# Podstawowe elementy i zagadnienia związane z procesowaniem języka naturalnego

Procesowanie języka naturalnego jest bardzo obszerną dziedziną data science. Pokrywa szeroką gamę zagadnień rozpoczynając od najprostszych typu przewidywanie kolejnego słowa przy pisaniu SMSów na telefonie, po interpretację tekstu oraz tłumaczenie maszynowe. To właśnie zagadnienia związane z tłumaczeniem były tematem pierwszych projektów w obszarze NLP już w latach pięćdziesiątych XX wieku.

Podstawowe elementy przygotowywania danych do analiz związanych z przetwarzaniem języka naturalnego to:

\begin{itemize}

\item tokenizacja, czyli podział tekstu na segmenty, najczęściej pojedyncze słowa,

\item stemming ma na celu obcięcie wszystkich przyrostków i przedrostów aby zbliżyć słowo do podstawowej postaci,

\item lematyzacja to przypisanie do każdego słowa jego formy podstawowej, która go reprezentuje,

\item tworzenie wektorów własnościowych (word embeddings) w uproszczeniu będących wektorową reprezentacją znaczenia danego słowa.

\end{itemize}

# Przygotowanie danych

# Usunięcie wierszy z pustym speech\_raw

* 1. Pozostało w bazie 272 221

# Algorytm przypisania autora tekstu (SQL).

Podczas przeglądania danych zidentyfikowano błędy w przypisaniu id autorów przemówień. Konsekwencją tego jest brak możliwości łatwego przypisania autora danego wystąpienia do jego tekstu. Błędy występowały dla większości wpisów w bazie i mimo tego, że sprawiały wrażenie systematycznej zmiany (poprawna wartość wydawała się bliska z kontekście porządku leksykograficznego), nie udało się ustalić algorytmu, który mógłby przypisać poprawne wartości w sposób automatyczny.

W związku z powyższym przypisanie autora (*author\_final*) wykonano w następujących krokach (10\_ author\_final.sql):

1. Wiele wystąpień zaczyna się od schematu „Poseł Imię Nazwisko:”. Pierwsze występujące w takim kontekście imię i nazwisko przypisano jako dane autora z tekstu (kolumna *author\_by\_text*).
2. Przypisano imię i nazwisko autora w oparciu o *id\_* z bazy (kolumna *author\_by\_id*).
3. Dla przypadków gdzie dla danego id, chociaż raz *author\_by\_text* jest równe *author\_by\_id*, przypisano te wartości jako *author\_final*.
4. Dla pozostałych przypadków sprawdzono jaka wartość *author\_by\_text* pojawia się najczęściej w obrębie danego id. Jeśli wartość była niepusta, została przypisana jako *author\_final*.
5. Pozostałe przypadki to takie, gdzie najczęściej pole *author\_by\_text* było puste, tj. w tekście wystąpienia nie występowało zestawienie „Poseł Imię Nazwisko:”. Wynika to z różnej postaci plików transkrypcją – dla części z nich fragmenty pogrubione na poniższym obrazku nie znalazły się w tekście dostępnym w bazie.



1. W trakcie analizy danych niektóre id były weryfikowanych ręcznie w oryginalnych plikach pdf. Wartości dla nich wprowadzono ręcznie w kodzie.

Po zastosowaniu powyższych kroków w bazie brakowało przypisania *author\_final* dla 795 id (na 3828). Dla pojedynczych fragmentów wystąpień uzupełnienie wyniosło ok 83%.

# Czyszczenie fragmentów przemówień (Python)

Wykonano następujące etapy czyszczenia danych:

* usunięto znaki specjalne \n \r \t
* wykasowano tytuły przemówień pojawiające się na początku każdego fragmentu
* usunięty fragmenty tekstu w nawiasach (np. (Oklaski.), (Dzwonek.))
* Usunięto fragmenty identyfikujące mówcę, tj. fragment „Poseł Imię Nazwisko:”

Następnie przygotowano drugi zestaw tekstów, które oczyszczono jeszcze bardziej z fragmentów mało informacyjnych.

* Wypowiedzi innych posłów i komentarze typu "Oklaski" pojawiające się w nawiasach
* "Poseł Imię Nazwisko:"
* Panie Marszałku!
* Pani Marszałek!
* Wysoka Izbo!
* Panie Ministrze!
* Dziękuję bardzo.

Po tym etapie czyszczenia w bazie pozostało 272 217 fragmentów przemówień, z których 225 385 ma przypisanego autora. Oba zestawy tekstów z różnym poziomem oczyszczenia będą stosowane w różnych analizach.

# Łączenie fragmentów przemówień (python)

Fragmenty tych samych wypowiedzi, zgrupowane po dacie, autorze i tytule został y połączone.

Po usunięciu tekstów, które po wszystkich modyfikacjach stały się puste, w bazie zostało 158 885 tekstów z przypisanym autorem.

# Wstępna analiza danych

Dla tekstów z przypisanym autorem przeprowadzono wstępną analizę danych. Po usunięciu cyfr, znaków interpunkcyjnych oraz słów z listy polskich *‘stop-words’* analizowano częstości w celu weryfikacji i rozszerzenia listy *‘stop-*words’. W wyniku analizy dodanie do niej następujące elementy, ze względu na niską wartość informacyjną i ryzyko wprowadzenia zaburzeń do dalszych analiz:

zł

pkt

art

ustawy

r

Po wykonaniu tego czyszczenia, w korpusie znajduje się nieco ponad 42 mln słów. Przed lematyzacją w korpusie występuje 367 716 różnych słów.

Słów, które można określić jako „rzadkie”, tj. występujące w całym korpusie nie więcej niż 5 razy jest 226 599 (w tym 126382 występuje tylko jeden raz), czyli stanowią one istotną większość. Przykładowe słowa pojawiające się dokładnie jeden raz w całym korpusie:

zaanektować

babette

opowiadaną

odejmowana

trawiona

kolektywistyczna

zagonów

troić

maćkowy

centralizowany

Słowa pojawiające się najczęściej, to:

[('rząd', 95603),

('projektu', 95657),

('prawa', 96092),

('chodzi', 99480),

('panie', 108932),

('projekt', 113847),

('pytanie', 116217),

('pracy', 118301),

('komisji', 146153),

('państwa', 148339)]

Najczęściej wypowiadający się posłowie to:

| **author\_final** | **Liczba slów** | **Liczba wypowiedzi** |
| --- | --- | --- |
| **Stanisław Stec** | 477862 | 1368 |
| **Mirosław Pawlak** | 203379 | 1134 |
| **Romuald Ajchler** | 401879 | 1120 |
| **Jan Kulas** | 547001 | 1101 |
| **Andrzej Szlachta** | 242306 | 982 |

Posłowie z największą liczbą słów w wypowiedziach to:

| **author\_final** | **Liczba slów** | **Liczba wypowiedzi** |
| --- | --- | --- |
| **Jan Kulas** | 547001 | 1101 |
| **Józef Zych** | 514652 | 878 |
| **Jerzy Jaskiernia** | 481512 | 658 |
| **Stanisław Stec** | 477862 | 1368 |
| **Tadeusz Tomaszewski** | 412781 | 917 |

# Generatory przemówień bazujące na modelach n-gramowych

Na tym etapie podjęto decyzję o budowie prostego generatora przemówień opartego o modele n-gramowe. Takie modele bazują na statystykach występowania n-gramów w analizowanym korpusie i mogą służyć do przewidywania kolejnego elementu sekwencji jak również do generowania nowych sekwencji w oparciu o zaobserwowane zależności.

Przykładowo prawdopodobieństwo, że w naszej sekwencji, po słowie „praca” wystąpi słowo „zaliczeniowa” wynosi:

P(zaliczeniowa|praca) = cnt(„praca zaliczeniowa”)/cnt(„praca \_\_\_”)

Gdzie \_\_\_ oznaca dowolne słowo a liczniki „cnt” zostały wygenerowane na korpusie, na którym trenowano model.

# Konstrukcja modeli

Model powstał w dwóch wersjach dla 1-gramów oraz 2-gramów. Dla obu opcji w pierwszym kroku przygotowano słownik zawierający wszystkie występujące kombinacje odpowiednich n-gramów ze słowami po nich następującymi.

Generowanie przemówień polega na przypisaniu słowa początkowego (lub wybraniu losowego) a następnie w sposób wybieraniu słów kolejnych na bazie słownika prawdopodobieństw zbudowanego w poprzednim punkcie.

Wraz ze wzrostem n, rośnie zapotrzebowanie na dane do modelu. W przypadku gdy danych do uczenia jest zbyt mało, model będzie po prostu odtwarzał zdania, które pojawiały się w danych uczących. Z drugiej strony takie modele potrafią generować sekwencje lepszej jakości, bardziej przypominające prawdziwe zdania, ponieważ zachowują więcej logicznych powiązań pomiędzy kolejnymi słowami.

Do zbudowania najprostszych modeli nie były wykorzystane żadne z pakietów związanych z NLP.

Aby uprościć i zautomatyzować konstrukcję modeli, przygotowano klasę ng\_models (ng\_models.py). Obiekt tej klasy jest modelem n-gramowym utworzonym zgodnie z przekazanymi parametrami *n* oraz *corpus*. Tworzenie modelu odbywa się z uwzględnieniem znaków specjalnych oznaczających początek i koniec zdania. Podczas jego konstrukcji następuje przetworzenie korpusu metodą *process\_text*, która m.in. usuwa znaki specjalne, duże litery, wielokrotne spacje.

Za generowanie wypowiedzi odpowiada metoda **generate** pozwalająca generować sekwencje zdań zadanej długości. Metoda zaczyna od losowego słowa z listy słów pojawiających się na początku zdań. Każde zdanie generowane jest oddzielnie do momentu aż zostanie wylosowany znacznik końca zdania, lub zostanie przekroczony parametr ograniczający długość zdania.

Konstrukcja modelu 1 gramowego na lokalnym komputerze trwa ok. 4 minuty a 2 gramowego 6 minut .

Poniżej przykładowe wyniki.

Przykładowe przemówienie dla modelu 1-gramowego wygenerowane z parametrami (steps=10, max\_sent=10):

'Panie marszałku. Poza tym pracownikom projekt ordynacji podatkowej dla polski 1. Nie przekreślamy sprawę jasno sformułowane są niezgodne z dwiema. Jak zostanie zwrócony jako szef zespołu na wykreśleniu pkt. Boję się o swoich list to zasłużone dla strażaków. Chciałbym skupić wyłącznie do spożycia alkoholu oraz refundacji. Wysoka izbo. Jakie gwarancje zastawnika niż pobierających świadczenia z góry rozdzielanie. Swoich dzieci. W praktyce zostało wyartykułowane.'

Przykładowe przemówienie dla modelu 1-gramowego wygenerowane z parametrami (steps=10, max\_sent=20):

'Natomiast takich uchwał dotyczących kwestii warunków prawnych w budowie. Szanowny panie marszałku. Pierwsza. W rozporządzeniu rady sądownictwa dopiero na swoich kompetencji i. To kolejny projekt i szczególnej państwowej służby ochrony państwa. W takiej potrzeby zwiększania środków europejskich demokratów formułuje wnioski. Szanowni państwo dzisiaj o szczególne znaczenie specjalnych jest wierną. Powiedzieliśmy do spółki na stworzenie nowej rezerwy zostanie skazany. Panie marszałku. Manipulowanie tą wersją projektu skłania do kilkuletnich zaniedbań i. Pierwszy pracę. Dopiero w związku z przeznaczeniem a przede wszystkim informacyjne. To proszę państwa dla poszczególnych powiatów oraz zniesienie ograniczeń. Przypuszczam że każda ze strony młodzieży oraz upoważnienie na. Później próbuje ogarnąć tej sprawie wyjścia naprzeciw wspomnę np. Można było. Potrzeba uchwalenia zaproponowanego przez radę ministrów. Jej do pana ministra tchórzewskiego byłego likwidatora majątku zagrożonego. Wprowadza tę izbę nie udało mi wiadomo że albo. Wysoki sejmie.'

Łatwo jest ocenić, że powyższe wypowiedzi nie pochodzą z prawdziwych wystąpień sejmowych. Możliwych kombinacji słów następujących po sobie jest zbyt dużo aby taki model mógł zachować logikę pomiędzy kolejnymi elementami zdania.

Przykładowe przemówienie dla modelu 2-gramowego wygenerowane z parametrami (steps=10, max\_sent=10):

'Mam kłopot z budowaniem nowych mieszkań w złym świetle respektowanie. To podchodzi pod administrację. Chodzi zapewne o wiele trudniejsza jest sytuacja w państwie członkowskim. Jeżeli obie strony uznały za obowiązujące zajmowanie przez 10 lat. Panowie osiągnęli a my słyszymy sto takich zgromadzeń. Jeżeli uzyskaliśmy technologie na warunkach partnerskich; wtedy jest ona nam. Do orzekania... Tyle jesteś samorządny o tyle obecnie średnio 133 osoby na. Przyjęcie kompromisowych rozwiązań łączących projekt rządowy nad którym dyskutujemy to. Jak kupuję jakiś towar.'

Przykładowe przemówienie dla modelu 2-gramowego wygenerowane z parametrami (steps=10, max\_sent=20):

'Chciałbym wykreślić ze sprawozdania... Powoli zanika a mogła to być może wykorzystawszy także niezły poziom techniczno-technologiczny jest znacznie lepszy niż ten który złożył przysięgę. Jednak prenumerować przez pocztę przez swoich i dlatego nie ma powodu do jakichkolwiek sytuacji które mogą naruszać wolności wykonywania zawodu. Czy zrewaloryzowana ale też o rozruchu. Nie ustosunkował.właśnie do tego że komitet badań naukowych lub organizacji społecznej o czym mówił pan prezes socha był łaskaw pan. To przekształcane w pracowniczy program emerytalny jest instytucją samofinansującą się która jest w pełni popiera i w imieniu posłów z. Będziemy kontrolować waszą działalność trybunał stanu który decyzją sejmu termin ten wydaje się tu toczy będzie miała zastosowanie w postępowaniu. Nie ochronę uprawnień najemców i dzierżawców które już dzisiaj plagi społecznej. Mówię wewnętrzna organizacja tego ratownictwa także. Gdyby obecne napięcia między przedstawicielami ministerstwa skarbu czy po przyjęciu będzie miała jedno pytanie: co pani poruszyła są prawnie chronione.'

Widać, że w drugim modelu tekst bardziej przypomina realnie wystąpienia i prawdziwe zdania. Oczywiście w dalszym ciągu nie ma żadnych wątpliwości, że nie są to realne wypowiedzi, ale ich czytanie jest nieco przyjemniejsze niż dla poprzedniego modelu.

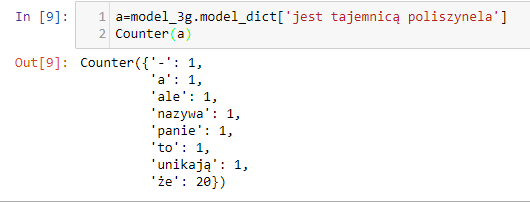
Podjęto również próbę zbudowania modelu 3-gramowego. Czas kalkulacji wydłużył się istotnie. Budowa słownika do modelu trwała prawie 30 minut.

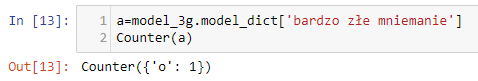
Przykładowy tekst wygenerowany z parametrami (steps=10, max\_sent=10):

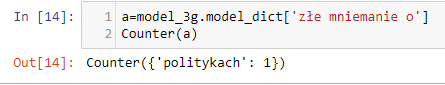
'Czy elementem realizacji polityki gospodarczej. Już zadała to pytanie bo jest tajemnicą poliszynela unikają one płacenia miliardowych podatków i nadmiernie wykorzystują pracowników. I marek belka nie budzi naszego zaufania. Dziękuję unii wolności za to że ma wygrać najlepszy. Chciałabym przytaczać w tej chwili problem bezrobocia mimo że ten dokument został odtajniony decyzją wiceministra spraw wewnętrznych i administracji i innym. Wiadomo są jedne z głównych czynników utrudniających udany proces integracji repatriantów z polskim społeczeństwem z obywatelami żeby przez zaniechania rządu nie. Przecież trudno się spodziewać tego że przy wyważeniu argumentów ograniczenie czasu zgłaszania weta też ma pewne koncepcje które należałoby wysokiej izbie. Połowa to mogą być grupy członkowskie. Wtedy ta nowelizacja rzeczywiście mogła wejść w życie na drugim etapie jego nowelizacji. Panie prokuratorze zwrócić uwagę że istnieje bardzo złe mniemanie o politykach i o polityce wobec cudzoziemców polityce imigracyjnej jest takim dzwonem.'

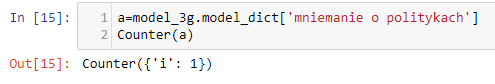
Zdania sprawiają wrażenie lepiej skonstruowanych niż dla poprzednich modeli. Jednak stosunkowo mały zbiór danych uczących sprawia, że model w dużym stopniu powtarza całe zdania ze zbioru uczącego.

Przyjrzyjmy się jak wygląda słownik dla niektórych 3-gramów występujący w powyższym tekście.









Z powyższego wynika, że do budowy dobrego modelu dla większych wartości n dla języka polskiego potrzebne są dużo większe zestawy danych. Dla języków z prostszymi gramatykami, gdzie nie występuje tak wiele odmian słów, wymagania te mogą być niższe.

# Badanie prawdopodobieństwa wystąpienia danego zdania

Mamy również możliwość przeanalizowania spójności danej sekwencji z modelem. Tj. jak prawdopodobne jest, że dane zdanie pochodzi z danego modelu. Można to szacować za pomocą wartości perplexity, która generalnie służy do oceny jakości modelu n-gramowego na danych testowych. Dla pojedynczego zdania możemy wyliczać:

P=*p*(*wtest*)−1*N*, where *p*(*wtest*)=∏*i*=1*N*+1*p*(*wi*|*wi*−1*i*−*n*+1)

Gdzie N jest długością zdania testowanego. Liczniki i indeksowanie są tak dopasowane aby uwzględniać pozycję początkową i końcową wyrazu w zdaniu poprzez uwzględnianiu w analizach dodatkowych tokenów oznaczających początek i koniec zdania.

Aby zaadresować kwestię zerowania się iloczynu w przypadku pojawienia się n-gramów spoza korpusu, na którym trenowano model, wprowadza się różne metody wygładzania tej miary. Tutaj zastosujemy „add-1 smoothing”, która modyfikuje kalkulację pojedynczych prawdopodobieństw w następujący sposób:

*p*(*wi*​∣*wi*−*n*+1*i*−1​)= *c*(*wi*−*n*+1*i*​)+1​ / *c*(*wi*−*n*+1*i*−1​)+*V*

Za kalkulację tej wielkości odpowiada metoda **perplexity** zaimplementowana dla klasy *ng\_models*.

Przykładowo dla modelu 2-gramowego z poprzedniego punktu otrzymujemy dla zdania ‘'Dzisiejsze posiedzenie ma na celu omówienie projektu ustawy'’ wartość 109921 a dla zdania 'Lwy to zwierzęta kotowate żyjące na sawannie' 4240107.

# Modele per osoba

# Badanie prawdopodobieństwa danego zdania w zależności od autora wystąpienia

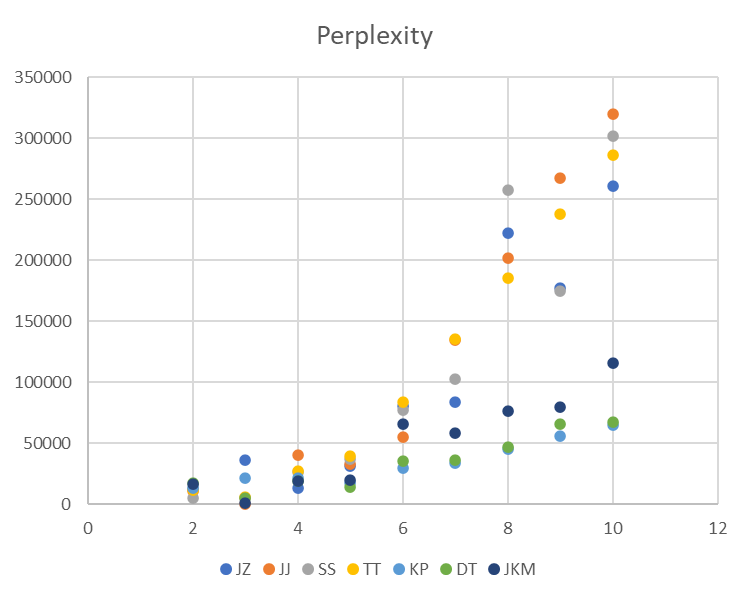
Metodę opisaną w powyższym punkcie wykorzystano do weryfikacji jakości modeli jakie można zbudować dla konkretnego autora. Ponieważ dzieląc zbiór uczący na poszczególnych autorów, zasób danych staje się mocno ograniczony, analizy te wykonywano dla modeli 1-gramowych.

Zbudowano oddzielne modele dla 5 polityków z największą liczbą słów w danych (400 – 500 tys.) oraz dla trzech z istotnie mniejszą ilością słów (<100 tys.). Porównano wyniki dla kilku mniej lub bardziej prawdopodobnych zdań.



Na szaro zaznaczone są osoby, ze stosunkowo dużym zbiorem danych uczących. Widać, że ma to istotny wpływ w sytuacjach gdy oceniane są zdania „mało prawdopodobne”. Wówczas w kalkulacji prawdopodobieństwa ciągu wyrazów pojawiają się bardzo małe wartości, co z kolei przekłada się na wysokie perplexity. Ogółem prawdopodobieństwa mają podobne trendy dla wszystkich autorów, ale warto zauważyć zaburzenie kolejności pomiędzy zdaniami „Demokracja jest najlepszym systemem politycznym” a „Zupełnie przypadkowy układ pięciu słów”. Niemniej cieszy, że sekwencja „lew czarownica i stara szafa” w każdym przypadku są opcją najmniej prawdopodobną.

Poniższy wykres pokazuje rozrzut wartości zależny od modelu, który pokazuje, że przy tak małych zbiorach uczących, trudno jest porównywać wartości pomiędzy modelami.



Niemniej przeglądając słowniki dla poszczególnych modeli budowanych w ten sposób, widać, że cechuje je duża przypadkowość. Zbyt wiele n-gramów nie będzie się w nich odnajdywać i będą kontrybuowały do miary z tą samą, najniższą wartością. Zdecydowanie też miara ta nie nadaje się do porównywania modeli budowanych na korpusach o istotnie różnej liczności.

# Budowa wypowiedzi w oparciu o maksymalizację prawdopodobieństwa

Poniższe analizy wykonano na tekście oczyszczonym z najczęściej pojawiających się, mało informacyjnych fraz typu „Panie Marszałku!”.

Znając prawdopodobieństwa następowania kolejnych słów po sobie, można spróbować zbudować zdanie minimalizujące wartość perplexity, tj. maksymalizujące prawdopodobieństwo „zobaczenia go w modelu”. W tym celu rozszerzono klasę *ng\_models* o metodę *mprob\_sent*. Celem tego ćwiczenia była możliwość otrzymania zdania, które byłoby najbardziej charakterystyczne dla danego korpusu, czyli na przykład można by je określić jako dewizę związaną z danym tematem, osobą czy też grupą.

Metoda zakłada pobranie pierwszego słowa zdania od użytkownika a następnie wybieraniu kolejnych słów wg prawdopodobieństw w modelu. Czyli jeśli słowem początkowym jest słowo „poseł”, metoda sprawdzi jakie n-gramy pojawiały się w modelu zaczynające się od słowa poseł i wybierze najczęstszy. Przykładowe najczęstsze następniki słowa „poseł” w modelu 1-gramowym dla wybranego mówcy.

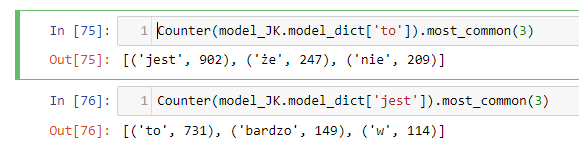


Niestety okazuje się, że taki algorytm bardzo łatwo wpada w pętle. Przykładowe zdania otrzymane w ten sposób:

'Poseł sprawozdawca mówił o tym że w tym że w tym że w tym że w tym że w tym że w tym że w tym że w tym że w tym.'

'Ja to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to jest to.’

Powód jest zobrazowany na poniższym obrazku.



W takim przypadku, jeśli model trafi w pewnym momencie na słowo „to”, zapętli się i będzie powtarzał 2-gram „to jest”.

Jednym z możliwych rozwiązań jest zakaz powtarzania słów w obrębie jednego zdania. W przypadku trafienia na słowo, które już się pojawiło w trakcie generowania, wybieramy kolejne wg częstości pojawiania się. Kończymy generowanie, gdy sytuacja powtarza się zbyt często. Wówczas otrzymujemy np.:

'Poseł sprawozdawca mówił o tym że w tej ustawy.'

'Ja to jest bardzo ważne.'

W przypadku, gdy do metody nie zostanie przekazane pierwsze słowo, generowanie rozpocznie się od słowa, które najczęściej pojawia się jako pierwsze w zdaniach w danym korpusie. Dla modeli budowanych dla pojedynczych mówców są ta zdania m.in.:



Długość otrzymanych zdań jest bezpośrednią pochodną skłonności do zapętlania się słów w danym korpusie. Widać, że dla większości najczęstszym słowem jest „w”.

Jako alternatywną metodę, która mogłaby wpłynąć na „urozmaicenie” generowanych zdać, rozważano usunięcie z korpusu tzw. Stop-words, czyli nieinformacyjnych słów, które mają wysoką częstotliwość występowania w zdaniu. To jednak mogłoby generować zdania nienaturalne w swojej gramatyce oraz

# Modele dla partii

Ponieważ modele budowane dla pojedynczych osób wydawały się budowane na zbyt małych korpusach, powtórzono analizy grupując autorów wg najnowszego przypisania do partii (SQL). Liczności słów w korpusie dla głównych partii wyglądają następująco:

|  |  |  |
| --- | --- | --- |
| **partia** | **liczba\_sl\_raw** | **date\_** |
| Prawo i Sprawiedliwość | 12 409 825 | 31 630 |
| Sojusz Lewicy Demokratycznej | 13 143 036 | 27 129 |
| Platforma Obywatelska | 9 438 332 | 24 977 |
| Polskie Stronnictwo Ludowe | 8 256 113 | 17 851 |
| posłowie niezrzeszeni | 3 735 227 | 7 358 |
| Akcja Wyborcza Solidarność | 3 304 864 | 5 642 |
| Samoobrona | 2 027 951 | 4 407 |

Wartości są istotnie wyższe, oczekiwane się więc ciekawsze wyniki. W pierwszej kolejności sprawdzimy jaki jest czas generowania modeli z zależności od stopnia złożenia i wygenerujemy wyniki.

Modele 1-gramowe (7 modeli) wygenerowały się w nieco ponad minutę, 2 – gramowe minutę dłużej.

# Generowanie przemówień dla partii

W ramach ciekawostki poniżej dwa przemówienia wygenerowane dla dwóch różnych partii z modeli 2-gramowych.

'Proponowane przez rząd projekt ustawy był przygotowany przez poprzedni rząd. Dobrze ponad 50% mandatów w sejmie był forsowany przede wszystkim. Chciałbym przeprosić pana ministra o wyjaśnienie waszej strategii politycznej bo. Dzisiaj ustosunkować do apeli osób głuchoniemych dotyczących sztucznego systemu językowo-migowego. Natomiast wybory samorządowe dlatego też możliwość przedłużenia funkcjonowania podatku importowego. Z playbacku. Nie utrudnią nam szukania tego co wiadomo z którym zwracały. Nie zgodnie z art. Panie piotrowicz do sytuacji w praktyce w tym sejmie na. Nie zabiegać.'

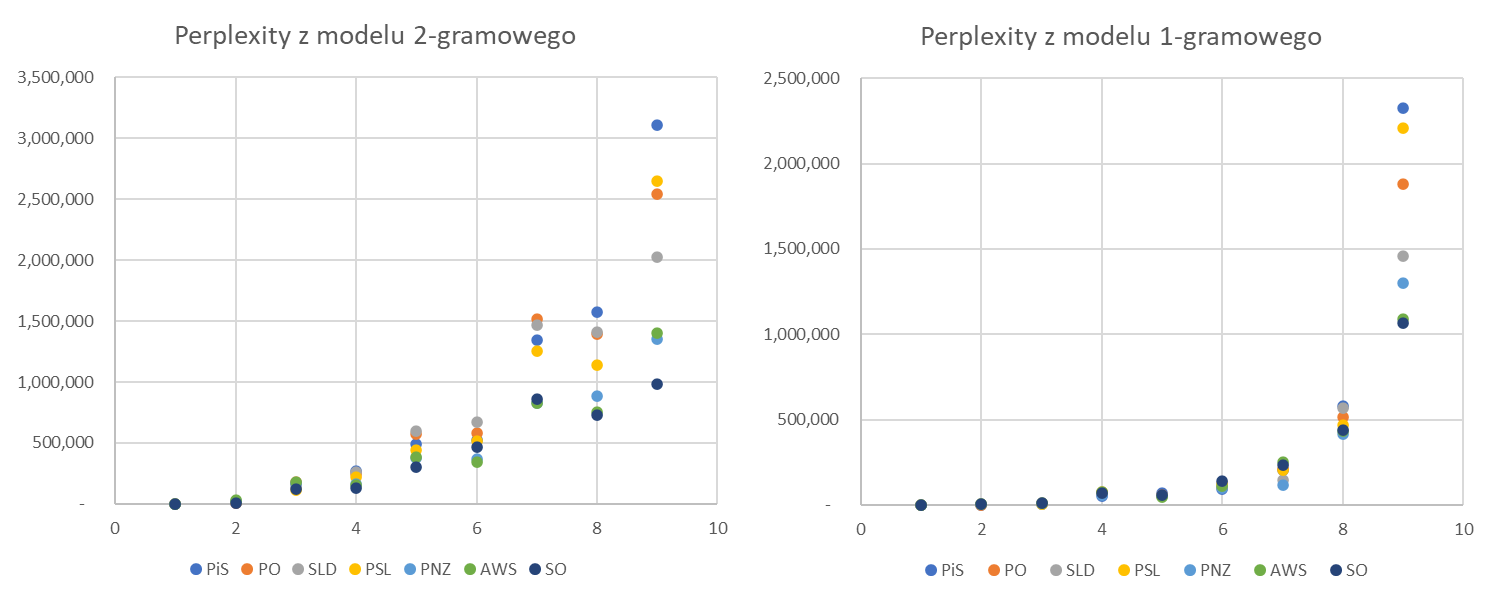
'Prezydent złamał konstytucję a składał przysięgę że będzie też źle. Po najniższych cenach natomiast ten kto dysponuje mediami obok zarabiania. Wywiadów środowiskowych. Nieszczęściem w postaci podatków zapłacą wasze dzieci i walczyć o. Wysoki koszt kredytu inwestycyjnego zaciągniętego przez kasy chorych musi mu. W posiedzeniu komisji gospodarki o rządowym projekcie ustawy o zmianie. Zdziwienie i twierdzi że możliwa tu będzie. I nagłaśniane przez media od lat popieramy wszystkie racjonalne inicjatywy. Można wyrobić zysk z pracy. Wysoka cena będą odpowiedzialne za dopuszczenie do obciążenia podatkowymi kosztami.'

# Badanie prawdopodobieństwa danego zdania w zależności od partii

Modele dla partii budowane są na zdecydowanie większych korpusach, co pozwala oczekiwać, że zwracane wyniki będą bardziej stabilne. Sprawdzono wartości perplexity dla tych samych zdań. Dla modeli 1-gramowych otrzymano następujące wyniki.



Porównano następnie wyniki dla modeli 1 i 2 gramowych.



Można zauważyć, że dla modelu 1-gramowego wartości perplexity są dużo niższe co wynika m.in. z tego, że łatwiej trafić na wyrażenie dwuwyrazowe, które się w modelu pojawia niż 3 wyrazowe. Na pewno widać większą stabilność wyników. Porównywanie konkretnych wartości pomiędzy modelami zdecydowanie ma więcej sensu.

# Najbardziej prawdopodobne zdanie dla partii

Niestety generowanie najbardziej prawdopodobnego zdania z modeli dla partii przyniosło rozczarowujące rezultaty. Okazuje się, że dla tego zadania zwiększony rozmiar korpusu powoduje, że modele 1-gramowe stają się zbyt podobne. Generowane zdania były prawie identyczne.



Dla modeli 2-gramowych zdania są nieco bardziej zróżnicowane, ponieważ im dłuższe są n-gramy, na których opiera się model, tym mniejsza szansa, że będą się powtarzać w różnych korpusach równie często. Dalej jednak są dość podobne.



# Dalszy preprocessing danych

Do dalszej pracy z danymi wykorzystano pakiet pythonowy stanza [1] ((<https://stanfordnlp.github.io/stanza/index.html#about>). Jest to pakiet, który może być wykorzystywany m.in. do tokenizacji tekstu, lematyzacji słów, określania części mowy oraz zadań z obszaru NER (*Named Entity Recognition*). Modele w pakiecie obejmują ponad 70 języków, w tym polski. Bazują na projekcie Universal Dependencies [2], którego celem jest spójny opis gramatyk dla różnych języków, który jest rozwijany w celu trenowania rozwiązań dla wielu języków jednocześnie.

W niniejszej pracy wykorzystano modele oparte o korpus LFG, który zawiera 17 246 zdań i 130 967 tokenów. Zawiera oznaczenia 15 części mowy: ADJ, ADP, ADV, AUX, CCONJ, DET, INTJ, NOUN, NUM, PART, PRON, PROPN, PUNCT, SCONJ, VERB.

Przykładowe wywołanie dla zdania „Pisanie pracy zaliczeniowej to bardzo trudny proces”:

{

"id": "1",

"text": "Pisanie",

"lemma": "pisać",

"upos": "NOUN",

"xpos": "ger:sg:nom:n:imperf:aff",

"feats": "Aspect=Imp|Case=Nom|Gender=Neut|Number=Sing|Polarity=Pos|VerbForm=Vnoun",

"head": 0,

"deprel": "root",

"misc": "start\_char=0|end\_char=7"

},

{

"id": "2",

"text": "pracy",

"lemma": "praca",

"upos": "NOUN",

"xpos": "subst:sg:gen:f",

"feats": "Case=Gen|Gender=Fem|Number=Sing",

"head": 1,

"deprel": "nmod",

"misc": "start\_char=8|end\_char=13"

},

{

"id": "3",

"text": "zaliczeniowej",

"lemma": "zaliczeniowy",

"upos": "ADJ",

"xpos": "adj:sg:gen:f:pos",

"feats": "Case=Gen|Degree=Pos|Gender=Fem|Number=Sing",

"head": 2,

"deprel": "amod",

"misc": "start\_char=14|end\_char=27"

},

{

"id": "4",

"text": "to",

"lemma": "to",

"upos": "AUX",

"xpos": "pred",

"feats": "Mood=Ind|Tense=Pres|VerbForm=Fin|VerbType=Quasi",

"head": 1,

"deprel": "cop",

"misc": "start\_char=28|end\_char=30"

},

{

"id": "5",

"text": "bardzo",

"lemma": "bardzo",

"upos": "ADV",

"xpos": "adv:pos",

"feats": "Degree=Pos",

"head": 6,

"deprel": "advmod",

"misc": "start\_char=31|end\_char=37"

},

{

"id": "6",

"text": "trudny",

"lemma": "trudny",

"upos": "ADJ",

"xpos": "adj:sg:nom:m3:pos",

"feats": "Case=Nom|Degree=Pos|Gender=Masc|Number=Sing|SubGender=Masc3",

"head": 7,

"deprel": "amod",

"misc": "start\_char=38|end\_char=44"

},

{

"id": "7",

"text": "proces",

"lemma": "proces",

"upos": "NOUN",

"xpos": "subst:sg:nom:m3",

"feats": "Case=Nom|Gender=Masc|Number=Sing|SubGender=Masc3",

"head": 1,

"deprel": "nsubj",

"misc": "start\_char=45|end\_char=51"

},

{

"id": "8",

"text": ".",

"lemma": ".",

"upos": "PUNCT",

"xpos": "interp",

"feats": "PunctType=Peri",

"head": 1,

"deprel": "punct",

"misc": "start\_char=51|end\_char=52"

}

Bazując na powyższym pakiecie oraz liście polskich *stopwords*, przygotowano nową wersję tesktów przemówień, która:

- nie zawiera znaków interpunkcyjnych,

- składa się z lematów słów

- nie zawiera słów z listy stopwords[[1]](#footnote-1).

# Modelowanie tematyczne

Modelowanie tematyczne (ang. \textit{topic modeling}) jest dziedziną, która skupia się na wykrywaniu tematów w zbiorze dokumentów. Bazuje na stowarzyszeniu określonych słów z tymi tematami. To zagadnienie może występować jako przykład uczenia nadzorowanego. Będzie tak w przypadku, gdy posiadamy zestaw dokumentów z przypisanymi do nich tematami. Wówczas staramy się zbudować model, który nauczy się rozpoznawać te tematy w nowych dokumentach. Z uczeniem nienadzorowanym mamy do czynienia w przypadku, gdy dopiero chcemy odkryć strukturę danego zestawu dokumentów. Nie wiemy jakie tematy mogą się tam pojawiać i stosujemy różne narzędzia statystyczne aby je zidentyfikować.

Jednym z podejść, które można zastosować w takiej analizie jest schemat \textit{tf-idf} (od ang. \textit{term frequency – inverse document frequency}). Polega on na redukcji wymiarowości zbioru danych poprzez konstrukcję macierzy, w której dla każdego dokumentu w zbiorze przypisujemy wektor bazujący na częstościach występowania w nim określonych słów. Wartość TF-IDF oblicza się ze wzoru:

{\displaystyle \mathrm {(tf{\text{-}}idf)\_{i,j}} =\mathrm {tf\_{i,j}} \times \mathrm {idf\_{i}} ,}{\displaystyle \mathrm {(tf{\text{-}}idf)\_{i,j}} =\mathrm {tf\_{i,j}} \times \mathrm {idf\_{i}} ,}

gdzie {\displaystyle tf\_{i,j}}tf\_{i, j} to tzw. „term frequency”, wyrażana wzorem:

{\displaystyle \mathrm {tf\_{i,j}} ={\frac {n\_{i,j}}{\sum \_{k}n\_{k,j}}},}{\displaystyle \mathrm {tf\_{i,j}} ={\frac {n\_{i,j}}{\sum \_{k}n\_{k,j}}},}

gdzie: {\displaystyle n\_{i,j}}n\_{i,j} jest liczbą wystąpień słowa {\displaystyle (t\_{i})}{\displaystyle (t\_{i})} w dokumencie {\displaystyle d\_{j},}{\displaystyle d\_{j},} a mianownik jest sumą liczby wystąpień wszystkich słów w dokumencie {\displaystyle d\_{j}.}{\displaystyle d\_{j}.} {\displaystyle idf\_{i}}{\displaystyle idf\_{i}} to „inverse document frequency” wyrażana wzorem:

{\displaystyle \mathrm {idf\_{i}} =\log {\frac {|D|}{|\{d:t\_{i}\in d\}|}},}{\displaystyle \mathrm {idf\_{i}} =\log {\frac {|D|}{|\{d:t\_{i}\in d\}|}},}

gdzie:

{\displaystyle |D|}{\displaystyle |D|} – liczba dokumentów w korpusie,

{\displaystyle |\{d:t\_{i}\in d\}|}{\displaystyle |\{d:t\_{i}\in d\}|} – liczba dokumentów zawierających przynajmniej jedno wystąpienie danego termu.

Otrzymujemy w ten sposób wektory równych długości, które można wykorzystać do analizy podobieństwa dokumentów. Algorytm ten jest stosowany m.in. w wyszukiwarkach internetowych oraz systemach antyplagiatowych. Aby zaadresować potrzebę jeszcze większej redukcji wymiarów, pojawiły się inne metody opisywania dokumentów. Jedną z popularniejszych jest latent Dirichlet allocatioc (LDA) przedstawiona w \cite{LDA}. Zakłada, że każdy dokument jest mieszanką różnych ukrytych (\textit{latent}) tematów, a każdy z tematów opisuje rozkład względem słów korpusu.

# Latent Dirichlet Allocation

Do analiz związanych z analizą tematu wykorzystano pythonową bibliotekę gensim.

Aby uzyskać listę tematów obecnych w badanych dokumentach w pierwszej kolejności należy przygotować słownik zawierający wszystkie słowa z badanego korpusu. Tą operację wykorzystujemy na maksymalnie oczyszczonych danych. Funkcja \textit{ gensim.corpora.Dictionary} buduje słownik mapowań konkretnych słów na ich liczbowe odpowiedniki oraz informację o tym ile razy dane słowo występuję w dokumentach i ile dokumentów je zawiera. Opierając się o te parametry, dobrze jest przefiltrować słownik. Z jednej strony należy usunąć słowa pojawiające się zbyt rzadko, ponieważ nie będą one w stanie określić tematu pojawiającego się w wielu dokumentach, a z drugiej strony trzeba również usunąć te, które pojawiają się zbyt często, aby nie zaburzały konstrukcji tematów.

Słownik zbudowany na naszym korpusie zawiera przed oczyszczeniem 152 755 elementów. Po zastosowaniu domyślnych parametrów eliminacji słów pozostało 52 425. Domyślne parametru usuwają słowo jeśli pojawia się w mniej niż 5 dokumentach lub w więcej niż połowie z nich.

Dla tak przygotowanego słownika możemy zamienić dokumenty na format \textit{bag of words} (BoW), to jest zbiór par $(id, licznik\_id)$ wszystkich słów jakie się w nim pojawiają. Przykładowo da wybranego dokumentu, z tego formatu można uzyskać następującą informację:

\textit{Word 11 ("czas") appears 2 time.\\

Word 16 ("decyzja") appears 2 time.\\

Word 20 ("dotyczyć") appears 4 time.\\

Word 22 ("duży") appears 1 time.\\

Word 24 ("długo") appears 2 time.\\

Word 30 ("głos") appears 1 time.\\

Word 35 ("intencja") appears 1 time.\\

Word 38 ("istotny") appears 1 time.\\

Word 40 ("izba") appears 4 time.\\

Word 43 ("kierować") appears 2 time.\\

Word 45 ("komisja") appears 2 time.\\

Word 59 ("marszałek") appears 1 time.\\

Word 60 ("minister") appears 8 time.\\

Word 62 ("musieć") appears 2 time.\\

Word 64 ("myśleć") appears 1 time.\\

Word 65 ("mówić") appears 6 time.}

Następnie możemy otrzymać model \textit{tf-idf}, który wyznaczy macierz tych wartości. Zauważmy, że informacje z BoW agregują słowa wyłącznie na poziomie lokalnym, a TF-IDF dodaje do tego wymiar globalny. Sprawdźmy czy modele LDA budowane na tych dwóch zestawach danych będą się od siebie istotnie różnić. Wykorzystamy do tego funkcję \textit{gensim.models.LdaMulticore}. Wywołując tę funkcję możemy określić parametry $num\_topics$ czyli liczbę tematów jakie chcemy określić w naszym korpusie. Domyślna wartość to 100, co może się wydawać dużą wartością.

Po wywołaniu funkcji z parametrami $ num\\_topics=100, passes=1$ otrzymujemy wyniki pokazane na obrazku \ref{tm01}. Wizualizacja prezentuje 10 słów najistotniejszych w określaniu danego tematu. Niektóre tematy wydają się bardzo precyzyjnie określone np. Topic 0 dotyczy wymiaru sprawiedliwości, Topic 1 podatków, Topic 8 to służba zdrowia a Topic 90 transport. Pojawiają się też tematy „mieszane”, którym trudno opisać np. Topic 2 czy Topic 5. W definicji tematów pojawiają się takie słowa jak „poseł”, „minister”, „polski”, które wydają zbyt pospolite aby dobrze określać poszczególne tematy. Statystyki dla tych słów wyglądają następująco:\\

Słowo 128 - poseł pojawia się w korpusie 182857 razy i występuje w 66968 dokumentach (42.2%).\\

Słowo 60 - minister pojawia się w korpusie 236987 razy i występuje w 77965 dokumentach (49.1%).\\

Słowo 123 - polski pojawia się w korpusie 245334 razy i występuje w 73013 dokumentach (46.0%).\\

Wykonano ponownie proces oczyszczania słownika ze zmianą parametru $no\\_above$ na 40%. Zmieniło to wielkość słownika jedynie o 12 elementów. Zobaczmy czy wpłynęło to na generowane tematy. Wyniki zaprezentowane na rysunku \ref{tm02} sprawiają jednak wrażenie bardziej chaotycznych niż dla poprzedniego zestawu parametrów.

Wyniki testowano dla kilku zestawów parametrów filtrowania słownika oraz liczby tematów i oceniano subiektywnie pod kątem spójności tematów. Wydaje się, że 20 tematów to zbyt mało aby pokryć pełne spektrum przemówień polskich polityków. Zanim

- wizualizacji

- rozkład tematów

- tf idf

- Optymalizacja liczby tematów

# Analiza sentymentu

# Bibliografia

[1] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton and Christopher D. Manning. 2020. [Stanza: A Python Natural Language Processing Toolkit for Many Human Languages.](https://arxiv.org/abs/2003.07082) In Association for Computational Linguistics (ACL) System Demonstrations. 2020. [[pdf](https://arxiv.org/pdf/2003.07082.pdf)]

[2] <https://universaldependencies.org/>

Stanza is licensed under the [Apache License, Version 2.0](https://www.apache.org/licenses/LICENSE-2.0) (the “License”), you may not use the software package except in compliance with the License.

1. Listę stopwords pozyskano z repozytorium <https://github.com/bieli/stopwords> [↑](#footnote-ref-1)