contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

8. Modele Markowa

W poprzednim rozdziale przedstawiliśmy problem znakowania części mowy, a także poruszyliśmy szerszy problem predykcji struktur. W szczególności podkreśliliśmy potrzebę uproszczenia problemu poprzez zdekomponowanie go na mniejsze części. Różne modele robią to na różne sposoby – dzisiaj poznamy dwa z nich.

8.1 Ukryte modele Markowa

Rozpocznijmy od analizy ciekawego zdania zaczerpniętego z Wikipedii:

THE SAILOR DOGS THE HATCH.

które oznacza „Marynarz zamyka właz". Wyrazy SAILOR oraz THE zawsze są odpowiednio rzeczownikiem i określnikiem, więc główną trudnością tego zadania jest określenie części mowy wyrazu DOGS który najczęściej jest rzeczownikiem, ale może być też czasownikiem oraz wyrazu HATCH który może być i czasownikiem i rzeczownikiem. Określenie części mowy tego drugiego wyrazu nie jest trudne, gdyż przed nim stoi THE które nie pojawia się w zdaniu przed czasownikiem - więc musi być to rzeczownik. Analogicznie, wyraz DOGS raczej nie występuje tutaj jako rzeczownik, ponieważ po rzeczowniku raczej nie powinien stać kolejny rzeczownik, z kolei po rzeczowniku często stoi czasownik.

Prześledźmy jeszcze raz nasze rozumowanie: w jakim sposób my wykonaliśmy zadanie oznaczania częściami mowy? Wykorzystaliśmy dwa typy informacji:

- informację kontekstu części mowy – części mowy mają pewne zasady związane z występowaniem po sobie. Na przykład po określniku DET zwykle występuje rzeczownik NN lub przymiotnik JJ. Przymiotnik z kolei stawia się w angielskim zwykle przed rzeczownikiem, a nie po rzeczowniku.
- informację dot. konkretnego słowa – dane słowo ma tendencję do występowania w jakiejś części mowy częściej, a w innych w ogóle nie występuje np. THE tylko i wyłącznie występuje jako rodzajnik określony DET.

Skonstruujmy model który również zawiera w sobie 1) informację o przypisaniu słowa do części mowy i tendencji do jego występowania w tekście z częściej bądź rzadziej oraz 2) informację o tym że dana część mowy pojawia się po drugiej często, rzadko bądź w ogóle.

8.1.1 Model

Przedstawiony model będzie modelem generatywnym, a więc rozpoczniemy modelowanie od rozkładu łącznego zdania i sekwencji znaczników:

$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n)$$

gdzie y_i to etykiety (np. części mowy) a x_i to kolejne słowa w zdaniu. Zastosujmy regułę łańcuchową prawdopodobieństwa:

$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n) = P(x_1, x_2, ..., x_n | y_1, y_2, ..., y_n) P(y_1, y_2, ..., y_n)$$

Lewy czynnik wzoru to prawdopodobieństwo warunkowe otrzymania danych słów x_i po warunkiem że dane są nam części mowy y_i. Pomiędzy losowymi słowami w zdaniu występują oczywiście skomplikowane zależności, nawet jeśli mamy określone ich części mowy[1], jednak ich modelowanie raczej nie powinno nas interesować w tym zadaniu. Z kolei rozważając wpływ sekwencji części mowy na wygenerowanie słowa x_i (w końcu jest to prawdopodobieństwo warunkowe), raczej naturalne wydaje się uzależnienie generacji słowa x_i od i-tej części mowy w sekwencji. Upraszczając otrzymujemy więc:

$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i | y_i) P(y_1, y_2, ..., y_n)$$

Po prawej stronie pozostało nam prawdopodobieństwo całej sekwencji części mowy. Na szczęście, wcześniej radziliśmy sobie z prawdopodobieństwem sekwencjami wyrazów (modelem języka), więc możemy zastosować ten sam trik: wprowadzić założenie Markova np. pierwszego rzędu.

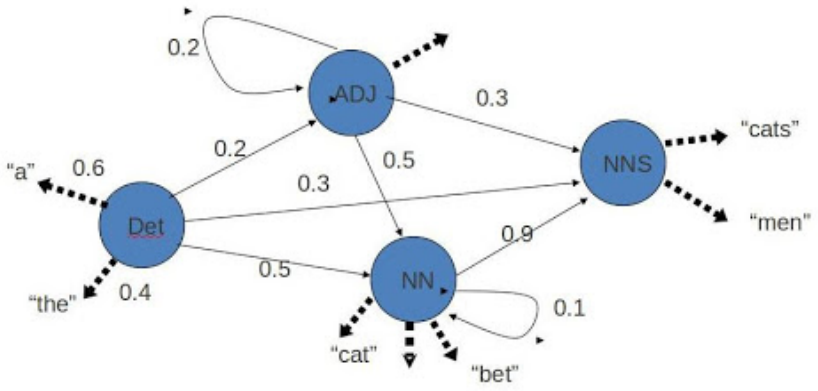
$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i | y_i) \prod_{i=2}^{n} P(y_i | y_{i-1}) P(y_1)$$

Zakładając, że sekwencja *tagów* zaczyna się od START możemy troszkę uprościć powyższy zapis.

$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i | y_i) \prod_{i=1}^{n} P(y_i | y_{i-1})$$

W przypadku modeli języka dodawaliśmy jeszcze token STOP , aby poradzić sobie ze zmienną długością zdania. Tutaj, o ile zależy nam tylko na predykcji tagów nie jest to konieczne: wykonujemy predykcję wśród sekwencji o znanej z góry długości, równej długości zdania. Modele dla różnych n nie muszą więc sumować się do 1 w celu wykonania predykcji. Jednakże dodanie tego tagu pozwala nam modelować fakt, że nie każda część mowy może znajdować się na końcu zdania np. na końcu zdania nie pojawi się rodzajnik DET. Dodając więc tag STOP nie tylko poprawiamy wyniki poprzez lepsze modelowanie

[1] P(x_1, x_2, x_3 | NN V NN) może wygenerować „Ala ma kota” lub „Hieny jedzą padlinę” jednak raczej nie powinno generować „Ala jedzą pedlinę”.



Rysunek 8.1: Ukryty Model Markowa jako graf stanów, które emitują obserwacje.

problemu, ale także pozwala nam używać modelu generatywnie (jeden model dla wielu długości). Mamy więc podwójne zwycięstwo: praktyczne i teoretyczne. Ostatecznie więc otrzymujemy:

$$P(y_1, y_2, ..., y_n, x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i|y_i) \prod_{i=1}^{n+1} P(y_i|y_{i-1})$$

Powyższy model to (bi-gramowy) ukryty model Markowa (ang. *Hidden Markov Model, HMM*). Rozszyfrujmy tę nazwę: „model Markowa" bo na sekwencji znaczników zastosowaliśmy założenie Markowa, jednak ta sekwencja jest dla nas „ukryta" - nieznana w momencie testowania.

Aby lepiej zrozumieć model, prześledźmy sposób w jaki generuje on dane. Generacje próbki rozpoczynamy od wylosowania części mowy y_1 pierwszego wyrazu w zdaniu z rozkładu prawdopodobieństwa $P(y_1|\boxed{\text{START}})$. Mając określoną część mowy pierwszego wyrazu możemy go wylosować z rozkładu $P(x_1|y_1)$. Następnie przechodzimy do generacji kolejnego wyrazu: określamy jego część mowy na podstawie jedynie części mowy poprzedniego słowa $P(y_2|y_1)$, oraz emitujemy słowo o danej części mowy – słowo to jest warunkowo niezależne od poprzedniego słowa. Proces działa tak długo aż z rozkładu $P(y_i|y_{i-1})$ wyemitujemy znacznik $\boxed{\text{STOP}}$.

Alternatywnie ukryty model Markowa możemy rozpatrywać w kategoriach przejść w grafie stanów. Każda część mowy jest wierzchołkiem-stanem w takim grafie. Możemy przejść z jednego stanu do drugiego podążając za prawdopodobieństwem tranzycji $P(y_i|y_{i-1})$, które skojarzone jest z krawędziami w grafie. Każdy stan emituje obserwacje zgodnie z prawdopodobieństwem emisji $P(x_1|y_1)$. W czasie generowania zdania, zaczynamy od specjalnego wierzchołka $\boxed{\text{START}}$ i probabilistyczne podążamy po krawędziach w grafie. Przykładowy taki graf znajduje się na rysunku 8.1.

Jak pewnie zauważyłeś używamy analogicznych nazw prawdopodobieństw jak tych w grupowaniu Browna oraz oba wzory są do siebie podobne. Różnicą pomiędzy poznanym grupowaniem Browna a HMM jest to że w omówionym modelu klasowym jedno słowo należało do tylko jednej klasy, podczas gdy tutaj jedno słowo może mieć kilka części mowy. Oczywiście, w naszym zastosowaniu znakowania części mowy będziemy używać HMM

w uczeniu nadzorowanym, ale jest możliwość nauki HMM w sposób nienadzorowany (algorytmem Bauma–Welcha[2]) czyli taki jak modeli klasowych. W takim ustawieniu stany y_i są ukryte nie tylko w czasie predykcji ale i nauki.

Ukryte modele Markowa mają dużo innych zastosowań niż tylko znakowanie części mowy. Używa się ich w auto-korektorach, w wykrywania pisma odręcznego oraz systemach rozpoznawania czy syntezy mowy. Mają one swoje liczne zastosowania poza inżynierią lingwistyczną np. w systemach nawigacji gdzie służą do korekcji odczytów sygnałów GPS (aby samochód jechał w nawigacji po drodze a nie po budynku).

Problem 8.1 Wymień założenia poczynione przez ukryty model Markowa do znakowania częściami mowy.

8.1.2 Trening

Trening nadzorowany bigramowego modelu HMM nie niesie ze sobą żadnych nowości: robimy to poprzez zliczanie. Istnieje też możliwość stosowania interpolacji estymat i innych metod wygładzania, które poznaliśmy w kontekście modeli języka.

■ **Przykład 8.1** Mając poniższy korpus uczący:
- Ala [N] ma [V] kota [N]
- Jacek [N] lubi [V] pluszowe [JJ] misie [N].

Policz prawdopodobieństwo sekwencji: Ala [N] lubi [V] misie [N].

$$\begin{aligned} P(x_1^n, y_1^n) &= \prod_{i=1}^{n} P(x_i|y_i) \prod_{i=1}^{n+1} P(y_i|y_{i-1}) \\ &= P(N|\boxed{\text{START}})P(V|N)P(N|V)P(\boxed{\text{STOP}}|N)P(Ala|N)P(lubi|V)P(misie|N) \\ &= 1 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{256} \end{aligned}$$

■

8.1.3 Predykcja

Mając nauczony model pora dokonać predykcji. W czasie wnioskowania mamy do dyspozycji zdanie czyli zmienne x_i do którego chcemy przypisać części mowy y_i. Szukamy więc rozwiązania do poniższego równania:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(x_1^n, y_1^n)$$

gdzie $\mathcal{Y}$ to zbiór wszystkich możliwych n-elementowych sekwencji znaczników, a oznaczenia typu x_i^j oznaczają sekwencję elementów x od i do j.

Powyższe równanie możemy uzyskać w standardowy sposób, poprzez przekształcenie wzoru na prawdopodobieństwo warunkowe.

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y_1^n|x_1^n) = \arg\max_{y \in \mathcal{Y}} \frac{P(x_1^n, y_1^n)}{P(x_1^n)} = \arg\max_{y \in \mathcal{Y}} P(x_1^n, y_1^n)$$

Druga równość wynika z definicji prawdopodobieństwa warunkowego, a trzecia z faktu że $P(x_1^n)$ jest stałe (nie zależy od argumentu funkcji maksimum).

[2] Algorytmu tego nie będziemy omawiać, jednak jest on specyficzną wersją algorytmu oczekiwanie-maksymalizacja, który zostanie przez nas omówiony w kontekście tłumaczenia maszynowego.

Jedną z możliwości ewaluacji powyższego wzoru jest zastosowanie wyszukiwania wyczerpującego (ang. *brute force*), które wymaga od nas przeiterowania po wszystkich możliwych n-elementowych sekwencjach części mowy. Choć taka metoda gwarantuje znalezienie optymalnego wyniku, nie jest stosowana w praktyce gdyż złożoność c^n nawet dla średniej długości zdania $n = 15$ wyrazów i stosunkowo małego zbioru znaczników $c = 20$ wymaga astronomicznej $20^{15} \approx 3.27 \cdot 10^{19}$ liczby iteracji.

Inną możliwością jest zastosowanie algorytmu zachłannego (ang. *greedy*). Model HMM można zapisać w poniższy sposób

$$P(x_1^n, y_1^n) = P(\boxed{\text{STOP}}|y_n) \prod_{i=1}^{n} P(x_i|y_i) P(y_i|y_{i-1})$$

który wymnaża prawdopodobieństwa związane z kolejnymi słowami. Możemy więc stworzyć wynikową sekwencję element po elemencie wybierając zawsze najbardziej prawdopodobny tag.

$$\hat{y}_1 = \arg\max_y P(x_1|y) P(y|x\boxed{\text{START}})$$

$$\hat{y}_2 = \arg\max_y P(x_2|y) P(y|\hat{y}_1)$$

$$\hat{y}_3 = \arg\max_y P(x_3|y) P(y|\hat{y}_2)$$

$$\ldots$$

$$\hat{y}_n = \arg\max_y P(x_n|y) P(y|\hat{y}_{n-1}) P(\boxed{\text{STOP}}|y)$$

Choć jest to metoda bardzo szybka i prosta, nie zwraca ona najbardziej prawdopodobnej sekwencji a tylko jej przybliżenie. Można pokazać, że strategia ta jest optymalna jeśli poczynimy dwa założenia: chcemy minimalizować błąd klasyfikacji na poszczególnych pozycjach i nie interesuje nas ocena sekwencji jako całości oraz nauczony model jest *prawdziwym* modelem. W problemie znakowania części mowy pierwsze założenie jest prawidłowe, nie możemy tego jednak powiedzieć o innych problemach jak np. wykrywanie jednostek nazewniczych. Założenie drugie jest z kolei zawsze błędne i w praktyce algorytm analizujący wszystkie możliwe sekwencje zwraca lepsze rezultaty, także wtedy gdy interesuje nas błąd klasyfikacji na poszczególnych pozycjach.

Problem 8.2 Udowodnij powyższą własność.

Powyższy zapis problemu dla modelu HMM sugeruje jednak, że można go zdekomponować na części a wyczerpująca iteracja po wszystkich sekwencjach wielokrotnie liczyłaby takie same fragmenty wzoru. Sugeruje to możliwość wykorzystania podejścia opartego na programowaniu dynamicznym, które pozwala nam na obliczenie optymalnego wyniku bez konieczności wykonania eksponencjalnej liczby iteracji.

Spróbujmy dalej przekształcić wzór predykcji modelu HMM, tak aby doprowadzić go do postaci w której rozdzielimy wyniki cząstkowe, które będziemy mogli ponownie wykorzystać w wielu obliczeniach. Dla uproszczenia dalszych rozważań załóżmy że tag $\boxed{\text{STOP}}$ jednocześnie zawsze generuje słowo $\boxed{\text{STOP}}$ – pozwala to nam traktować znacznik końca sekwencji w taki sam sposób jak inne znaczniki. Dalej zamiast obliczeniem $\arg\max_y$ będziemy zajmować się obliczeniem $\max_y$, a potem poprzez drobną modyfikację algorytmu wrócimy do rozwiązania oryginalnego problemu.

Wzór na prawdopodobieństwo najbardziej prawdopodobnej sekwencji możemy zapisać jako:

$$\max_{y_1,y_2,\ldots,y_n} \prod_{i=1}^{n} P(x_i|y_i)P(y_i|y_{i-1}) = \max_{y_1,y_2,\ldots,y_n} P(x_n|y_n)P(y_n|y_{n-1})\cdots P(x_2|y_2)P(y_2|y_1)P(x_1|y_1)P(y_1|y_0)$$

Zwróć uwagę, ze od y_1 zależą tylko trzy czynniki po prawej stronie wzoru.

$$\begin{aligned}&\max_{y_1,y_2,\ldots,y_n} \prod_{i=1}^{n} P(x_i|y_i)P(y_i|y_{i-1}) \\ &= \max_{y_2,\ldots,y_n} P(x_n|y_n)P(y_n|y_{n-1})\cdots P(x_2|y_2) \max_{y_1} P(y_2|y_1) \underbrace{P(x_1|y_1)P(y_1|y_0)}_{\text{zależy tylko od } y_1}\end{aligned}$$

Po tym przekształceniu pojawiła się mała część wzoru która jest zależna tylko od znacznika y_1, tokenu y_0 który zawsze równa się $\boxed{\text{START}}$ oraz x_1 które jest stałe. Pokreślony fragment wzoru ma więc tylko c możliwych wartości (po jednej dla każdego możliwego znacznika-części mowy), pomimo tego że jego wartość używana jest do obliczenia wykładniczej liczby sekwencji.

Możemy więc trochę przyśpieszyć obliczenia poprzez obliczenie tych wartości i zapisanie ich do pamięci podręcznej $B_1(y)$ (bufor zachowuje wartość dla każdego możliwej wartości y_1 – jest to tablica c-elementowa) :

$$B_1(y_1) = P(x_1|y_1)P(y_1|y_0) = P(x_1|y_1)P(y_1|\boxed{\text{START}})$$

Podstawiając do wzoru i zauważając że od y_2 zależy również tylko fragment wzoru otrzymujemy:

$$\begin{aligned}&\max_{y_1,y_2,\ldots,y_n} \prod_{i=1}^{n} P(x_i|y_i)P(y_i|y_{i-1}) \\ &= \max_{y_3,y_4,\ldots,y_n} P(x_n|y_n)P(y_n|y_{n-1})\cdots P(x_3|y_3) \max_{y_2} P(y_3|y_2) \underbrace{P(x_2|y_2) \max_{y_1} P(y_2|y_1)B_1(y_1)}_{\text{zależy tylko od } y_2 \text{ i bufora}}\end{aligned}$$

Znów, jeśli mamy obliczone wcześniej i zbuforowane wartości $B_1(y)$ to dostrzegamy fragment wzoru który zależy tylko od y_2. W podkreślonej części wzoru jest obecne również maksimum po poprzedniej etykiecie y_1 jednak wartość ta może być obliczona dla każdego możliwego podstawienia y_2. Dana wartość y_2 determinuje najlepszą możliwą etykietę y_1 i nie zależy od niczego innego. Kolejnym podkreślonym termem jest $P(x_2|y_2)$, które zależy tylko od y_2 bo x_2 jest stałe. Możemy więc obliczyć wszystkie możliwe wartości tego fragmentu i wykorzystać je w dalszych obliczeniach.

$$B_2(y_2) = P(x_2|y_2) \max_{y_1} P(y_2|y_1)B_1(y_1)$$

Takie przekształcanie wzoru możemy kontynuować, dochodząc do bufora dla y_n:

$$B_n(y_n) = P(x_n|y_n) \max_{y_{n-1}} P(y_n|y_{n-1})B_{n-1}(y_{n-1})$$

jednak mając taki bufor wystarczy po prostu odczytać z niego $\max_y B_n(y)$ aby otrzymać prawdopodobieństwo najbardziej prawdopodobnej sekwencji.

Powyżej opisany algorytm programowania dynamicznego to algorytm Viterbiego o złożoności $O(nc^2)$. Potrzebujemy obliczyć n buforów z których każdy ma c wartości, aby obliczyć wartość bufora musimy obliczyć wartość funkcji maksimum po c znacznikach. Aby odczytać najbardziej prawdopodobną sekwencję z algorytmu Viterbiego wystarczy jego drobna modyfikacja: należy dodać kolejny bufor, który będzie przechowywał znacznik który został wybrany podczas liczenia funkcji maksimum:

$$\overleftarrow{B}_i(y) = \cancel{P(x_i|y_i)} \arg\max_{y_{i-1}} P(y_i|y_{i-1}) B_{i-1}(y_{i-1})$$

z którego możemy odczytać najlepszy tag y_{i-1} jeżeli najlepszym tagiem dla y_i jest y. Po wykonaniu więc całego algorytmu uzupełniającego bufory $B_i(y)$ i $\overleftarrow{B}_i(y)$ kolejno od $i = 1$ do $i = n$, możemy określić ostatnią etykietę jako $y_n = \arg\max_y B_n(y)$, a poprzednią odczytać już z bufora $\overleftarrow{B}_n(y_n)$. Każdą kolejną etykietę odczytujemy w analogiczny sposób, iterując w tył.

8.1.4 Uwagi końcowe

Warto wspomnieć, że w praktyce najczęściej wykorzystuje się trzy-gramowe ukryte modele Markowa (ang. *trigram HMM*), które modelują sekwencję tagów prawdopodobieństwami tranzycji postaci $P(y_i|y_{i-1}, y_{i-2})$. W stosunku do przedstawionego wyżej modelu bigramowego oferują one delikatny wzrost trafności na zadaniu znakowania częściami mowy. Należy jednak zauważyć, ze złożoność algorytmu dekodującego (dokonującego predykcję, Viterbiego) rośnie. Potrzeba jest konstruowania buforów które zależą od dwóch znaczników zamiast jednego $B_n(y, y_{-1})$, a złożoność przeradza się w $O(nc^3)$.

Profesjonalne systemy znakujące częściami mowy oparte na modelach HMM, korzystają z różnych dodatkowy trików takie jak wygładzanie estymat (najczęściej interpolacją), uwzględnianiem morfologii słów (ich budowy: np. końcówki) itd. Korzysta się też z dobrodziejstw innych poznanych przez nas technik: używa się wyników grupowania Browna czy definiuje się specjalne pseudosłowa dla słów spoza słownika. Odpowiednio przygotowane systemy HMM uzyskują trafności dla języka angielskiego $\approx 97\%$ [2] podczas gdy najlepsze systemy oparte na głębokich sieciach rekurencyjnych oferują trafności ok. 1% wyższe. Podane wyniki są orientacyjne – zależą one zarówno od zbioru danych jak i zestawu znaczników.

Warto też wspomnieć, że algorytm Viterbiego nie jest jedynym możliwym algorytmem do dekodowania sekwencji w modelu HMM. Jak wspomnieliśmy, jedną z możliwych interpretacji modelu HMM jest graf stanów – problem dekodowania można sprowadzić do problemu szukania ścieżki w tym grafie i użyć np. algorytmu Dijkstry. Oferuje on taką samą złożoność w najgorszym przypadku i w praktyce działa minimalnie szybciej. Potencjalne zyski można zaobserwować przy problemach z bogatym zestawem znaczników. Zarówno algorytm Viterbiego jak i Dijkstry można przyśpieszyć poprzez ręczne ustalenie że pewne części mowy nie następują po sobie i pomijanie ich w krawędziach w grafie.

8.2 Modele Markowa o Maksymalnej Entropii

Powróćmy do przykładu THE SAILOR DOGS THE HATCH, który analizowaliśmy w sekcji 8.1. Choć opisane tam informacje wydają się wystarczać do rozszyfrowania części mowy w tym zdaniu, nietrudno zauważyć, że system w czasie predykcji mógłby użyć

także innych typów informacji. Na przykład, inną przesłanką, że DOGS jest czasownikiem która mogłaby być wykorzystana przez system jest to. że w zdaniu zawsze powinien być choć jeden czasownik. Żadne inne słowo w tym zdaniu nie może być czasownikiem, więc jest to bardzo silna przesłanka, że DOGS nim jest. Analogicznie po wyrazie DOGS stoi słówko THE, które również stanowi wskazówkę że DOGS nie jest rzeczownikiem.

Krótko mówiąc: chcielibyśmy dodać do klasyfikatora więcej cech. Jednak dodawanie cech w modelu HMM jest mocno ograniczone, bo cechy słowa musiałyby być modelowane wraz ze wszystkim zależnościami w $P(x_i|y_i)$ co jest zadaniem trudnym. Można pracować z założeniem o niezależności cech, ale możliwe wartościowe cechy np. suffix słowa czy fakt pisania go z wielkiej litery, są silnie zależne od samego słowa – założenie jest pogwałcone. Gdybyśmy jednak korzystali z prawdopodobieństwa $P(y_i|x_i)$ to nasz problem by się rozwiązał: wszystkie cechy są tam w części warunkowej (stałej), a rozkład modeluje tylko znacznik y_i. Prawdopodobieństwa tej postaci są wykorzystywane przez modele dyskryminacyjne.

Problem 8.3 Korzystając z wiedzy z uczenia maszynowego: jakie są wady i zalety korzystania z modeli dyskryminacyjnych i generatywnych?

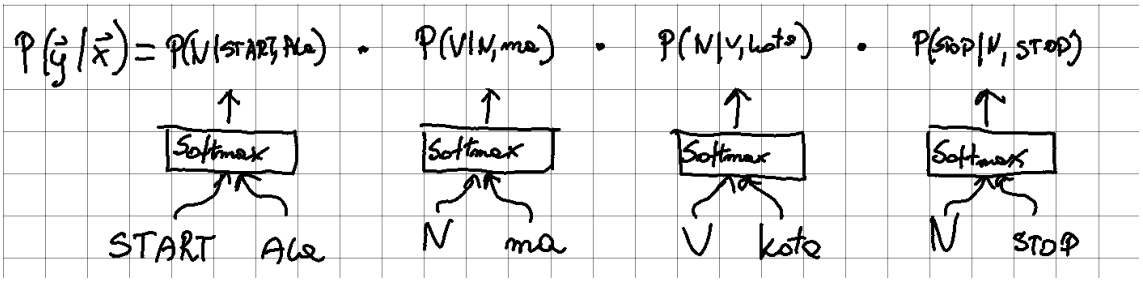
8.2.1 Model

Stwórzmy więc model dyskryminacyjny – rozpoczynamy przekształcanie wzoru na prawdopodobieństwo warunkowe. Prawdopodobieństwo to – w celu uzyskania uogólniania na dowolny rozmiar sekwencji – zdekomponujemy regułą łańcuchową i zastosujemy założenie Markowa.

$$\begin{aligned}
P(y_1^n|x_1^n) &= P(y_1, y_2, ..., y_n|x_1^n) \\
&= P(y_2, y_3, ..., y_n|y_1, x_1^n)P(y_1|x_1^n) \\
&= P(y_3, y_4, ..., y_n|y_1, y_2, x_1^n)P(y_2|y_1, x_1^n)P(y_1|x_1^n) \\
&= \cdots = \prod_{i=1}^{n} P(y_i|y_1^{i-1}, x_1^n) \\
&\approx \prod_{i=1}^{n} P(y_i|y_{i-1}, x_1^n)
\end{aligned}$$

Zwróć uwagę, że prawdopodobieństwo warunkowe kolejnego znacznika zależy w tym modelu od poprzedniego znacznika oraz *całego* wejścia. Jako estymator rozkładu $P(y_i|y_{i-1}, x_1^n)$ możesz wykorzystać w zasadzie dowolny klasyfikator, który zwraca prawdopodobieństwo np. drzewo decyzyjne, klasyfikator najbliższych sąsiadów czy las losowy. Zwróć uwagę, że aby obliczyć prawdopodobieństwo n-elementowej sekwencji musisz wykorzystać (ten sam) klasyfikator n razy – po jednym razie dla każdego elementu sekwencji (patrz ilustracja 8.2). Cechy tego klasyfikatora mogą zależeć od poprzedniego znacznika y_{i-1} oraz wszystkich słów w sekwencji x_1^n, w szczególności od słowa na pozycji i-tej. Przy każdym użyciu (ten sam) klasyfikator otrzymuje różne cechy na wejściu gdyż zmienia się y_{i-1} oraz słowo x_i od którego najczęściej zależy większość cech.

Najczęstszym wyborem jest zastosowanie w miejscu $P(y_i|y_{i-1}, x_1^n)$ wielomianowej regresji logistycznej (softmax), uzyskując model który nazywa się modelem Markowa o maksymalnej entropii (ang. *Maximum Entropy Markov Model, MEMM*). Jest to motywowane tym, że użycie właśnie modelu softmax prowadzi do interesujących właściwości



Rysunek 8.2: Obliczenie prawdopodobieństwa danej sekwencji tagów przy użyciu modelu MEMM dla zdania „Ala ma kota". Zwróć uwagę, że za każdym razem jest używany ten sam klasyfikator (softmax) z tym samym zestawem wag. Klasyfikator na wyjście zwraca cały zestaw prawdopodobieństw (dla każdego możliwego znacznika) – my odczytujemy jeden wybrany, zgodny z ocenianą sekwencją.

teoretycznych – uzyskania modelu o maksymalnej entropii, której nie będziemy tutaj analizować. Poza tym, regresja logistyczna jest naprawdę świetnym klasyfikatorem liniowym!

8.2.2 Softmax w nowej odsłonie

! Uwaga: w tej sekcji następuje zmiana notacji na ogólną (klasyczną w uczeniu maszynowym) tj. poprzez x będzie się rozumiało wejście klasyfikatora, a poprzez y jego odpowiedź. Porównując do poprzedniej notacji $P(y_i|y_{i-1}, x_1^n)$ cała część stojąca po prawej stronie warunku będzie oznaczana tutaj jako wejście do klasyfikatora x, a część przed warunkiem będzie oznaczana jako wyjście y.

Na tym etapie naszego kursu model softmax powinien być czytelnikowi dobrze znany i nie wymaga on dalszego omówienia. Wykorzystamy jednak ponowne pojawienie się softmax'a, aby zaprezentować alternatywny sposób jego zapisu, charakterystyczny dla szerszej gamy modelów logarytmiczno-liniowych często używanych w inżynierii lingwistycznej. Wprowadzona notacja nie tylko umożliwi lekturę innych źródeł o tych modelach, ale także będzie potrzebna w dalszej części kursu. Najczęściej używana notacja związana z wykorzystaniem softmaxa w sieciach neuronowych to:

$$\sigma(\mathsf{x})_y = \frac{e^{\mathsf{w_y}^T \phi(\mathsf{x})}}{\sum_{y'} e^{\mathsf{w_{y'}}^T \phi(\mathsf{x})}}$$

która zakłada istnienie tylu wektorów wag $\mathsf{w_i}$ ile jest klas oraz funkcję $\phi(\mathsf{x})$ która tworzy cechy na podstawie wejścia funkcji (często funkcja tożsamościowa). Alternatywna notacja, najczęściej stosowana w inżynierii lingwistycznej, zakłada istnienie tylko jednego wektora współczynników w oraz funkcję tworzącą cechy w zależności o wektora wejściowego i wybranej klasy $\phi(\mathsf{x}, y)$.

$$\sigma(\mathsf{x})_y = \frac{e^{\mathsf{w}^T \phi(\mathsf{x},y)}}{\sum_{y'} e^{\mathsf{w}^T \phi(\mathsf{x},y')}}$$

Zwróć uwagę, że przestawiona notacja jest uogólnieniem tej pierwszej. Mając kilka wektorów wag $\mathsf{w_1}, \mathsf{w_2}, \mathsf{w_3}$ – po jednej dla każdej klasy możemy skonkatenować je w jeden długi wektor $\mathsf{w} = [\mathsf{w_1}, \mathsf{w_2}, \mathsf{w_3}]$. Następnie funkcja cech musiałaby być zdefiniowana w następujący sposób:

$$\phi(\mathsf{x},y) = \begin{cases} [\phi(\mathsf{x}),0,0] & y=1 \\ [0,\phi(\mathsf{x}),0] & y=2 \\ [0,0,\phi(\mathsf{x})] & y=3 \end{cases}$$

Dla sprawdzenia obliczmy wartość licznika dla drugiej klasy:

$$e^{\mathsf{w}^T\phi(\mathsf{x},2)} = e^{[\mathsf{w}_1,\mathsf{w}_2,\mathsf{w}_3]^T[0,\phi(\mathsf{x}),0]} = e^{\mathsf{w}_2^T\phi(\mathsf{x})}$$

Zgodnie z oczekiwaniami otrzymaliśmy taką samą wartość jak przy używaniu notacji popularnej w głębokich sieciach neuronowych. Przy takim zastosowaniu nowej notacji rodzi się zarzut, że zwiększa ona wymagania pamięciowe i obliczeniowe poprzez tworzenie bardzo długich wektorów cech które z definicji są w sporej części wypełnione zerami. Oczywiście, jest to prawda, jednak w inżynierii lingwistycznej cechy są bardzo często rzadkie (czyli większość cech jest równa 0 np. one-hot-encoding) i stosuje się efektywne implementacje operacje na macierzach rzadkich dla których dłuższe wektory pełne zer nie stanowią większego wyzwania ani obliczeniowego ani pamięciowego.

Nowa notacja modelu umożliwia jednak większą swobodę w projektowaniu modelu i w rzeczywistości często prowadzi do ograniczenia rozmiaru modelu tj. liczby jego parametrów. Jedną z najbardziej popularnych cech przy znakowaniu jest po prostu tożsamość słowa x_i zakodowana w postaci wektora kodowania „1 z n". Mając słownik V i zbiór tagów/klas o liczności c uzyskujemy w ten sposób $c|V|$ cech i tyleż samo parametrów w „nowej" notacji oraz $|V|$ cech i $c|V|$ parametrów w starej. Jednak pewne słowa jak np. THE, A, AN mają określoną część mowy[3] i co więcej stanowią one zbiór zamknięty – żadne inne słowo w angielskim nie jest rodzajnikiem. Co oznacza, że przewidując klasę „rodzajnik" nie musimy kłopotać się tożsamością różnych innych słów: do jej przewidzenia wystarczy nam cecha $[\![x_i \in \{\text{THE}, \text{A}, \text{AN}\}]\!]$ oraz $[\![x_i \notin \{\text{THE}, \text{A}, \text{AN}\}]\!]$ które powinny całkowicie wystarczyć. Dzięki nowej notacji klasa „rodzajnik" może mieć tylko 2 cechy podczas gdy wszystkie inne klasy nadal mają $|V|$ cech.

Wracając do modelu MEMM, używając nowej notacji możemy go ostatecznie zapisać jako:

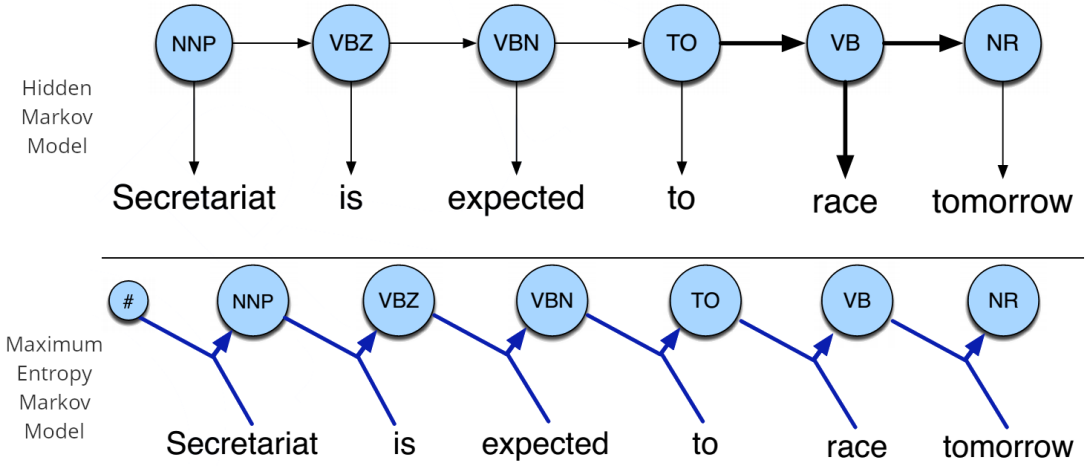
$$P(y_1^n|x_1^n) = \prod_{i=1}^{n} P(y_i|y_{i-1},x_1^n) = \prod_{i=1}^{n} \frac{e^{\mathsf{w}^T\phi(\mathsf{x},y_{i-1},y_i)}}{\sum_{y'} e^{\mathsf{w}^T\phi(\mathsf{x},y_{i-1},y')}} = \prod_{i=1}^{n} \frac{e^{\mathsf{w}^T\phi(\mathsf{x},y_{i-1},y_i)}}{Z(\mathsf{x},y_{i-1})}$$

gdzie $Z(\mathsf{x},y_{i-1}) = \sum_{y'} e^{\mathsf{w}^T\phi(\mathsf{x},y_{i-1},y')}$ nazywamy sumą statystyczną i jest to stała, która normalizuje decyzję klasyfikatora dla każdej z etykiet do jedynki. Funkcja cech ma trzy argumenty $\phi(\mathsf{x},y_{i-1},y_i)$ gdzie x,y_{i-1} to wejście do klasyfikatora a y_i to zmienna decyzyjna (znacznik), który staramy się przewidzieć.

8.2.3 Inżynieria cech dla wykrywania encji nazwanych

Poznajmy kilka standardowych cech, których używa się do konstrukcji systemów NER. Oczywiście, najprostszą i najpopularniejszą cechą jest po prostu samo słowo zakodowane w postaci one-hot-encoding. W naszej notacji aby stworzyć taką cechę dla słowa „Ala"

[3] Dla ścisłości A może także występować jako litera a nie jako rodzajnik



Rysunek 8.3: Porównanie Ukrytego Modelu Markova z (warunkowym) Modelem Markova o Maksymalnej Entropii. Przy założeniu, że te ostatni korzysta tylko z i-tego elementu wejścia (może korzystać ze wszystkich).[2]

trzeba w rzeczywistości wstawić całą serię cech, po jednej dla każdej możliwej klasy:

$$\phi_1(\mathrm{x}, y_{i-1}, y_i) = \begin{cases} 1 & x_i = \text{„Ala”} \wedge y_i = \text{PER} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

$$\phi_2(\mathrm{x}, y_{i-1}, y_i) = \begin{cases} 1 & x_i = \text{„Ala”} \wedge y_i = \text{O} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

$$\phi_3(\mathrm{x}, y_{i-1}, y_i) = \begin{cases} 1 & x_i = \text{„Ala”} \wedge y_i = \text{LOC} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

i tak dalej. W literaturze często spotkasz się z ogólnymi definicjami takich cech np.

$$\text{Dla każdego } w_k \in V \wedge y \in Y \text{ skonstruuj } \phi_k(\mathrm{x}, y_{i-1}, y_i) = \begin{cases} 1 & x_i = w_k \wedge y_i = y \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Inne popularne cechy to prefiksy oraz sufiksy słowa o różnych długościach oraz cechy tzw. „kształtu słowa”. Tworzy się je zastępując wszystkie duże litery na literę „X”, wszystkie małe litery na „x”, cyfry zastępuje się „0” a potencjalne znaki interpunkcyjne pozostawia się bez zmian. I tak „USA” czy „FBI” zostaną zamienione na „XXX” a „Ostrów Wielkopolski” na „Xxxxxx Xxxxxxxxxxx”. Często stosuje się też spłaszczone kształty słów tj. każdy powtarzający się znak w kształcie słowa jest eliminowany czyli „USA” zostanie przekształcone na „X” ale już „U.S.A.” pozostanie „X.X.X.”. Równie popularną cechą jest wykryta część mowy znakowanego słowa.

Każda z powyższych cech używana jest w wersji unigramowej, bigramowej czy trzygramowej (trzygramy części mowy, trzygramy kształtów słów itd.). Często wykorzystuje się zaletę korzystania z modeli dyskryminacyjnych jaką jest warunkowanie na całej sekwencji-zdaniu i używa się trzygramu patrzącego w przód i w tył tj. $w_{i-1}w_iw_{i+1}$. Klasycznym

przykładem motywującym cechy wybiegające w przód jest tytuł gazety „New York Times" – dopiero po spojrzeniu dwa słowa dalej wiemy czy „New" zaczyna nazwę lokalizacji czy gazety. Rzadko jednak spotyka się cechy zależące rzeczywiście od dalekich słów lub budowy całego zdania, a nie tylko od słów przylegających do w_i.

Innym popularnym typem cechy jest obecność słowa na liście miast, państw, firm, tytułów filmów itd. które można trudniej lub prościej skolekcjonować z zasobów Internetu. Takie cechy nazywane są po angielsku „gazeteers".

W końcu, tworzy się też cechy uwzględniające poprzednio przyznany tag y_{i-1}. Cecha znakująca połączenie poprzedniego tagu z obecnym słowem lub z częścią mowy poprzedniego i obecnego słowa albo nawet połączenie tagu z kształtem słów pozwalają często podnieść wyniki systemu.

Jak widzisz, liczba cech w systemie NER nierzadko idzie w miliony, jednak duża liczba kombinacji cech ze znacznikiem w ogóle nie występuje w zbiorze uczącym przez co estymacja powiązanych z nimi parametrów mija się z celem. Również cechy które rzadko występują w zbiorze uczącym będą miały zaszumione estymaty. Z powyższych powodów zwykle stosuje się przycięcie (ang. *prunning*) cech do tylko tych które występują częściej niż 5 razy w zbiorze uczącym. (Pięć jest oczywiście magiczną liczbą, którą można stroić).

Kwestię inżynierii cech trzeba zakończyć przestrogą: cechy dobre dla wykrywania jednostek nazewniczych będącymi lokalizacjami geograficznymi, osobami i organizacjami nie koniecznie są dobre dla zadania wykrywania genów i protein w tekstach biomedycznych. Stworzenie dobrego systemu NER dla wybranej domeny zastosowania będzie wymagało kreatywności i sporej liczby testów do ewaluacji nowych cech na zasadzie prób i błędów, jednak opisane powyżej cechy możesz potraktować jako dobry punkt wyjścia do dalszych działań.

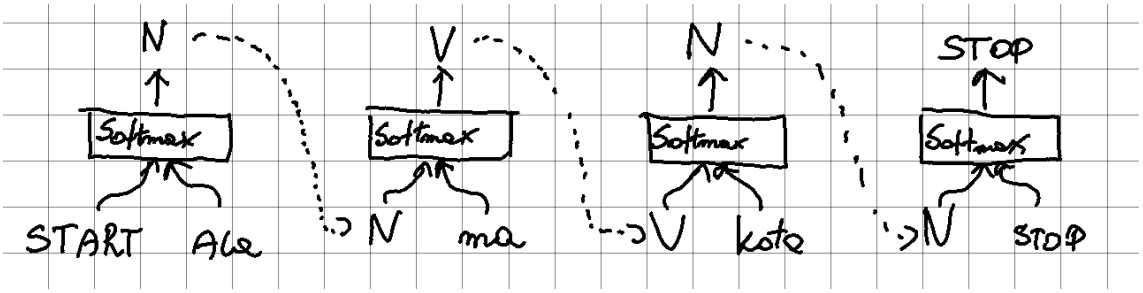
8.2.4 Trening i wnioskowanie

Do wykonania predykcji modele MEMM możemy użyć algorytmu zachłannego, który zawsze wybiera najbardziej prawdopodobną etykietę. Rozpoczyna się od wykonania predykcji dla ($\boxed{\text{START}}$, x_1), w ten sposób otrzymując wartość pierwszego znacznika $\hat{y}_1$. Następnie uruchamiamy klasyfikator ponownie dla drugiego słowa x_2, a jako poprzedni znacznik używa się wcześniej przewidzianego $\hat{y}_1$ itd. Cały proces został zilustrowany na rysunku 8.4. Zwróć uwagę, że pomimo tego że na rysunku widnieje n klasyfikatorów to każdy z narysowanych klasyfikatorów jest *tym samym* klasyfikatorem (te same wagi), tylko użytym dla różnych wejść. Taki sposób wizualizacji klasyfikatorów dla sekwencji poprzez wielokrotne narysowanie tego samego klasyfikatora nazywamy czasami rozwijaniem klasyfikatora w sekwencję.

Inną możliwością wykonania predykcji, pozwalającą na uzyskanie najbardziej prawdopodobnej sekwencji jest algorytm Viterbiego, który wymaga drobnego dostosowania w stosunku do jego wersji z rozdziału 8.1.3. To dostosowanie pozostawiam czytelnikowi do samodzielnego rozwiązania[4].

Dla treningu modelu MEMM stosujemy standardowe techniki dla uczenia wielomianowej regresji logistycznej. Jedynym elementem, który pokrótce omówię to przygotowanie zbioru uczącego. Zwróć uwagę, że model przewiduje każdy tag z osobna, więc sekwencja n-elementowa to n przykładów uczących. W zasadzie $n+1$ jeśli uwzględnimy $\boxed{\text{STOP}}$. Mając przykładową sekwencję „Ala [N] ma [V] kota [N]" przekłada się ona na przykłady uczące:

[4] A znając mnie: przy okazji zapytam o to na kolokwium



Rysunek 8.4: Wykonanie zachłannej predykcji sekwencji znaczników modelem MEMM dla zdania „Ala ma kota". Wykorzystanie klasyfikatora po raz czwarty jest zbędne – wiemy z definicji, że ostatni tag to [STOP] i powinniśmy go wstawić nawet jeśli klasyfikator popełniłby błąd i nie przewidziałby go prawidłowo. Czwarta, zbędna, klasyfikacja została uwzględniona na rysunku dla spójności z innymi rysunkami. Zwróć uwagę, że za każdym razem jest używany ten sam klasyfikator (softmax) z tym samym zestawem wag oraz że wynik klasyfikacji dla poprzedniego słowa wchodzi na wejście klasyfikatora dla następnego słowa.

Wejście (na podstawie którego konstruujemy cechy)	Zmienna decyzyjna
[START] , Ala	N
N , ma	V
V , kota	N
N , [STOP]	[STOP]

Choć w tabelce, dla uproszczenia, w pierwszej kolumnie umieszczaliśmy tylko aktualnie rozważane słowo x_i jako reprezentanta sekwencji x_1^n na którym możemy konstruować cechy i które ma szczególne znaczenie dla danej decyzji, to przypominam że cechy mogą być konstruowane w oparciu o całą sekwencję wejściową bo jest ona znana z góry.

8.2.5 Problem obciążenia etykietą

Modele MEMM, choć ogólnie nieźle sobie radzą w praktyce, są całkowicie bezradne w pewnych specyficznych sytuacjach na wskutek tzw. problemu obciążenia etykietą (ang. *label bias problem*). Polega on na tym, że znacznik po którym występuje tylko jedna możliwa tranzycja do innego znacznika całkowicie ignoruje w przypisywaniu prawdopodobieństwa obserwację tj. słowo x_i. Bezpośrednim powodem występowania tego problemu jest rozbicie prawdopodobieństwa sekwencji na mnożenie prawdopodobieństw które dla danego znacznika y_i sumują się do jedynki. Następuje więc lokalna normalizacja prawdopodobieństwa po każdym znaczniku – skoro więc po danym znaczniku musi występować inny konkretny znacznik (tylko jedna możliwa tranzycja) to zostanie on zawsze przewidziany z prawdopodobieństwem 1 całkowicie ignorując to czy dane słowo x_i pasuje do tego znacznika czy nie.

Problem jest niestety trochę bardziej ogólny: znaczniki z niską entropią rozkładów następnych znaczników będą czerpały mało informacji z obserwacji/słów i podążały za wiedzą związaną z przejściami etykiet [3]. Używając prostszego języka: problem pojawia się gdy mamy do czynienia ze stanem z małą liczbą możliwych następników lub gdy „efektywna" liczba następników jest mała tj. wiele następników jest możliwych, ale rozkład prawdopodobieństwa przejść między nimi jest skoncentrowany na małej jej liczbie.

Aby lepiej zilustrować problem przeanalizujmy kilka przykładowych sytuacji. Pierwszy przykład zaczerpnięty z [4] pozwoli nam na lepsze zrozumienie powiązania problemu z niską entropią rozkładu przejść. Rozważmy graf przedstawiony na rysunku 8.5 gdzie mamy rozpisany przykład składających się z czterech słów (obserwacji) którym należy przyporządkowywać jedną z pięciu etykiet (stanów). Analiza prawdopodobieństw przejść pomiędzy etykietami, zapisanymi przy krawędziach grafu, pozwala zauważyć, że Stan 1 preferuje zamienianie na Stan 2, z kolei Stan 2 zawsze przypisuje jedno z najwyższych prawdopodobieństw pozostaniem w tym stanie. Co więcej, stan ten modeluje przejścia do stanów, które są ślepymi zaułkami – nie ma z nich krawędzi pozwalających na kontynuacje znakowania sekwencji, patrząc na graf jest więc jasne że przejścia te nie powinny być rozpatrywane. Jednakże szybkie przeliczenie prawdopodobieństw sekwencji przejść:

$$P(1,1,1,1) = 0.4 \cdot 0.45 \cdot 0.5 = 0.09 \qquad P(2,2,2,2) = 0.018 \qquad P(1,2,2,2) = 0.054$$

nie pozostawia złudzeń: najbardziej prawdopodobną sekwencją jest pozostanie cały czas w stanie 1, pomimo tego że stan ten preferuje przejście do stanu 2! Problem polega na tym, że stan 2 ma dużo możliwych krawędzi wyjściowych, a konieczność lokalnej normalizacji powoduje że musi on rozdzielić na nie prawdopodobieństwa sumujące się do 1. Takie lokalne prawdopodobieństwo nie patrzy też w przód sekwencji – nie możemy wiedzieć, że w tej sekwencji po stanach 3, 4, 5 nie ma możliwości kontynuacji, a mechanizm predykcji po prostu szuka sekwencji z maksymalnym prawdopodobieństwem.

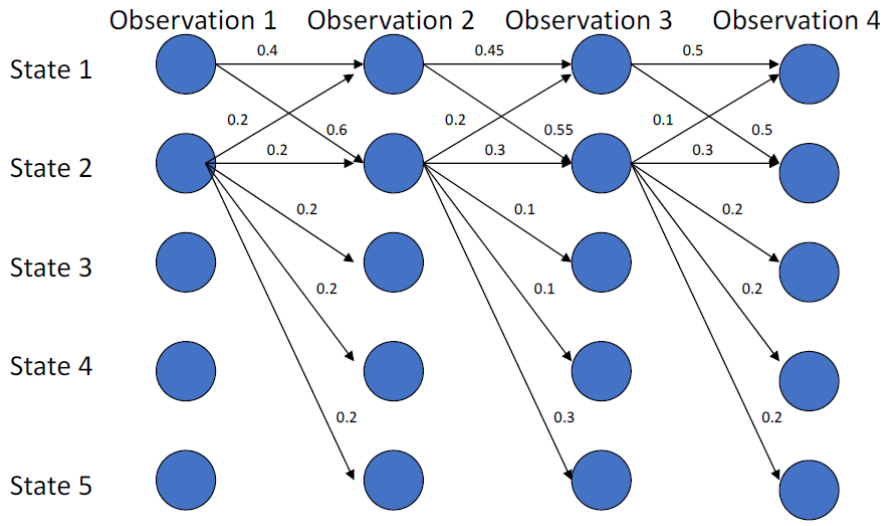
Zauważ, że jeśli algorytm dojdzie do stanu w którym model całkowicie nie wie jaki znacznik jest dalej bo np. taka sytuacja rzadko pojawiała się w zbiorze uczącym, model nie jest w stanie poinformować algorytmu konstruującego końcową sekwencję (np. Viterbiego) że nie powinno się specjalnie przejmować jego predykcją. Model może przekazać informację o swojej niepewności poprzez rozdzielenie prawdopodobieństwa po równo na wychodzące krawędzie (znaczniki). Mając dwie takie sytuacje w sekwencji: jedną dla znacznika z 2 możliwymi następnikami oraz dla znacznika z 10 możliwymi następnikami, mechanizm predykcyjny zacznie silnie preferować pierwszy znacznik. Pomimo tego, że predykcje dla obu znaczników są tak samo bezużyteczne (rozkład jednorodny).

Drugi przykład zaczerpnięty z [1], dotyczy wykrywania jednostek nazewniczych z tagowaniem BIO. Mamy dostępne dwa typy bytów nazwanych: osoba PER i lokalizacja LOC (używamy więc 5 znaczników: B-LOC, I-LOC, B-PER, I-PER, O) oraz korpus:

- Harvey Ford – występujący 9 razy jako PER i raz jako LOC
- Harvey Park – występujący 9 razy jako LOC i raz jako PER
- Myrtle Ford – występujący 9 razy jako PER i raz jako LOC
- Myrtle Park – występujący 9 razy jako LOC i raz jako PER

Zwróć uwagę, że bardzo silnym predyktorem typu jednostki nazewniczej jest jej drugie słowo np. „* Park" na 90% jest lokalizacją. Zakładając, że token START ma klasę O oraz że klasyfikator używa jako cechy po prostu słowo x_i oraz poprzednią etykietę y_{i-1} wyznaczmy kilka przykładowych wartości prawdopodobieństwa zwracanych przez klasyfikator.

- P(B-PER| O, Harvey) = $\frac{10}{20} = 0.5$
- P(B-LOC| O, Harvey) = $\frac{10}{20} = 0.5$
- P(I-PER| B-PER, Ford) = $\frac{18}{18} = 1$
- P(I-LOC| B-LOC, Park) = $\frac{18}{18} = 1$
- P(I-PER| B-PER, Park) = $\frac{2}{2} = 1$
- P(I-LOC| B-LOC, Ford) = $\frac{2}{2} = 1$



Rysunek 8.5: Przykładowy graf przejść między stanami-znacznikami i obserwacjami-słowami wraz z określonymi prawdopodobieństwami. [4]

Zauważ, że moc predykcyjna drugiego słowa została całkowicie zatracona! Klasyfikator wie, że po B-PER/B-LOC następuje I-PER/I-LOC i całkowicie ignoruje wartość obserwacji (słowa). Aby podkreślić dramat sytuacji, obliczmy prawdopodobieństwo że „Harvey Ford" jest osobą wg. modelu MEMM:

$$P(\text{B-PER I-PER}|\text{ Harvey Ford}) = P(\text{B-PER}|\text{O}, \text{Harvey})P(\text{I-PER}|\text{B-PER}, \text{Ford}) = \frac{1}{2}$$

Pomimo tego, że w zbiorze uczącym „Harvey Ford" w dominującej liczbie przykładów był znakowany jako nazwa osoby, model MEMM w czasie predykcji po prostu rzuca monetą!

Rozwiązaniem tego problemu zajmiemy się na kolejnych zajęciach.

Dodatki

Materiały powtórkowe

Opis modeli HMM i MEMM można znaleźć w książce [2].

Materiały dla chętnych

Dla zainteresowanych polecam szczegółowy opis problemu label bias dostępny pod tym linkiem: `https://awni.github.io/label-bias/`.

Bibliografia

[1] Ralph Grishman. Label bias (natural language processing course), 2016. URL `https://cs.nyu.edu/courses/spring16/CSCI-GA.2590-001/index.html`.

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

[3] John D. Lafferty, Andrew McCallum, i Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, 2001.

[4] Ramesh Nallapati. Conditional random fields (probablistic graphical models course). URL `http://www.cs.stanford.edu/~nmramesh/crf`.