnych i określenie ich kategorii. Do najpopularniejszych typy jednostek nazewniczych to: jednostki geograficzne (GEO), organizacje (ORG), osoby (PER), jednostki geopolityczne (GPE), wskaźniki czasu[1] (TIM) czy wydarzenia (EVE). Należy jednak podkreślić, że kategorie silnie zależą od obszaru zastosowania np. przy konstrukcji systemu dialogowego dla systemów SmartTV możemy być zainteresowani wykryciem jednostek takich jak: tytuł filmu, nazwa serialu, reżyser, aktor, bohater (filmowy). Umiejętność wykrycia takich jednostek nazewniczych w wypowiedzi użytkownika pozwala na ekstrakcje właściwych słów, wpisanie ich do zapytania do bazy danych oraz dostarczenie odpowiedzi użytkownikowi. Innym możliwym zastosowaniem jest np. aplikacja która automatyczne wykrywa w mailu datę, miejsce i cel spotkania a następnie automatycznie tworzy odpowiednie wydarzenie w kalendarzu. Wykrywanie jednostek nazewniczych jest podzadaniem ekstrakcji informacji czyli działu inżynierii lingwistycznej zajmującego się wydobywaniem faktów, encji i powiązań pomiędzy nimi z czystego tekstu.

Zadanie wykrywania jednostek nazewniczych można częściowo rozwiązać poprzez stosowanie list miast, imion, państw czy firm – do których często odnosimy się jako „gazetteers". Inną możliwością jest stworzenie takich list bezpośrednio z korpusu uczącego – nie jest to jednak wystarczające. O ile pełną listę państw można prosto skonstruować np. z Wikipedii to już konstrukcja pełnej listy firm czy miast (w kontekście międzynarodowym) jest niewykonalna. Co więcej pojawia się problem ujednoznaczniania: Marcin Konieczny z Poznania? Mistrz świata w Ironmanie? Weterynarz? A może... :). W niektórych zastosowaniach mamy też do czynienia z klasami dla których brakuje ogólnodostępnych list. Poleganie na liście bytów nazwanych z korpusu uczącego jest dalece niewystarczające - większość bytów nazwanych jest rzadka, co oznacza że nie wystąpi nigdy nawet przy dużym korpusie. Ponadto cały czas pojawiają się nowe nazwy np. firm czy organizacji przez co poleganie na nich tworzy problem ciągłej ich aktualizacji. Przy okazji zasygnalizowaliśmy trudność zadania: dużo jednostek nazewniczych z korpusu testowego nigdy nie pojawiło się w korpusie uczącym – konieczna jest więc umiejętność określania typu jednostki z kontekstu.

Zadanie wykrywania bytów nazwanych możemy przetransformować do problemu znakowania sekwencji poprzez zastosowanie odpowiedniego kodowania. Najprostszym z nich jest kodowania IO (ang. *inside, outside*), które używa tyle etykiet ile jest kategorii jednostek nazewniczych (inside) oraz dodatkowej etykiety specjalnej O (outside). Na przykład: „The [O] Fed [I-ORGANIZATION] raises [O] interest [O] rates [O] despite [O] pressure [O] from [O] Trump [I-PERSON]".

Problemem tego kodowania jest jednak niejasna obsługa jednostek które mają kilka słów np. w zdaniu „Rzecznik [O] firmy [O] Apple [I-ORG] Steva [I-PER] Jobsa [I-PER] Ignacy [I-PER] Kowalski [I-PER]" nie jest jasne ile jest wykrytych osób: cztery? Rozważaniem tego problemu było kodowanie IOB (ang. *inside, outside, beginning*), które stosowało dodatkowe tagi „B-klasa" z chwilą gdy trzeba było rozdzielić byty o tych samych typach np. „Rzecznik [O] firmy [O] Apple [I-ORG] Steva [I-PER] Jobsa [I-PER] Ignacy [B-PER] Kowalski [I-PER]". Kodowanie to jednak zupełnie się w praktyce nie sprawdziło: systemowi ciężko się nauczyć, że musi zastosować inny tag w sytuacji dwóch przylegających encji o tych samych typach. Dużo prostszym z punktu widzenia nauki jest

[1] Wykrywanie znaczników czasu ze względu na szczególną charakterystykę czasami jest traktowane jako oddzielne zadanie: Timex.

kodowanie BIO (ang. *beginning, inside, outside*) które zawsze korzysta z tagów „B-...” przy rozpoczęciu nowej jednostki nazewniczej, a jej ewentualna kontynuacja jest oznaczana poprzez tagi „I-...". Przykładowe zdanie w tagowaniu BIO wygląda w następujący sposób: „Rzecznik [O] firmy [O] Apple [B-ORG] Steva [B-PER] Jobsa [I-PER] Ignacy [B-PER] Kowalski [I-PER]”.

Kodowanie BIO jest najpopularniejsze, ale istnieją także inne schematy kodowania które wprowadza się z myślą osiągnięcia lepszych wyników uczenia. Przykładem może być kodowanie IOBES (ang. *inside, outside, beginning, end, singleton*) które wyróżnia encje będące jednym słowem „S-...” oraz początek „B-...” i koniec „E-...” jednostki nazewniczej posiadającej kilka tokenów. W przypadku bytów z trzema słowami i więcej wszystkie słowa pomiędzy pierwszym i ostatnim są tagowane „I-...”. Niektórzy badacze preferują taki sposób kodowania i czasami działa on lepiej niż BIO, jednak niektóre badania eksperymentalne sugerują że tagowanie BIO jest prostsze w nauce.

Inną krytyką kodowania BIO jest brak możliwości wyrażenia dodatkowych informacji czy brak możliwości rozdzielenia tagowania od stosowania konkretnego typu tokenizacji. Z tego powodu często preferuje się korzystanie z tagów w formacie XML (firma <ORG>Apple</ORG>). Większość systemów uczenia maszynowego jednak – niezależnie od kodowania w korpusie – i tak stosuje wewnętrznie kodowanie BIO lub podobne.

W praktyce nadal często spotyka się zaawansowane systemy regułowe (oparte np. o gramatyki) do wykrywania jednostek nazewniczych – jednak zgodnie z filozofią naszego przedmiotu: skupimy się na podejściach uczenia maszynowego. W przeciwieństwie do zadania znakowania części mowy, gdzie jako miarę jakości stosuje się trafność klasyfikacji, w zadaniu wykrywania jednostek nazewniczych stosujemy uśrednione miary F1. Jest to uzasadnione faktem, że mamy do czynienia ze sporym niezrównoważeniem klas: tag „O” będzie najczęstszym znacznikiem w zbiorze.

## 7.4 Problem predykcji struktur

Jak zasygnalizowaliśmy wcześniej, w problemach znakowania sekwencji dużą rolę odgrywa otoczenie słowa w zdaniu oraz występujące zależności pomiędzy klasami. Nie osiągnie się więc satysfakcjonujących wyników poprzez opisanie słowa cechami i zastosowanie klasyfikacji wieloklasowej – należy uwzględnić w jakiś sposób kontekst i potraktować go jako problem predykcji sekwencji. Jednak, aby dobrze zrozumieć problem osadźmy go w kontekście klasyfikacji wieloklasowej.

W problemie klasyfikacji np. obrazów do klas: pies, kot, człowiek, najprostszy klasyfikator liniowy przybiera postać:

$$\hat{y} = \arg \max_{y \in \{\text{kot, pies, człowiek}\}} \mathsf{w}_y^T \boldsymbol{\phi}(\mathsf{x})$$

gdzie $\boldsymbol{\phi}(\mathsf{x})$ oznacza funkcję obliczającą na podstawie obrazu $\mathsf{x}$ cechy, a $w_y$ to wektor wag związany z daną klasą. Modelowanie problemu sprowadza się do zadania inżynierii cech czyli konstrukcji $\boldsymbol{\phi}(x)$, która w najprostszym przypadku może być nawet funkcją tożsamościową tj. $\boldsymbol{\phi}(x) = x$ – klasyfikacja na samych pikselach. Zadanie uczenia wymaga wyestymowania kilku wektorów $w_y$ – na którym skupiała się cała nasza uwaga z punktu widzenia systemów uczących się. Dokonanie predykcji jest trywialne: wystarczy wypróbować wszystkie klasy (zwykle kilka, może kilkanaście) licząc dla nich ocenę i zwrócić klasę z najwyższą oceną.

Stosując ten mechanizm bezpośrednio do naszego problemu otrzymujemy

$$\hat{\mathrm{y}} = \arg\max_{y \in \mathscr{Y}} w_y^T \phi(\mathrm{x})$$

gdzie $\phi(x)$ jest znów funkcją zamieniającą sekwencję w cechy. Z tym, że nawet w przypadku sekwencji liczb, nie może to już być funkcja tożsamościowa, bowiem sekwencja x ma dowolną długość, a uczony wektor wag powinien mieć stałą długość. Gdyby wektor wag musiałby być coraz dłuższy dla coraz dłuższych sekwencji, procedura uczenia się byłby pozbawiona nadziei na sukces. Nie ma bowiem szans, że w korpusie uczącym znajdziemy wystarczającą liczbę sekwencji o wszystkich możliwych długościach. Kolejną trudnością jest liczba wektorów wag $\mathrm{w}_y$ których jest tyle ile klas. Z tym że klasą w tym przypadku jest cała sekwencja! Zakładając 16 części mowy i sekwencje o konkretnej długości $n$ – wektorów wag jest $16^n$! Przekłada się to również na trudność predykcji, która stała się problematyczna: w jaki sposób możemy efektywnie przeliterować po wszystkich możliwych sekwencjach i zwrócić najlepszą z nich?

Tego typu problemy rodzą się nie tylko w predykcji sekwencji, ale także w większej kategorii problemów jaką jest predykcja struktur (ang. *structure prediction*). Predykcja struktur jest poddziedziną uczenia maszynowego która zajmuje się systemami uczącymi które powinny modelować sekwencje, ramki w bazie danych, drzewa, grafy oraz inne problemy które można przetransformować do postaci grafowej. W stosunku do klasycznych problemów klasyfikacji, gdzie osią prac jest uogólnianie wiedzy z cech na nowe ich kombinacje, w uczeniu struktur pojawia się nowe, wielkie wyzwanie: uogólnianie wiedzy na nowe, niewidziane wcześniej *klasy*! Takie uogólnianie jest konieczne ponieważ liczba klas jest w tych problemach astronomiczna, a np. przy nieograniczonej liczbie węzłów w grafie, wręcz nieskończona.

W obliczu przedstawionych problemów konieczna wydaje się dekompozycja struktury na części składowe. Rozdzielenie struktury na części zwykle umożliwia znacznie prostszą inżynierię cech – zamiast tworzyć cechy opisującą całą strukturę, możemy skupić się na opisywaniu wybranej jej części i potem zastosowanie stworzonych cech iteracyjnie do wszystkich takich części obecnych w strukturze. Z resztą, czy gdyby przyszło nam opisywać całą strukturę i tak nie zastosowalibyśmy takiej strategii? Co więcej, dekompozycja struktury umożliwia nam skalowanie jej na różne rozmiary bez konieczności douczania kolejnych wektorów wag, gdyż te także mogą być współdzielone pomiędzy częściami tych samych typów. Wreszcie, sprytnie zaprojektowana dekompozycja problemu pozwoli nam na opracowanie efektywnych procedur predykcji, niewymagających iterowania po wszystkich możliwościach, ale jednocześnie nadal biorąca je pod uwagę. Tak wykonana procedura predykcji, może także pozwolić na uniknięcie pewnych błędów jeśli popełniony zostanie błąd systemu uczącego się przy ocenie jednej składowej, procedura predykcji nadal wybrać prawidłową strukturą o ile pozostałe części prawidłowego wyniku zostaną wysoko ocenione. Jednakże, źle wykonana dekompozycja problemu może znacznie ograniczyć możliwości systemów uczących się i powodować niskie wyniki.

W tym module skupimy się na modelowaniu sekwencji – czyli specyficznego rodzaju struktury, jednak wiedza ta stanowi solidne podstawy do poznania ogólnych metod predykcji struktur. Tak jak zauważyliśmy wcześniej, wszystkie trzy elementy systemu uczącego się: modelowanie (w skład którego wchodzi dekompozycja problemu), trening/nauka i wnioskowanie/predykcja są tutaj nietrywialne, dlatego dalszy opis algorytmów będziemy dzielić na te właśnie trzy obszary.

# Dodatki

## Materiały dla chętnych

Dla zainteresowanych polecam lekturę całego artykułu wprowadzającego metodę ataku prawdopodobieństwem ważonego istotnością słów [2] oraz obejrzenie bardzo interesującego wykładu (w kontekście rozpoznawania obrazów) o przykładach antagonistycznych `https://www.youtube.com/watch?v=CIfsB_EYsVI&t=1s`.

# Bibliografia

[1] Dan Jurafsky i James H. Martin. *Speech and Language Processing (3rd ed. draft, 16 Oct. 2019)*. 2019.

[2] Shuhuai Ren, Yihe Deng, Kun He, i Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1103. URL `https://www.aclweb.org/anthology/P19-1103`.