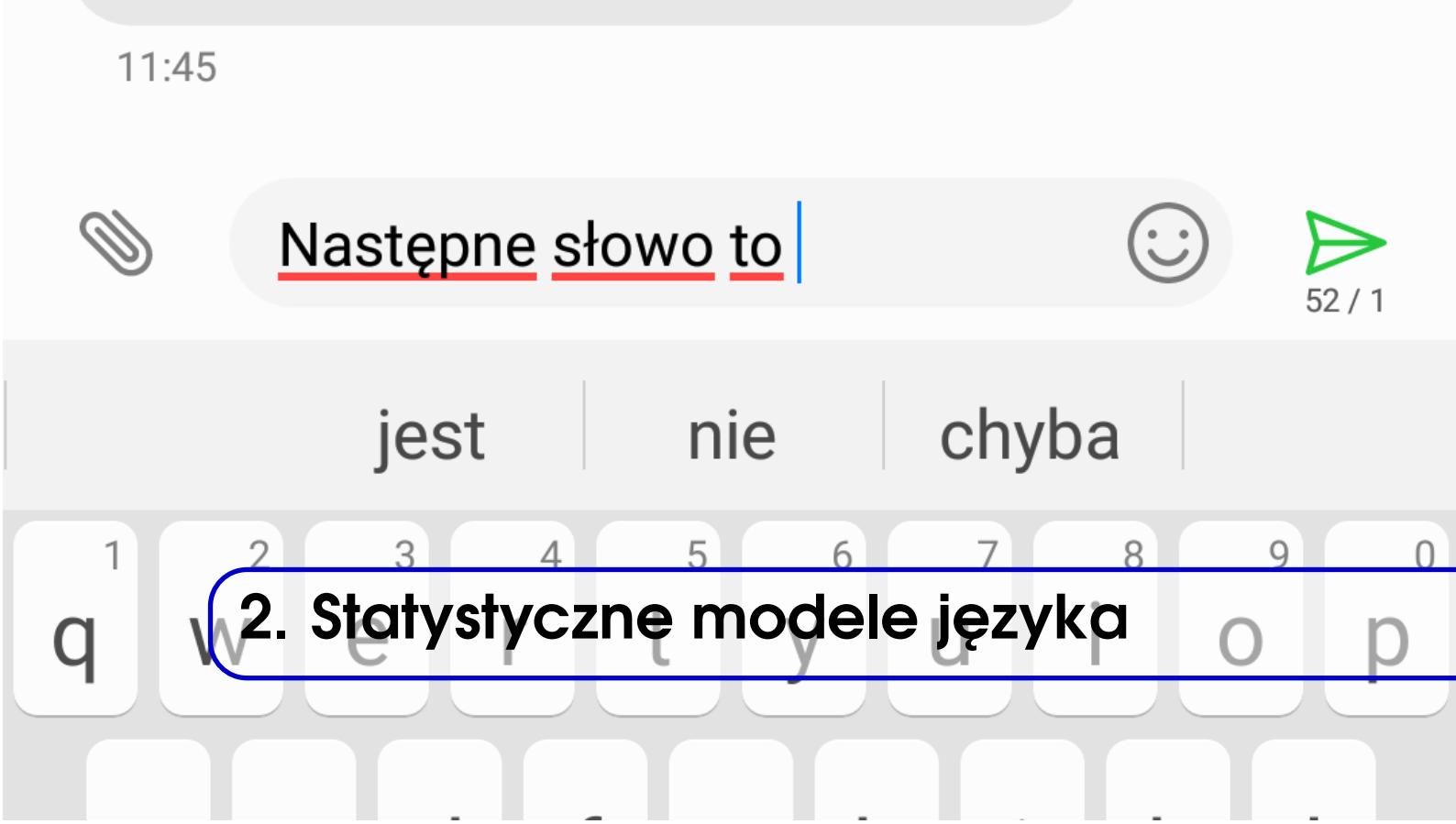


2. Statystyczne modele języka

Na ostatnich zajęciach poznaliśmy rodzinę modeli *n*-gramowych ze szczególnym naciskiem na modele trzy-gramowe. Choć modele *n*-gramowe są bardzo proste, w praktyce – dzięki wykorzystaniu pewnych dodatkowych trików – działają zaskakująco dobrze i do czasu „głębokiej" rewolucji działały lepiej niż podejścia neuronowe. No właśnie, co to znaczy że model języka „działa lepiej"? W jaki sposób możemy zmierzyć jakość jego działania? Na czym polegają główne problemy modeli *n*-gramowych i w jaki sposób można próbować je rozwiązać przy wykorzystaniu wspomnianych „trików"? O tym będzie dzisiaj ;)

2.1 Ewaluacja modeli języka

Metody oceny jakości działania modelu języka można podzielić na dwie główne rodziny: metody zewnętrzne i wewnętrzne. Metody zewnętrzne (ang. *extrinsic evaluation*) oceniają model języka pod kątem działania na konkretnym zadaniu. Jak wspominaliśmy, modelowanie języka nie jest zwykle celem samym w sobie, ale częścią działania większego systemu np. systemu rozpoznawania mowy, tłumaczenia maszynowego czy nawet klasyfikacji tekstu. Możliwe jest więc wykorzystanie modelu języka jako część takiego systemu i sprawdzić jego przydatność w rozwiązywaniu konkretnego zadania. W szczególności, możemy zmierzyć zmianę specyficznej dla zadania miary jakości systemu (np. dla rozpoznawania mowy: liczba błędnie zapisanych słów, dla klasyfikatora: trafność) w zależności od tego jakiego modelu języka użyjemy. Jest to szczególnie ważne, gdyż nawet rozważając tylko modele *n*-gramowe wytrenowane na tym samym korpusie, może okazać się że do jednego zadania lepiej działa model bigramowy, a do drugiego trzy- czy cztero-gramowy. Ewaluacja zewnętrzna jest więc najlepsza z praktycznego punktu widzenia, gdyż ocenia ona tzw. efekt końcowy – o ile wykorzystanie modelu języka poprawiło nam jakość działania systemu.

Jednakże, ewaluacja zewnętrzna ma wadę, która jest też jej największą zaletą: wymaga

całego systemu. W praktyce może to oznaczać, że aby zmierzyć jakość działania modelu języka potrzebny jest trening całego systemu tłumaczenia maszynowego czy rozpoznawania mowy i dopiero wtedy będzie wiadome jak dobry jest model języka. Podobnie, gdy system jest jeszcze w fazie rozwoju różne zmiany w jego konstrukcji mogą wpływać na ocenę modelu języka co utrudnia z kolei tworzenie lepszych modeli języka. Z tego powodu istnieją także metody wewnętrzne (ang. *intrinsic evaluation*), które wymagają podzielenia korpusu na część uczącą i testową, w którym testowa służy do obliczenia jakiejś miary jakości działania. Należy jednak podkreślić, że lepsza jakość modelu języka wg. pewnej miary niekoniecznie musi przekładać się na lepszą jakość działania całego systemu inżynierii lingwistycznej (w praktyce lepsza miara może czasami powodować gorsze działanie *całego* systemu!).

Estymacje prawdopodobieństw w modelach n-gramowych robiliśmy przy użyciu metody największej wiarygodności, więc naturalną miarą jakości modelu jest wartość logarytmicznej funkcji wiarygodności. Intuicyjnie, dobry model języka powinien przypisywać stosunkowo wysokie prawdopodobieństwa prawidłowym (rzeczywistym) zdaniom, a niepoprawne losowe sekwencje słów powinny mieć zerowe prawdopodobieństwa. De facto, przypisanie niezerowego prawdopodobieństwa nieprawidłowej sekwencji automatycznie musi zabierać prawdopodobieństwo z jakichś innych zdań. Rozkład prawdopodobieństwa powinien być więc jak najmocniej skoncentrowany na prawidłowych wypowiedziach, przypisując jak najmniej prawdopodobieństwa innym, niepoprawnym sekwencjom słów.

Źródłem prawidłowych zdań jest oczywiście korpus uczący, któremu w trakcie nauki model uczy się przypisywać jak najwyższe prawdopodobieństwa. Moglibyśmy więc mierzyć jakość modelu języka poprzez wartość prawdopodobieństwa którą przypisuje on całemu korpusowi (im więcej, tym lepiej). Łatwo jednak pokazać, że najlepszą jakość otrzymywałby model języka całkowicie przeuczony do korpusu tj. koncentrujący na nim cały rozkład i przypisujący zerowe prawdopodobieństwa do jakichkolwiek zdań spoza korpusu. Tej pułapki unikamy używając korpusu testowego czyli sprawdzając jak mocno rozkład jest skoncentrowany na niewidzianych wcześniej rzeczywistych zdaniach. Wartość logarytmicznej funkcji wiarygodności (ang. *log-likelihood*) dla zdań s w zbiorze testowym D_{test} obliczmy standardowym wzorem[1]

$$\ell_{test} = \sum_{s \in D_{test}} \log P(s)$$

Dla modelu trzy-gramowego wzór ten przyjmie następującą postać

$$\ell_{test} = \sum_{s \in D_{test}} \log \prod_{i=1}^{n_s+1} P\left(w_i^{(s)} | w_{i-1}^{(s)}, w_{i-2}^{(s)}\right) = \sum_{s \in D_{test}} \sum_{i=1}^{n_s+1} \log P\left(w_i^{(s)} | w_{i-1}^{(s)}, w_{i-2}^{(s)}\right)$$

gdzie n_s to długość zdania s, a $w_i^{(s)}$ to i-te słowo w zdaniu s. Nietrudno zauważyć, że obie sumy de facto iterują po kolei po wszystkich słowach w korpusie.

> ! Warto przemyśleć: zadanie modelowania języka jest zadaniem nienadzorowanym tj. nie wymaga etykiet czy anotowanych danych. Pomimo tego, wymagany jest podział na zbiór testowy i uczący, typowy dla uczenia nadzorowanego!

[1] Podane wzory zakładają, że zdania są niezależnymi obserwacjami lub że model przypisuje prawdopodobieństwa zdaniom (czyli prawdopodobieństwo kilku zdań jest wyznaczane poprzez mnożenie prawdopodobieństw modelu dla każdego ze zdań z osobna). Jeśli tak nie jest to uzyskamy wzór $\ell_{test} = \log P(D_{test})$ i każdy kolejny wzór powinien być odpowiednio zmieniony.

Używanie samej logarytmicznej funkcji wiarygodności ma swoje wady: nie uwzględnia długości korpusu, co można prosto rozwiązać poprzez wyciągnięcie średniej zamiast sumy. Uśrednioną po słowach wartość uzyskamy poprzez przemnożenie ℓ_{test} przez $\frac{1}{N}$ gdzie $N = \sum_{s \in D_{test}} (n_s + 1)$. Dodanie jedynki wynika z używania przez model tokenów $\boxed{\text{STOP}}$, jeśli model ich nie używa to $N = \sum_{s \in D_{test}} n_s$.

Ze względów historycznych i interpretacyjnych, najczęściej raportujemy miarę nieokreśloności (ang. *perplexity*) obliczaną poprzez zastosowanie funkcji eksponencjalnej do zanegowanej $\frac{1}{N}\ell_{test}$. Ostatecznie więc nieokreśloność jest wyrażona wzorem:

$$
\begin{aligned}
PP_{test} &= e^{-\frac{1}{N}\sum_{s \in D_{test}} \log P(s)} && \text{/mnożnik logarytmu to potęga jego argumentu/} \\
&= e^{\sum_{s \in D_{test}} \log P(s)^{-\frac{1}{N}}} && \text{/suma logarytmów to logarytm mnożenia/} \\
&= e^{\log \prod_{s \in D_{test}} P(s)^{-\frac{1}{N}}} && \text{/funkcja eksponencjalna i logarytm znoszą się/} \\
&= \prod_{s \in D_{test}} P(s)^{-\frac{1}{N}} \\
&= \sqrt[N]{\prod_{s \in D_{test}} P(s)^{-1}} = \sqrt[N]{\frac{1}{\prod_{s \in D_{test}} P(s)}}
\end{aligned}
\tag{2.1}
$$

gdzie $N = \sum_{s \in D_{test}} (n_s + 1)$. Zwróć uwagę, że w nowym wzorze prawdopodobieństwo jest w mianowniku ułamka, czyli im wyższe prawdopodobieństwo tym niższa nieokreśloność. Naszym celem jest więc konstruować modele języka z jak najniższą nieokreśloności (i wysokim prawdopodobieństwem).

Problem 2.1 Załóżmy, że używasz modelu trzy-gramowego w którym każdy warunkowy rozkład prawdopodobieństwa jest rozkładem jednorodnym. Ile wynosi nieokreśloność?

Ważną właściwością nieokreśloności PP_{train} jest fakt, że osiąga ona swoje maksimum dla modelu generującego słowa z rozkładu jednorodnego i jest ono równe wielkości słownika $|V| + 1$ (do którego wliczamy token $\boxed{\text{STOP}}$) . Prowadzi to do ciekawej interpretacji tej miary. Załóżmy, że model języka generuje kolejne słowa poprzez wybór listy słów, które wydają się odpowiednie, a potem wybiera całkowicie losowe słowo z listy (z takim samym prawdopodobieństwem). Średnia długość listy z której wybiera model za każdym razem to właśnie wartość nieokreśloności. Im te listy w kolejnych iteracjach są krótsze tym model języka jest pewniejszy co do słów, które generuje.

W praktyce model oczywiście nie generuje słów z zawężonej listy słów, a wszystkie wybory robi zgodnie z własnym modelem prawdopodobieństwa (raczej nie jest to rozkład jednorodny). Chodzi tutaj jednak o znalezienie punktu odniesienia do niepewności (entropii) rozkładu przy wyborze słowa. Miara nieokreśloności porównuje niepewność (entropię) rozkładu do rozkładu jednorodnego i wskazuje na ilu elementach trzeba by taki rozkład jednorodny zdefiniować, aby różnorodność obu rozkładów była taka sama. Inaczej: rzucając kostką z PP_{train} ściankami wygenerujesz sekwencję z taką samą entropią jak sekwencja z modelu języka.

W przypadku modeli języka dekomponujących prawdopodobieństwo sekwencji regułą łańcuchową, nieokreśloność jest też równa odwróconej średniej geometrycznej prawdopodobieństw kolejnych słów – co jest interpretacją samą w sobie. Na przykład, dla modelu

trzy-gramowego nieokreśloność można zapisać jako:

$$PP_{test} = \sqrt[N]{\frac{1}{\underbrace{\prod_{s \in D_{test}} \prod_{i=1}^{n_s+1}}_{N \text{ czynników}} P\left(w_i^{(s)} | w_{i-1}^{(s)}, w_{i-2}^{(s)}\right)}}$$

Jak zauważyłeś, interpretacje nieokreśloności nawiązują do liczby słów co przekłada się na pewną uwagę praktyczną. Wartości nieokreśloności można porównywać tylko pomiędzy modelami mającymi taki sam rozmiar słownika! Problem predykcji kolejnego słowa w zdaniu jest de facto problem klasyfikacji z $|V|+1$ klasami. Nie powinno się porównywać prawdopodobieństw uzyskanych z klasyfikatora 10-klasowego ($\frac{1}{10}$ to rozkład jednorodny) ze 100-klasowym ($\frac{1}{10}$ to może być już rozsądna pewność i prawidłowa klasyfikacja).

Aby dać pewną intuicję co do wartości nieokreśloności, podajmy ich typowe wartości dla typowego korpusu Penn Treebank dla języka angielskiego. Najlepsze modele 5-gramowe (z wygładzaniem Knesera-Neya – patrz następna sekcja) osiągają wartości nieokreśloności ok. 125, co jest wynikiem osiąganym również przez niektóre modele oparte na sieciach rekurencyjnych. Wykorzystanie sieci rekurencyjnych opartych o neurony LSTM pozwoliło jednak na redukcje nieokreśloności do wartości 75-80, a modele oparte na architekturze transformer potrafią osiągać ok. 56. Wyniki dla języka polskiego są jednak znaczenie wyższe (czyli gorsze). Na korpusie składającym się m.in. z polskiej Wikipedii, publicznej części Narodowego Korpusu Języka Polskiego, wiadomości z forów internetowych i książek najlepszy osiągnięty wynik przez podejścia neuronowe to nieokreśloność równa 117, podczas gdy model 5-gramowy (z wygładzaniem) osiąga nieokreśloność 146 [4]. Na chwilę obecną nie widać więc dużego przełomu w modelowaniu języka polskiego przy użyciu podejść neuronowych.

Problem 2.2 Na stornie internetowej `https://planspace.org/perplexity/` zmierz swoją własną nieokreśloność. Aplikacja liczy nieokreśloność po literach angielskich a nie po słowach – stąd uzyskiwane wartości będą bardzo niskie (tylko ok. 30 możliwości zamiast $|V|$!).

2.2 Wygładzanie modeli językowych

Modele n-gramowe pomimo swojej prostoty potrafią zamodelować wiele użytecznych zależności w języku. Model n gramowy nauczy się, że po słowie „dzień” najczęściej stoi „dobry” i że po „Los” może występować „Angeles” albo „Alamos”, gdzie to pierwsze jest dużo częstsze. Będzie też wiedział że po „ja” stoją czasowniki odmienione w pierwszej osobie oraz wie że po takim czasowniku zwykle jest przymiotnik lub rzeczownik.

Modele n-gramowe mają też swoje ograniczenia. Najbardziej oczywistym jest brak możliwości modelowania długich zależności pomiędzy słowami, ale nie najważniejszym. Głównym problemem tych modeli to bardzo małe uogólnianie wiedzy, którego jednym źródłem jest pocięcie zdania na n-elementowe, nakładające się na siebie kawałki. Z tego powodu obecne w korpusie powyższe zdanie zwiększyłoby w modelu trzy-gramowym nie tylko prawdopodobieństwo siebie samego, ale także podwyższyłoby wszystkie zdania zaczynające się od „Głównym problemem” czy zawierające „którego jednym źródłem”.

Jest to jednak jedyna forma uogólniania w podstawowej wersji modelu n-gramowego, co jest jego mocnym ograniczeniem.

Po pierwsze, aby uzyskać dokładniejszy model języka powinniśmy wydłużyć rozmiar n-gramu jednak powoduje to gorsze uogólnianie wiedzy bo im dłuższy zbitek słów, tym rzadziej pojawia się on w korpusie. Jednocześnie, wraz z rosnącą długością zlepków słów, liczba wszystkich możliwych n-gramów rośnie eksponencjalnie[2] , a wraz z nią liczba prawdopodobieństw które potencjonalnie trzeba oszacować w modelu języka. Prowadzi to (nawet dla modeli trzy-gramowych) do problemu rzadkości danych (ang. *data sparsity*) czyli sytuacji w której zdecydowana większość estymat jest równa zero, a reszta najczęściej ma bardzo niskie wsparcie w danych tj. obarczona jest dużą niepewnością (wariancją). W szczególności, w czasie używania modelu na zdaniach spoza korpusu uczącego używamy estymat n-gramów które nigdy nie występowały w zbiorze danych (zero!) albo występowały bardzo mało razy (duża wariancja!).

Problem zerowych estymat można prosto rozwiązać poprzez zastosowanie wygładzania addytywnego tj. dodawać do wartości zliczenia zawsze jakąś stałą liczbę k. Wygładzanie Laplace'a, które poznałeś na uczeniu maszynowym, jest właśnie taką techniką z ustalonym $k = 1$ – odstępujemy więc od jej opisu. Należy jednak wspomnieć, że wygładzanie Laplace'a (bez strojenia wartości k) jest bardzo złą techniką dla modeli języka i nie należy jej stosować. Jednak nawet gdy k jest strojone: wynikiem jest model który traktuje wszystkie niewidziane w korpusie uczącym n-gramy za równie prawdopodobne! Jest to daleko idące uproszczenie, którego możemy uniknąć stosując technikę wygładzania rekurencyjnego (ang. *back-off model*).

Problem 2.3 Jeżeli w korpusie testowym obecne jest zdanie zawierające n-gram który nie był obecny w zbiorze uczącym – jaka będzie wartość nieokreśloności zakładając brak rozmywania?

A gdyby tak stworzyć oba modele i wziąć z nich to co najlepsze?

W praktyce, jeżeli rzadkość danych nie jest problemem w twoim modelu języka... oznacza to że skonstruowany model jest zwyczajnie zbyt prosty, a poprzez zwiększenie długości n-grama powinieneś być w stanie poprawić jakość działania modelu.

2.2.1 Wygładzanie rekurencyjne

Wygładzanie rekurencyjne rozwiązuje problem zerowych estymat poprzez wykorzystanie następującej obserwacji. Rozważając model trzy-gramowy, zerowe prawdopodobieństwo $P(w_i|w_{i-1}, w_{i-2})$ jakiejś części zdania oznacza, że trzy-gram „$w_{i-2}\ w_{i-1}\ w_i$" nigdy nie wystąpił w zbiorze uczącym. Jest jednak spora szansa, że bigram „$w_{i-1}\ w_i$" zostałby odnaleziony w korpusie! Pomysł jest więc bardzo prosty: używamy modelu trzy-gramowego, ale jeżeli dla danego słowa model zwraca zerowe prawdopodobieństwo to sprawdzamy prawdopodobieństwo w modelu bigramowym. Jeżeli prawdopodobieństwo nadal jest zerowe: użyjmy modelu unigramowego. Nadal zero? Cóż, słowo jest spoza słownika, a obsługą takich słów zajmiemy się w następnej sekcji.

[2] Pracując ze słownikiem $|V| = 10.000$ istnieje 10 tysięcy unigramów, 100 milionów bigramów i bilion trzy-gramów. Rozmiar korpusów używanych do modelowania języka systematycznie powiększa się, jednak raczej nie eksponencjalnie...

Idea jest prosta, ale jej użycie wymaga pewnej pracy gdyż po prostu wcielenie jej w życie spowoduje uzyskanie nieprawidłowego rozkładu prawdopodobieństwa. W szczególności prawdopodobieństwa nie będą sumowały się do 1. Zauważ, że wszystkie prawdopodobieństwa warunkowe $P(w_i|w_{i-1},w_{i-2})$ sumują się do 1 dla konkretnego warunku np. $\sum_{w_i \in V} P(w_i|w_{i-1} = Ala, w_{i-2} = ma) = 1$. Jeśli więc jeden z zerowych składników tej sumy zastąpimy niezerowym prawdopodobieństwem z modelu bigramowego, musimy uzyskać wynik większy niż 1. Będziemy musieli odjąć trochę prawdopodobieństwa z modelu trzy-gramowego, aby zrobić miejsce na potencjalne prawdopodobieństwa z modelu bi-gramowego.

ADVANCED

Zanim przejdziemy do pokazywania formuł, dla uzyskania większej ogólności zapisu wprowadźmy notację w_i^j która będzie oznaczała n-gram zaczynający się od pozycji i-tej zdania i kończący się na pozycji j-tej. Wcześniej używane „$w_{i-2}\ w_{i-1}\ w_i$" staje się więc jednym symbolem w_{i-2}^i. Zauważ, że teraz wszystkie modele n-gramowe (uni-, bi-, trzy-, cztero-, ... gramy) można zapisać jako:

$$P(s) = P(w_1^{|s|+1}) = \prod_{i=1}^{|s|+1} P(w_i|w_{i-n+1}^{i-1})$$

gdzie $|s|$ to liczba słów w zdaniu s.

W standardowym modelu n-gramowym, bez wygładzania, kolejne prawdopodobieństwa estymujemy wzorem:

$$\hat{P}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i)}{\sum_{v \in V} c(w_{i-n+1}^{i-1} v)}$$

gdzie $c(w_{i-n+1}^i)$ zwraca liczbę wystąpień n-gramu w_{i-n+1}^i w korpusie uczącym. Wyjaśniając ew. wątpliwości dot. notacji: poprzez $c(w_{i-n+1}^{i-1} v)$ rozumiemy liczbę wystąpień n-gramu w_{i-n+1}^i z zastąpionym ostatnim słowem w_i przez słowo v (czyli jest to liczba wystąpień n-gramu $w_{i-n+1}\ w_{i-n+2}\ w_{i-n+3}\ \ldots\ w_{i-1}\ v$ w korpusie uczącym).

Aby „zrobić miejsce" na wygładzanie rozkładu prawdopodobieństw n-gramami niższego rzędu, użyjemy formuły zdyskontowanej (ang. *discounting*) dla wyznaczenia prawdopodobieństwa n-gramu. Konkretnie, pomniejszymy liczbę wystąpień każdego n-gramu o pewną stałą d, otrzymując wzór: $\frac{c(w_{i-n+1}^i)-d}{\sum_{v \in V} c(w_{i-n+1}^{i-1} v)}$. Stała d jest dodatnia i mniejsza od 1, a jej typowe ustawienie to $d = 0.75$. Formułę zdyskontowaną stosuje się tylko do n-gramów istniejących w korpusie uczącym (nie odejmujemy d od zera, bo prawdopodobieństwo nie może być ujemne).

Łatwo zauważyć, że estymaty zdyskontowane nie sumują się do 1 i zostawiają pewną masę prawdopodobieństwa do przypisania modelom niższego rzędu. Liczbę takich n-gramów możemy wyrazić matematycznie jako $|\{v \in V : c(w_{i-n+1}^{i-1} v) > 0\}|$ czyli od sumy prawdopodobieństw odjęliśmy łącznie

$$\lambda(w_{i-n+1}^{i-1}) = \frac{d \cdot |\{v \in V : c(w_{i-n+1}^{i-1} v) > 0\}|}{\sum_{v \in V} c(w_{i-n+1}^{i-1} v)}$$

. Mnożąc model $n-1$-gramowy (sumujący się do 1) przez $\lambda(w_{i-n+1}^i)$ i dodając do niego zdyskonotowane estymaty modelu n-gramowego otrzymujemy sumę prawdopodobieństwa

równą 1. Ostatecznie więc naszą estymatę możemy zapisać jako

$$\hat{P}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^{i})-d}{\sum_{v\in V} c(w_{i-n+1}^{i-1}v)} & \text{jeżeli } c(w_{i-n+1}^{i}) > 0 \\ \lambda(w_{i-n+1}^{i-1})\hat{P}(w_i|w_{i-n+2}^{i-1}) & \end{cases} \tag{2.2}$$

Zwróć uwagę, że powyższa definicja jest rekurencyjna – jeśli na poziomie $n-1$ gramu nadal liczba wystąpień będzie równa zero, zbadamy model $n-2$ gramowy itd. W przypadku dojścia do unigramu są dwie możliwości: a) zastosowanie estymat bez dyskontowania i zakończenie rekursji – jeśli nie ma unigramu to zwracamy 0 b) stosujemy estymaty zdyskontowane i gdy nie znajdziemy unigramu stosujemy rozkład jednorodny czyli: $\hat{P}(\emptyset) = \lambda(w_i^i)\frac{1}{|V|}$.

Powyższy zapis równania 2.2 jest obarczony pewną nieścisłością: prezentowany rozkład sumuje się do liczby mniejszej niż 1. Stosując rekurencyjnie podany wzór dla estymat niższego rzędu one np. normalizowane sumą po wszystkich słowach w słowniku $v \in V$. Zwróć jednak uwagę, że estymacje niższego rzędu nigdy nie zostaną użyte o ile istnieje niezerowa estymata wyższego rzędu dla danego słowa! Aby uzyskać prawidłową estymatę z podanego wzoru, odwołując się do modelu niższego rzędu powinniśmy pomniejszyć jego V o słowa do których istnieją n-gramy wyższego rzędu.

■ **Przykład 2.1** Załóżmy, że $V = \{a,b,c,d\} \cup \boxed{\text{STOP}}$, $d = 0.25$ a korpus uczący dla modelu trzy-gramowego zawiera zdania:

- a a a
- b b a c
- a
- c d

Oblicz pełen rozkład dla $P(w_i|w_{i-1} = a, w_{i-2} = a)$ i sprawdź czy sumuje się do 1. Załóż, że estymaty unigramowe są liczone bez dyskontowania.

Wyznaczmy najpierw prawdopodobieństwa $P(a|aa)$ i $P(\boxed{\text{STOP}}|aa)$. W korpusie uczącym widzimy „a a" tylko w pierwszym zdaniu i raz kończy ono zdanie, a raz jest ono kontynuowane przez „a". Otrzymujemy więc

$$c(aaa) = 1 \quad c(aab) = 0 \quad c(aac) = 0 \quad c(aad) = 0 \quad c(aa\boxed{\text{STOP}}) = 1$$

Bez wykonywania rekurencji do estymat bi-gramowych możmy od razu obliczyć:

$$P(a|aa) = \frac{1-d}{2} = \frac{3}{8} \qquad P(\boxed{\text{STOP}}|aa) = \frac{1-d}{2} = \frac{3}{8}$$

Przechodząc do estymat bigramowych potrzebujemy obliczyć wartość $\lambda(aa)$ warunku – czyli określić ile prawdopodobieństwa zostało nam do przydzielenia niższym estymatom.

$$\lambda(aa) = \frac{d \cdot |\{v \in V : c(w_{i-2}^{i-1}v) > 0\}|}{\sum_{v\in V} c(w_{i-2}^{i-1}v)} = \frac{0.25 \cdot |\{a, \boxed{\text{STOP}}\}|}{\sum_{v\in V} c(aav)} = \frac{0.25 \cdot 2}{2} = \frac{1}{4}$$

. Skracamy warunek i szukamy $\hat{P}(w_i|w_{i-1}=a)$ zawężając słownik do $V=\{b,c,d\}$

$$c(ab)=0 \quad c(ac)=1 \quad c(ad)=0$$

Bez wykonywania rekurencji do estymat unigramowych możemy obliczyć

$$\hat{P}(c|aa)=\lambda(aa)\cdot\frac{1-d}{1}=\frac{1}{4}\cdot\frac{3}{4}=\frac{3}{16}$$

Przechodząc do estymat unigramowych potrzebujemy obliczyć wartość $\lambda(a)$ warunku :

$$\lambda(a)=\frac{d\cdot|\{v\in V: c(w_{i-1}v)>0\}|}{\sum_{v\in V}c(w_{i-1}v)}=\frac{0.25\cdot|\{c\}|}{\sum_{v\in V}(av)}=\frac{0.25\cdot 1}{1}=\frac{1}{4}$$

.

Skracamy warunek i szukamy $\hat{P}(w_i)$ zawężając słownik do $V=\{b,d\}$

$$c(b)=2 \quad c(d)=1$$

$$P(b|aa)=\lambda(aa)\lambda(a)\frac{2}{3}=\frac{2}{48} \qquad P(d|aa)=\lambda(aa)\lambda(a)\frac{1}{3}=\frac{1}{48}$$

Sprawdźmy czy prawdopodobieństwa sumują się do 1:

$$\frac{1}{48}+\frac{2}{48}+\frac{3}{16}+\frac{3}{8}+\frac{3}{8}=1$$

. Porównaj otrzymane estymaty: model spodziewa się że najbardziej prawdopodobnymi kontynuacjami „a a" będzie „a" lub „STOP" gdyż widział takie przykłady w korpusie. Spodziewa się jednak, ze jest mała szansa, że będzie to „c" bo widział „a c". W końcu jest malutka szansa że będzie to „b" lub „d" – jednak „b" występuje w korpusie dwukrotnie częściej niż „d" więc prawdopodobieństwo przydzielamy proporcjonalnie.
■

Model n-gramowy z trochę bardziej wyrafinowaną wersją wygładzania rekurencyjnego, nazywanym zmodyfikowanym wygładzaniem Knessera-Neya (ang. *modified Knesser-Ney smoothing*), przez ponad dekadę był punktem odniesienia dla zadania modelowania języka, skutecznie konkurując z chociażby z modelami neuronowymi. Metoda Knesser-Neya modyfikuje modele n-gramowego niższego rzędu, zastępując je estymatami prawdopodobieństwa że dany $n-1$ gram nie ma swojego n-gramowego rodzica w korpusie uczącym Metoda jest opisana np. w książce [2] do której odsyłam zainteresowanych czytelników.

2.2.2 Interpolowanie kilku modeli

Alternatywnym pomysłem do wygładzenia rekurencyjnego, jest interpolowanie modeli języka. Wygładzanie rekurencyjne sięga po n-gramy niższego rzędu tylko wtedy gdy n-gramy wyższego rzędu nie istnieją w zbiorze uczącym. Z kolei modele interpolacyjne, zawsze wykorzystują n gramy niższego rzędu, ale z mniejszą wagą. Taka strategia jest potencjalnie lepsza, gdy estymaty wyższego rzędu są tworzone w oparciu o zaledwie kilka (w szczególności: jedno) wystąpienie w korpusie i są obarczone wysoką wariancją. Wygładzenie takiej estymaty pewniejszymi estymatami niższego rzędu może przynieść

dobry efekt. Z drugiej strony: n-gramy niższego rzędu modelują krótsze zależności i powinniśmy preferować korzystanie z n-gramów wyższych rzędów o ile tylko istnieją – filozofia wygładzania rekurencyjnego. W praktyce obie techniki często działają podobnie, a konkretne pomysły na modele interpolacyjne są często przerabiane na ich rekurencyjne odpowiedniki i na odwrót.

Interpolowany model trzy-gramowy wyrażony jest wzorem:

$$P(s) = \prod_{i=1}^{|s|+1} \lambda_1 P(w_i|w_{i-2}, w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i)$$

gdzie $\lambda_i > 0$ to pewne wybrane stałe, które sumują się do 1. Modele interpolacyjne najczęściej uczy się dwuetapowo. W pierwszym etapie normalnie konstruuje się modele n-gramowe, a następnie na zbiorze walidacyjnym szuka się dobrych wartości λ_i.

Główną różnicą pomiędzy różnymi propozycjami modeli interpolacyjnymi to liczba i sposób wyznaczania współczynników λ_i. W powyższym wzorze zastosowane statyczne współczynniki λ_i, które nie zależą ani od n-gramu ani od jego liczności. Bardziej zaawansowane techniki używają dynamicznych współczynników λ_i, które są funkcjami zależącymi od wartości n-gramów $\lambda(w_{i-n+1}^i)$. Duża liczba współczynników λ czyni ich strojenie/naukę trudniejszym.

W praktyce czasami stosujemy interpolowanie jako metodę łączenia kilku różnych modeli (niekoniecznie n-gramowych i niekoniecznie niższych rzędów) w jeden silniejszy model.

Problem 2.4 Udowodnij, że prawdopodobieństwa modelu interpolowanego sumuje się do 1.

2.3 Problem słów spoza słownika

Niezależnie od korzystania z dobrodziejstw modelu interpolowanego lub wygładzania rekurencyjnego, trzeba zdecydować się na jakąś metodę radzenia sobie ze słowami spoza słownika czyli słów które nie wystąpił w zbiorze uczącym a występują w czasie predykcji.

Token UNK

Najprostszą taką techniką jest wprowadzanie do słownika dodatkowego sztucznego tokenu/słowa UNK reprezentującego nieznane słowa, a w momencie predykcji każde nierozpoznane słowo zastępować właśnie tym tokenem. Proces uczenia modelu z tokenem UNK poprzedza się wtedy dodatkowym etapem wstępnego przygotowania danych (ang. *data preprocessing*), w którym wszystkie słowa które wystąpiły w korpusie $\leq k$ razy są zastępowane tokenem słowa nieznanego. Częstą wartością dla k jest po prostu 1 (słowa takie i tak nie pozwolą na uzyskanie wysokiej jakości estymat) lub 5 (po prostu okrągła, mała liczba :). Czasami istnieją ograniczenia pamięciowe na rozmiar słownika – w takim wypadku definiujemy słownik z najczęstszych słów, a całą resztą zamieniamy na UNK .

Problem 2.5 Przy omawianiu nieokreśloności zwróciliśmy uwagę, że należy ją porównywać tylko na modelach korzystających z tego samego słownika. W kontekście wykorzystania słowa UNK – dlaczego jest to szczególnie ważne?

Pseudo-słowa

Bardziej wyrafinowaną wersją powyższej techniki jest wprowadzenie całego zestawu specjalnych tokenów dla nieznanych słów. Technika pseudo-słów pozwala na rozwiązanie problemu słów z niską częstotliwością bez całkowitego tracenia wszystkich informacji w nich zawartych. Każde słowo reprezentuje całą klasę rzadkich słów, który pasują do jakiegoś wyrażenia regularnego. Na przykład, możemy stworzyć wyrażenie regularne na tokeny składające się z samych cyfr i zastąpić je tokenem NUMBER , słowa składające się z samych wielkich liter ABBREV (są to często skróty np. USA, PIT, PP), a wszystkie nieznane słowa zaczynające się „http://" zastąpić URL . Oczywiście stworzenie takiej listy wyrażeń regularnych i zasad ich zamiany na tokeny muszą być zdefiniowane ręcznie przez programistę.

2.4 Klasowe modele języka

Wykorzystanie interpolacji, wygładzania rekurencyjnego lub wygładzania addytywnego wraz z obsługą słów spoza słownika pozwalają nam na uniknięcie problemu zerowego prawdopodobieństwa. Dodatkowo, interpolacja oraz wygładzanie rekurencyjne pozwalają nam trochę poprawić uogólnianie wiedzy: obecność trzy-gramu „Ala pisze powieść" zwiększa prawdopodobieństwo nie tylko tego konkretnego zlepka słów, ale także zlepek „Ala pisze" czy „pisze powieść" oraz każdego z pojedynczych słów. Generalizacja wiedzy w modelu n-gramowym następuje poprzez sklejanie ze sobą n-elementowych sekwencji słów, które nakładają się na siebie. Odzwierciedla to pewną intuicję, że nie wszystko jest w tekście połączone, a słowa bliżej siebie w zdaniu są zwykle mocniej statystycznie powiązane.

Pisaliśmy wcześniej, że model n-gramowy jest w stanie nauczyć się w ten sposób prawidłowych odmian np. że po „ona" będzie „pisze", „organizuje" ale nie „piszemy" czy „organizujemy". Będzie to po prostu wynikać ze zlepek słów zawierających wyraz „ona", co równocześnie oznacza że model nauczy się że po „ona" są czasowniki w trzeciej osobie tylko kiedy korpus zawiera wszystkie możliwe bigramy „ona + odmieniony czasownik". Co więcej, aby nauczyć model że po „Ala" czasowniki są odmienione w ten sam sposób: potrzebujemy korpus który zawiera wszystkie możliwe czasowniki po słowie Ala. Wystarczy pomyśleć o liczbie kombinacji wszystkich możliwych żeńskich imion z czasownikami by domyśleć się, że posiadanie korpusu z wystąpieniami tych wszystkich bigramów jest praktycznie niemożliwe.

Nie dotyczy to z resztą tylko odmiany czasowników: modele n-gramowe traktują każde słowo jako całkowicie odrębną jednostkę, która nie jest w żaden sposób podobna do żadnego innego słowa. Dlatego też model n-gramowy po zobaczeniu w korpusie „Pojechałem do szkoły metrem" nigdy się nie domyśli że zdania „Pojechałem do szkoły tramwajem" czy „Pojechałem do szkoły rowerem" są też prawdopodobne. Co więcej, możliwe że te zdania są dla niego tak samo podobne do oryginału jak „Pojechałem do szkoły być" czy „Pojechałem do szkoły śruba".

Dobry model języka powinien jednak potrafić wydedukować powiązania między słowami na podstawie korpusu. Załóżmy, ze w korpusie mamy zdania: „Pojechałem do szkoły metrem", „Pojechałem do szkoły tramwajem", „Pojechałem do teatru metrem", „Pojechałem do teatru samochodem" oraz „Śmignąłem do szkoły metrem". Z dwóch pierwszych zdań powinniśmy nauczyć się że „metrem" i „tramwajem" są podobne, ponieważ

są możliwymi kontynuacjami „Pojechałem do szkoły”. Dalej, z kolejnych dwóch zdań możemy wywnioskować że „metro” i „samochód” są podobne a z naszych wcześniejszych obserwacji stwierdzić że „samochód” i „tramwaj” są podobne oraz że... „Pojechałem do szkoły samochodem” jest również całkiem prawdopodobnym zdaniem. Z kolei szkoła w tym kontekście jest podobna do teatru, więc do teatru pewnie również można pojechać tramwajem. Ostatnie zdanie sugeruje że „śmignąłem” jest podobne do „pojechałem” i praktycznie można je zastąpić we wszystkich czterech pierwszych zdaniach.

Zauważ, że wszystkie zdania z tego korpusu możemy zastąpić takim generycznym schematem:

$$C_1\ C_2\ C_3\ C_4$$

gdzie $C_1 \in \{\text{Śmignąłem}, \text{Pojechałem}\}$, $C_2 \in \{\text{do}\}$, $C_3 \in \{\text{szkoły}, \text{teatru}\}$, a $C_4 \in \{\text{metrem}, \text{samochodem}, \text{tramwajem}\}$. Schemat ten jest sekwencją-zdaniem stworzoną nie ze słów, ale z *typów słów*. Ten jeden schemat nie tylko zapisuje wszystkie zdania z korpusu ale także automatycznie opisuje 7 innych zdań czyli więcej niż duplikując rozmiar korpusu. Jeśli model języka w pewnym momencie nauczy się że do C_3 również należy „urzędu” automatycznie uogólni to na 8 nowych zdań. Co więcej używając tych samych typów słów można zapisywać inne zdania jak $C_2\ C_3\ C_1\ C_4$ np. „Do szkoły pojechałem metrem” czy $C_1\ C_4$ „Pojechałem rowerem”. Model języka, który byłby w stanie nauczyć się typów słów C_i i wyrażać przy ich użyciu zadania w takich generatywnych schematach miałby potencjalnie dużo większe możliwości uogólniania wiedzy.

Pewnym mankamentem tego pomysłu jest to, że model traktujący zdania ze „Śmignąłem” identycznie jak zdania z „Pojechałem” musiałby przypisywać im takie samo prawdopodobieństwa (skoro są takie same). Empirycznie wiemy jednak że zdania ze „Śmignąłem” są raczej dużo mniej prawdopodobne. Tę wadę można jednak prosto rozwiązać: do listy przechowującej słowa danego typu można dopisać liczbę oznaczającą prawdopodobieństwo, że dany typ zamienia się w zdaniu na dane słowo np. C_1 można przechowywać jako $[\text{Śmignąłem}(0.2), \text{Pojechałem}(0.8)]$ lub zapisując to bardziej formalnie: $P(\text{Śmignąłem}|C_1) = 0.2$ i $P(\text{Pojechałem}|C_1) = 0.8$.

Modele działające w ten sposób mają duże możliwości uogólniania wiedzy, co niestety czasami prowadzi do zbyt dużej generalizacji. Kontynuując przykład: na podstawie zdania „Poszedłem do szkoły” model mógłby odkryć że do C_1 może należeć też „Poszedłem”. Jednak używając schematu $C_1\ C_2\ C_3\ C_4$ byłoby możliwe wtedy wygenerowanie „Poszedłem do szkoły samochodem”.

2.4.1 Model n-gramów klas

Modele n-gramów klas lub modele Browna [1] są modelami przypisującymi słowom klasy/typy i modelujące zdanie przez prawdopodobieństwa n-gramów klas/typów. W czasie nauki model taki musi zrealizować jednocześnie operację grupowania słów w typy oraz samą estymację modelu języka. Nie jest to łatwe zadanie i zostało zaproponowanych w literaturze co najmniej kilka algorytmów uczących dla tego modelu, niektóre z nich np. uwzględniają możliwość przypisania jednemu słowu kilka typów. Tutaj opiszemy tylko jeden z nich, który przypisuje słowom tylko jeden typ C_i a typ słowa przemienia się w zdaniu na konkretne słowo w z pewnym prawdopodobieństwem $P(w|C_i)$.

Poprzez $C(w)$ oznaczmy funkcję zwracającą grupę (typ) przypisany do słowa w. Na przykład, w ostatnim przykładzie funkcja $C(szkoły)$ zwróciłaby C_3. Ponieważ przypisanie słowa do typu jest deterministyczne[3]

$$P(w_1, w_2, ..., w_n) = P(w_1, w_2, ..., w_n, C(w_1), C(w_2), ..., C(w_n))$$

i stosując $P(w,c) = P(w|c)P(c)$ otrzymujemy:

$$P(w_1, w_2, ..., w_n) = \underbrace{P(w_1, w_2, ..., w_n | C(w_1), C(w_2), ..., C(w_n))}_{\text{prawdopodbieństwo przypisania słów do typów}} \underbrace{P(C(w_1), C(w_2), ..., C(w_n))}_{\text{prawdopodbieństwo schematu}}$$

Wzór ten można dalej prosto uprościć. Analizując pierwszy czynnik: zakładamy, że słowo w_i jest generowane z pewnym prawdopodobieństwem z typu $C(w_i)$, całkowicie niezależnie od innych słów w zadaniu oraz ich typów. Możemy więc dokonać faktoryzacji tego prawdopodobieństwa

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i | C(w_i)) P(C(w_1), C(w_2), ..., C(w_n))$$

Z kolei do schematu zdania stosujemy standardowy model n-gramowy. Przykładowo dla trzy-gramu wzór przekształcamy do

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n+1} P(w_i | C(w_i)) P(C(w_i) | C(w_{i-1}), C(w_{i-2}))$$

Powyższy zapis, podobnie jak poprzednio, zakłada że na końcu zdania znajduje się $\boxed{\text{STOP}}$, a na początku znajdują się tokeny $\boxed{\text{START}}$. Obydwa tokeny specjalne mają swoje własne typy do których należą tylko i wyłącznie one same. Prawdopodobieństwa $P(w_i|C(w_i))$ czasami nazywane są prawdopodobieństwami emisji, gdyż określają z jakim prawdopodobieństwo słowo w_i jest emitowane z $C(w_i)$. Z kolei prawdopodobieństwa $P(C(w_i)|C(w_{i-1}), C(w_{i-2}))$ nazywane są prawdopodobieństwami tranzycji.

Algortym uczący

Przejdźmy do opisu algorytmu uczącego. Na wstępie zauważmy, że mając już gotowy wynik grupowania czyli przypisanie słów do ich typów, wyestymowanie prawdopodobieństwa emisji jest bardzo proste.

$$\hat{P}(w|C_j) = \frac{c(w)}{\sum_{v \in C_j} c(v)} \tag{2.3}$$

gdzie $c(w)$ to liczba wystąpień słowa w w korpusie. Dalej, mając wynik grupowania, estymacja prawdopodobieństw tranzycji odbywa się analogicznie do modelu n-gramowego. W korpusie uczącym podmieniamy wszystkie słowa na odpowiadającą im grupę (jest to deterministyczna zmiana, bo każde słowo należy tylko do jednego typu $\exists_j P(C_j|w_i) = 1$), a następnie stosujemy klasyczne zliczanie (lub nawet z jakimś wygładzaniem).

$$\hat{P}(C(w_i)|C(w_{i-1}), C(w_{i-2})) = \frac{c(C(w_{i-2}), C(w_{i-1}), C(w_i))}{c(C(w_{i-2}), C(w_{i-1}))} \tag{2.4}$$

[3] Dla zachowania prostoty, operujemy na sekwencjach o stałej długości n, a na końcu rozwiążemy problem różnej długości dopisując token $\boxed{\text{STOP}}$.

Głównym problem jest więc konieczność wykonania jednoczesnego grupowania i estymacji prawdopodobieństwa, aby uzyskać jak najlepszy model np. maksymalizując wartość funkcji wiarygodności na zbiorze uczącym (czyli minimalizując nieokreśloność).

$$\ell_{train} = \sum_{s \in D_{train}} \sum_{i=1}^{n_s+1} \log P\left(w_i^{(s)} | C(w_i^{(s)})\right) P\left(C(w_i^{(s)}) | C(w_{i-1}^{(s)}), C(w_{i-2}^{(s)})\right) \tag{2.5}$$

Pewną rozsądnie działającą heurystykę oferuje poznane na uczeniu maszynowym aglomeracyjne grupowanie hierarchiczne (ang. *agglomerative hierarchical clustering*), które jest algorytmem zachłannym. Algorytm zaczyna od grup o wielkości 1, a następnie rozważając wszystkie możliwe pary scalenia skupień łączy te które maksymalizują funkcję celu. Zaczniemy więc od liczby typów równej rozmiarowi słownika $|V|$, przypisując każdemu słowu swój własny typ. Mając przypisane słowa do grup, obliczenie funkcji celu 2.5 jest już możliwe przy wykorzystaniu estymat 2.3 i 2.4. Algorytm próbnie łączy każdą parę typów w jeden typ i oblicza dla każdej próby funkcję celu. Po przetestowaniu wszystkich możliwych połączeń wybiera się to, które dało najwyższą wartość funkcji celu i powtarza się procedurę.

Procedurę przerywa się po osiągnięciu określonej z góry liczby typów. Po zakończeniu grupowania zwykle próbuje się jeszcze stosować jakieś procedury poprawy przypisań słów do grup. Najczęściej iterując po słowniku, przypisując słowo do grupy która zwiększa funkcję celu tak długo aż osiągnie się lokalnie optymalne rozwiązanie.

Jak pewnie zauważyłeś, opisana wyżej metoda jest bardzo kosztowna obliczeniowo. Algorytm wymaga $O(|V|)$ operacji łączenia z który w każdej musimy rozważyć wszystkie pary $O(|V|^2)$ i dla każdej z par obliczyć funkcję celu która iteruje po wszystkich tokenach w korpusie $O(N)$ używając estymat których obliczenie również kosztuje $O(N)$. Łącznie otrzymujemy $O(N^2|V|^3)$, co jest olbrzymią liczbą. Na szczęście wykonując kilka przekształceń i mądrze cachując wyniki pośrednie [1] możemy ograniczyć złożoność (dla modelu bigramowego) do $O(|V|^3)$, co... nadal jest kosztowne obliczeniowo. Z tego powodu do tego heurystycznego algorytmu stusuje się dalsze herustyki mające na celu redukcję kosztu obliczeniowego np. [3].

■ **Przykład 2.2** Zakładając korpus:

- Pojechałem rowerem
- Pojechałem samochodem
- Śmignąłem samochodem

wykonaj pierwszą iterację algorytmu uczącego bi-gramowy model klasowy.

Zgodnie z treścią zadania będziemy uczyć model bi-gramów klas, który wyrażamy wzorem

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n+1} P(w_i | C(w_i)) P(C(w_i) | C(w_{i-1}))$$

zauważ, że mnożenie jest rozdzielne tj. możemy najpierw przemnożyć wszystkie prawdopodobieństwa emisji, a potem przemnożyć ten wynik przez prawdopodobieństwa tranzycji. W praktyce stosujemy logarytm funkcji wiarygodności tak jak pokazano we wzorze 2.5, jednak dla zachowania prostoty zapisu na kartce będziemy pracowali na samej funkcji wiarygodności (bez logarytmu) co nie wpływa na zmianę wyniku. Wartość funkcji wiarygodności możemy obliczyć wymnażając wszystkie prawdopodo-

bieństwa emisji i wszystkie prawdopodobieństwa tranzycji wszystkich zmian (porównaj ze wzorem 2.5) .

W pierwszej iteracji wszystkie słowa mają swoje własne typy, więc (skoro są jedynym elementem grupy) są emitowane z prawdopodobieństwem 1. Przemnożenie więc przez wszystkie prawdopodobieństwa emisji daje po prostu 1. Pozostaje więc przemnożyć przez prawdopodobieństwa tranzycji np. $P(Pojechałem, \boxed{\text{START}}) = \frac{2}{3}$ bo dwa razy widzimy zdanie rozpoczynające się od „Pojechałem" a raz od „Śmignąłem".

Aby usystematyzować nasze obliczenia zapiszemy je w postaci tabeli. W pierwszej kolumnie umieścimy wymnożone prawdopodobieństwo emisji (E) po wszystkich słowach, a następnie dla każdego kolejnego prawdopodobieństwa tranzycji zarezerwujemy jedną kolumną. Aby uzyskać więc wartość funkcji celu wystarczy będzie przemnożyć wszystkie liczby w danym wierszu. Taką tabelkę z obliczeniem funkcji celu dla początkowego rozwiązania prezentujemy poniżej:

E	P|⋆	r|P	STOP |r	P|⋆	s|P	STOP |s	Ś|⋆	s|Ś	STOP |s	ℓ
1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{1}{3}$	1	1	0.037

Aby oszczędzić miejsce kolumny ponazywaliśmy skrótami i tak $P(C(Pojechałem)|C(\boxed{\text{START}}))$ stało się $P|\star$ gdzie P to pierwsza litera „Pojechałem", a $\boxed{\text{START}}$ zastąpiliśmy znakiem gwiazdki. Analogicznie $P(C(rowerem)|C(Pojechałem))$ zostało zapisane jako $r|P$.

Algorytm aby zdecydować o złączeniu dwóch grup w jedną, musi przeanalizować wszystkie możliwości. Zacznijmy o przeanalizowania połączenia „Pojechałem" i „Śmignąłem" w jedną grupę. Prawdopodobieństwa emisji nie będą już wynosić 1, gdyż „Pojechałem" (P) będzie emitowane z prawdopodobieństwem $\frac{2}{3}$, a „Śmignąłem" (Ś) z prawdopodobieństwem $\frac{1}{3}$ (grupa zawiera dwa słowa, ale P występuje w korpusie dwukrotnie). Reszta emisji będzie nadal równa 1, więc wszystkie prawdopodobieństwa emisji w funkcji celu przemnożone przez siebie wyniosą $\frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{27}$. Jeśli Ś jest połączone z P to pozmieniają się też prawdopodobieństwa tranzycji dotyczące tych słów. Przykładowe prawdopodobieństwo $P(C(Pojechałem)|C(\boxed{\text{START}}))$ wzrośnie do $\frac{3}{3} = 1$, bo każde zdanie zaczyna się od słowa o takim typie! Z kolei $P(C(rowerem)|C(Pojechałem))$ spadnie do $\frac{1}{3}$ – występują w korpusie trzy bi-gramy rozpoczynające się słowem o typie $C(Pojechałem)$ i tylko jeden raz jest on kontynuowany poprzez $C(rowerem)$... Końcowy wynik prezentowany jest w tabelce.

	E	P|⋆	r|P	STOP |r	P|⋆	s|P	STOP |s	Ś|⋆	s|Ś	STOP |s	ℓ
	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{1}{3}$	1	1	0.0370
P,Ś	$\frac{4}{27}$	1	$\frac{1}{3}$	1	1	$\frac{2}{3}$	1	1	$\frac{2}{3}$	1	0.0219

W analogiczny sposób powinniśmy obliczyć funkcję celu dla wszystkich możliwych połączeń – jednak poniżej zaprezentuje wyniki dla dwóch kolejnych kandydatów do połączenia: (r,s), który wydaje nam się intuicyjnie dobry patrząc na korpus oraz (P,r) który wydaje się nieodpowiedni.

	E	P|⋆	r|P	STOP |r	P|⋆	s|P	STOP |s	Ś|⋆	s|Ś	STOP |s	ℓ
	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{1}{3}$	1	1	0.0370
P,Ś	$\frac{4}{27}$	1	$\frac{1}{3}$	1	1	$\frac{2}{3}$	1	1	$\frac{2}{3}$	1	0.0219
r,s	$\frac{4}{27}$	$\frac{2}{3}$	1	1	$\frac{2}{3}$	1	1	$\frac{1}{3}$	1	1	0.0219
P,r	$\frac{4}{27}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	0.0008

Obliczenie reszty wartości pozostawiam czytelnikowi jako ćwiczenie, jednak żaden kandydat do połączenia nie uzyska wyniku wyższego niż 0.0219. Połączenie P z Ś lub

	n-gram		n-gram klasowy	
Liczność	1-gram	3-gram	1-gram	3-gram
>3	135 335	8 728 789	1000	6 917 746
>0	205 516	75 349 888	1000	26 913 330
wszystkie możliwe	260 741	$1.7 \cdot 10^{16}$	1000	10^9

Tablica 2.1: Porównanie liczby *n*-gramów na korpusie z ok. 366 milionami angielskich słów [1].

r z s jest tak samo dobrym możliwym pierwszym łączeniem (jest remis - arbitralnie decydujemy który łączymy).

■

Przeanalizowanie powyższego przykładu, mam nadzieję zbudowało pewne intuicje dotyczące tego jak algorytm działa od środka. Zwróć uwagę, że niezależnie od tego jakie grupy byśmy nie połączyli, w każdej iteracji otrzymujemy gorszą wartość funkcji celu. Dzieje się tak, bo model posiadający więcej grup ma więcej stopni swobody i jest w stanie lepiej zamodelować dane *uczące*. Jednak wartość funkcji celu na zbiorze testowym w początkowych iteracjach powinna rosnąć, a dopiero od pewnego momentu zacząć spadać.

2.4.2 Podsumowanie

Modele *n*-gramów klas, choć często działają lepiej niż modele *n*-gramowe, oferują raczej umiarkowaną poprawę nieokreśloności nad standardowymi modelami *n*-gramowymi – co pewnie jest częściowo skutkiem trudności z optymalizacją jego funkcji celu. Wymagają natomiast mniejszej liczby estymat prawdopodobieństwa i przez to mają niższe wymagania pamięciowe (patrz Tabela 2.1).

Niemniej jednak skutkiem ubocznym jego działania jest przypisanie słów w grupy według ich znaczeń, co może być przydatne w wielu zastosowaniach np. do konstrukcji cech. Przykładowe grupy możesz zaobserwować na rysunku 2.1. Nie trudno domyślić się dlaczego np. dni tygodnia zostały przypisane do jednej grupy: większość zdań po zamianie dnia tygodnia na inny nadal jest prawidłowa i podobnie prawdopodobna[4].

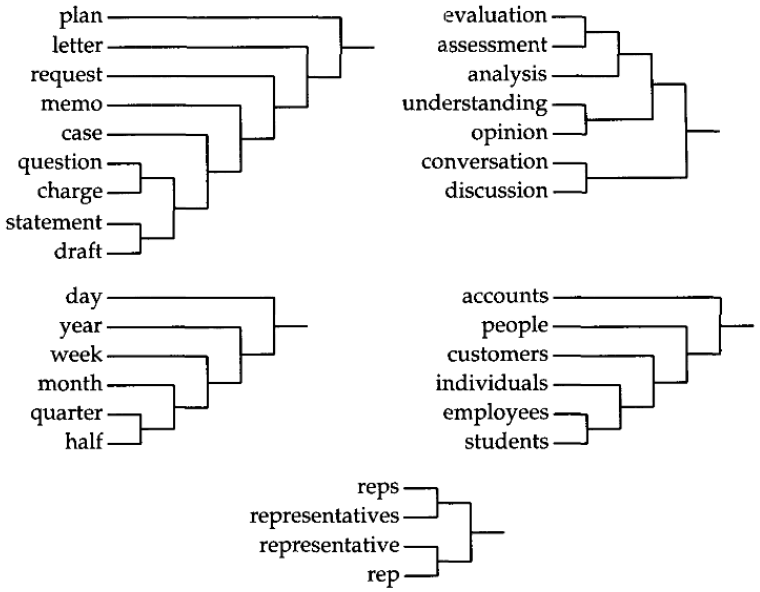
Co więcej, dendogram grupowania pozwala na zapis słowa w postaci ciągu binarnego, których porównywanie pozwala na oszacowanie podobieństwa semantycznego pomiędzy słowami. Konkretnie, możemy przeanalizować ścieżkę od korzenia dendogramu aż do wybranego słowa dopisując do jego reprezentacji 1 za każdym razem gdy musimy wybrać prawą odnogę i 0 gdy lewą. Powstaje w ten sposób ciąg zer i jedynek który określa pozycję słowa w dendogramie. Jak można zauważyć na rysunku 2.2, słowa które są blisko siebie w dendogramie są powiązane semantycznie. Zliczając długość wspólnego prefiksa opisanej zero-jedynkowej reprezentacji możemy określić podobieństwo pomiędzy słowami.

Eksploracja wyniku grupowania uzyskanego w czasie uczenia modelu klasowego stało się tak popularne, że często opisuję się *n*-gramowe modele klasowe jako grupowanie Browna, zapominając o stojącym za tym algorytmem modelem jezyka. Wyniki grupowania zostały zastosowanie m.in. w rozpoznawaniu encji nazwanych, parsowaniu zależnościowym czy segmentacji chińskich słów [5].

[4] Choć są wyjątki np. „Dobrze, że już piątek!”

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody
feet miles pounds degrees inches barrels tons acres meters bytes
director chief professor commissioner commander treasurer founder superintendent dean custodian
liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ
had hadn't hath would've could've should've must've might've
asking telling wondering instructing informing kidding reminding bothering thanking deposing
that tha theat
head body hands eyes voice arm seat eye hair mouth

Rysunek 2.1: Uzyskane grupy słów w modelu klasowym [1]



Rysunek 2.2: Poddrzewa dendogramu algorytmu grupującego Browna [1]

Dodatki

Materiały powtórkowe

Omówienie wygładzania modeli n-gramowych można znaleźć w rozdziale 3 książki [2], która jest dostępna za darmo w internecie `https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf`. Modele klasowe są dobrze opisane w pracy [3].

Materiały dla chętnych

Dla zainteresowanych polecam oryginalną pracę o modelach klasowych [1] wraz z wyjaśnieniem optymalizacji koniecznych do implementacji modeli klasowych, a także artykuł `https://thegradient.pub/understanding-evaluation-metrics-for-language-models/` o ewaluacji modeli języka.

Bibliografia

[1] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, i Robert L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992. URL `https://www.aclweb.org/anthology/J92-4003`.

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

[3] Percy Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[4] Maciej Ogrodniczuk i Łukasz Kobyliński, editors. *Proceedings of the PolEval 2018 Workshop*, Warsaw, Poland, 2018. Institute of Computer Science, Polish Academy of Sciences. ISBN 978-83-63159-27-6. URL `http://poleval.pl/files/poleval2018.pdf`.

[5] Lev Ratinov i Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W09-1119`.