مدل مارکوف پنهان

مدل‌های مارکوف پنهان[39] که به اختصار به آن (HMM) می‌گویند کاربردهای فراوانی به خصوص در بیوانفورماتیک[Koski T]، تشخیص گفتار[Rabiner L] و بسیاری از حوزه‌های دیگر به کار رود[Mc]. در یک HMM دو نوع حالت وجود دارد: حالت‌های قابل مشاهده[40] و حالت‌های پنهان[41]. هیچ تناظر یک به یکی بین حالت‌های قابل‌مشاهده و حالت‌های پنهان وجود ندارد؛ از همین رو ممکن نیست که صرفا از روی مشاهده حالت‌های قابل مشاهده بفهمیم که در چه حالت پنهانی قرار داریم. یک HMM معمولا با عناصر پیش‌رو توصیف می‌گردد[Rabiner L]:

- تعداد حالت‌های پنهان در مدل که آن را با N نشان می‌دهیم. با وجود این که حالات پنهان هستند، در بسیاری از برنامه‌های کاربردی معمولا اهمیت فیزیکی خاصی به حالات پنهان قائل هستیم. حالت‌های منفرد را به شکل زیر نشان می‌دهیم:
$$S=\{s_1, s_2, \dots, s_N\}$$
و حالتی را که اندازه t را دارد با Q_t نمایش می‌دهیم.

- حالت‌های مشاهده متمایز از هم به ازای هر حالت پنهان را با M نمایش می‌دهیم. نمادهای مشاهده متناظر با خروجی فیزیکی سیستمی که مدل‌سازی کرده‌ایم است. به عنوان مثال، «محصول تاییدشده» یا «محصول تاییدنشده» دو حالت مشاهده در یک فرآیند تولید می‌باشند. ما نمادهای منفرد را به شکل زیر نشان می‌دهیم:
$$V=\{v_1, v_2, \dots, v_M\}$$
و نمادی را که اندازه t دارد را با O_t نشان می‌دهیم.

- توزیع احتمال گذارحالات را با $[A]_{ij}=\{a_{ij}\}$ نشان می‌دهیم که در آن
$$a_{ij}=P(Q_{(t+1)}=s_i \mid Q_t=s_j), i\le i, j\le N$$

- توزیع احتمال نماد مشاهده‌شده در حالت پنهان j که با $[A]_{ij}=\{b_j(v_k)\}$ نشان می‌دهیم که در آن
$$b_j(v_k)=P(O_t=v_k \mid Q_t=s_j), 1\le j\le N, 1\le k\le M$$

- توزیع احتمال اولیه $\Pi=\{\pi_i\}$ که در آن
$$\pi_i=P(Q_1=s_i), 1\le i\le N$$

با داشتن مقادیر مناسب M, A, B, Π و N مدل پنهان مارکوف ما می‌تواند در جهت تولید سری مشاهده $O=\{O_1 O_2 \dots O_T\}$ به کار رود که در آن T تعداد مشاهدات در سری است. برای سادگی ما از نمادگذاری فشرده $\Lambda=(A, B, \Pi)$ برای نشان دادن پارامترهای یک HMM استفاده خواهیم کرد. طبق تعاریف بالا، توزیع احتمال گذار مرتبه اول برای حالت‌های پنهان به کار می‌رود.

سه مسئله مهم و کلاسیک در HMM وجود دارد:

---
[39] Hidden Markov Model
[40] Observable states
[41] Hidden states

- با داشتن سری مشاهدات $O=\{O_1 O_2 ... O_T\}$ و یک HMM چگونه می‌توان به صورت کارآمد احتمال آن سری مشاهدات را حساب کرد؟
- با داشتن سری مشاهدات $O=\{O_1 O_2 ... O_T\}$ و یک HMM چگونه می‌توان یک سری حالات متناظر $Q=\{Q_1 Q_2 ... O_T\}$ را به صورت بهینه انتخاب کرد؟
- با داشتن سری مشاهدات $O=\{O_1 O_2 ... O_T\}$ ، چگونه می‌توان پارامترهای یک HMM را انتخاب کرد؟

برای مسئله اول یک الگوریتم پویای پیش‌رو-پس‌رو برای محاسبه احتمال سری مشاهدات به صورت کارآمد پیش‌نهاد شده است[11].

برای مسئله دوم می‌بایست تلاش کنیم تا جنبه پنهان مدل را کشف کنیم یا به عبارتی دیگر حالت‌های «درست» را بیابیم. در بسیاری از مسائل کاربردی ما از یک معیار سنجش بهینه برای حل مسئله به بهترین شکل ممکن استفاده می‌کنیم. پراستفاده‌ترین معیار این است که بهترین سری حالات را که درست‌نمایی $P(Q \mid \Lambda, O)$ را بیشینه می‌کند بیابیم. این برابر است با بیشینه ساختن $(P(Q, O \mid \Lambda))$ چرا که

$$P(Q \mid \Lambda, O)=\frac{(P(Q, O \mid \Lambda))}{P(O \mid \Lambda)}$$

الگوریتم ویتربی[42] یک تکنیک برنامه‌نویسی پویا برای پیدا کردن این بهترین سری حالات $Q=\{Q_1 Q_2 ... O_T\}$ به ازای سری مشاهدات $O=\{O_1 O_2 ... O_T\}$ است[204].

برای مسئله سوم، ما تلاش خواهیم کرد که پارامتر Λ را چنان بیابیم که $P(O \mid \Lambda)$ را به کمک الگوریتم حداکثرسازی امید ریاضی[43] بیشینه سازیم.

چند مثال از مدل مارکوف پنهان

برای روشن‌سازی منظورمان از پارامترهای یک HMM بهتر است چند مثال را بررسی کنیم.

آب و هوا

از مثال‌های کلاسیک و پرکاربرد برای توضیح یک HMM به تصویر کشیدن آب و هوا است . با توجه به نمادگذاری و تعاریف بالا، برای مدل‌سازی یک سیستم ساده آب‌وهوا داریم:

- حالات پنهان:

$s_1=sunny, s_2=rainy$

[42]Viterbi Alogortihm
[43]Expectation Maximization یا EM

- حالات مشاهده:
$v_1 = Dry, V_2 = Wet$

- توزیع احتمال گذار حالت:
$A = \begin{bmatrix} a_{11} & a_{22} \\ a_{21} & a_{22} \end{bmatrix}$ که در آن: $a_{11} = 0.7, a_{12} = 0.3, a_{21} = 0.4, a_{22} = 0.6$

- توزیع احتمال نماد مشاهده‌شده:
$B = \begin{bmatrix} b_1(Dry) & b_1(wet) \\ b_2(Dry) & b_2(wet) \end{bmatrix}$ برای مثال
$b_1(Dry) = 0.8, b_1(wet) = 0.2, b_2(Dry) = 0.3. b_2(wet) = 0.7$ این بدین معناست که در مدل ما اگر در حالت Sunny (آفتابی) باشیم به احتمال 0.8 زمین خشک است و به احتمال 0.2 زمین خیس است.

- توزیع احتمال اولیه $\Pi = \{\pi_i\}$ که در آن قرار می‌دهیم: $\pi_1 = 0.6, \pi_2 = 0.4$ این بدین معناست که به احتمال 0.6 سیستم از حالت Sunny آغاز به کار می‌کند و به احتمال 0.4 از حالت Rainy.

برچسب‌گذاری ادات سخن[44]

از مهم‌ترین کاربردهای HMM، برچسب‌گذاری ادات سخن است که به اختصار به آن POS tagging یا POS می‌گویند.در POS ما به هر واژه در یک جمله با توجه به نقش ادایی و معنایی‌اش در جمله، یک برچسب‌گذاری دستوری می‌چسبانیم. به عنوان مثال به جمله زیر دقت کنید:

من شب‌ها غذا می‌پزم.

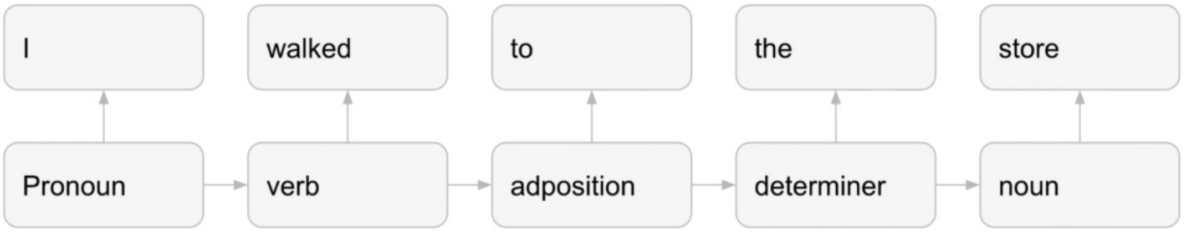
در این جمله نقش دستوری «من» فاعل، نقش دستوری «شب‌ها» قید زمان، نقش دستوری «غذا» مفعول و نقش دستوری «می‌پزم» فعل است. POS کاربردهای بسیار گسترده‌ای در پردازش زبان‌های طبیعی[45] دارد و معمولا جز اولین مراحل پردازش جمله است. کاربرد دیگر POS در ابهام‌زدایی و دریافت معنا از یک جمله است؛ به عنوان مثال واژه Drink در زبان انگلیسی هم به شکل فعل با معنای نوشیدن و هم به شکل اسم به معنای نوشیدنی استفاده می‌شود. اگر قرار است یک هوش مصنوعی متوجه منظور کاربر شود یا جمله‌ای را تولید کند می‌بایست نقش دستوری و معنایی Drink را از جمله استخراج کند و در جای درست خود به کار ببرد. بدین منظور از POS استفاده می‌شود کاربرد دیگر POS در مشخص‌سازی مضمون و محتوای متون است..

ما می‌توانیم POS را به کمک HMM مدل‌سازی کنیم. به عنوان مثال برای پارامترهای آن داریم:

- حالات مشاهده: حالات مشاهده در یک POS همان واژگانی است که به صورت عیان با آن سروکار داریم. طبق مثال قبل حالات مشاهده برابر با «من»،«شب‌ها»،«غذا»،«می‌پزم».
- حالات پنهان: همان حالات دستوری و ادایی در گرامر یک زبان طبیعی است. برای مثال قبل حالات پنهان برابر است با: فاعل، قید زمان، مفعول، فعل.

در پایین یک مثال به زبان انگلیسی آورده شده است.:

[44] Part-of-speech tagging
[45] Natural Language Processing یا NLP

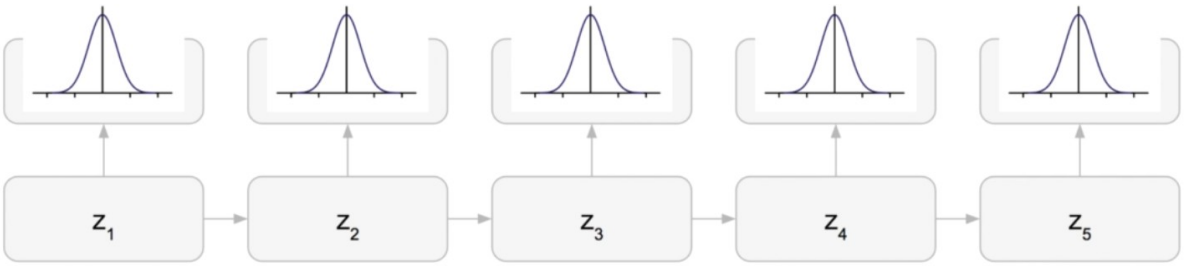


در عکس بالا ردیف اول حالات مشاهده و ردیف دوم حالات پنهان است. فلش‌هایی که از یک حالت در ردیف پایین به حالتی دیگر در ردیف پایین کشیده است در واقع همان توزیع احتمال گذار حالت است. فلش‌هایی که از ردیف دوم به ردیف اول کشیده شده است توزیع احتمال نماد مشاهده‌شده در حالت پنهان متنظار با آن است. در یک HMM با توجه به پردازش دیتاستی که داریم می‌توانیم نقش هر واژه را در جمله پیدا کنیم.

بازار بورس و سهام

با وجود این که هم‌اکنون مدل‌ها و روش‌های پیچیده‌تر و کاراتری برای مدل‌سازی بازار بورس و سهام ارائه شده است، قبلا از HMM برای مدل‌سازی بازار بورس و سهام استفاده می‌شد. به صورت شهودی می‌توان علت این مدل‌سازی را استنتاج کرد؛ هر حالت مشاهده (ضرر یا سود) وابسته به حالات پنهانی است که از دید ناظر مخفی است. این جا یک تفاوت عمده را با مثال قبلی (POS) حس می‌کنیم؛ در مثال قبلی ما اطلاعاتی از قبیل انواع، تعداد و منطق حالت‌های پنهان (نقش‌های دستوری) داشتیم ولی در یک مدل بازار بورس و سهام چنین اطلاعاتی را نداریم. در واقع از همین جهت هم مشخص می‌شود که چرا HMM برای بازار بورس و سهام چندان مناسب نیست. اول این که بازار ویژگی مارکوفی را نقض می‌کند: هر حالت به حالت‌های زیادی در گذشته بستگی دارد. دوم این که هر حالت مشاهده ممکن است تحت تاثیر تعداد زیادی حالت پنهان باشد. حتی اگر هم بتوانیم سازوکاری برای نظم بخشیدن این حالت‌های پنهان پیدا کنیم به علت این که توزیع احتمال گذارحالات یک ماتریس است، ضرب ماتریکس‌ها در بعدهای بالا می‌تواند بسیار کند و پیچیده باشد. با این وجود مدل‌سازی بازار بورس و سهام به کمک یک HMM قبلا صورت گرفته شده است.

برای حالات مشاهده و حالات پنهان می‌توان چنین دیاگرامی را رسم کرد:



بدیهی است که در این مدل‌سازی ما نیازمند یک HMM پیوسته می‌باشیم (برخلاف POS که با HMM گسسته به راحتی مدل‌سازیم می‌شود).

الگوریتم ویتربی

به یاد دارید که الگوریتم ویتربی برای پیدا کردن محتمل‌ترین سری حالات پنهان با داشتن یک سری مشاهدات است. می‌توان روی‌کرد این الگوریتم را در سه مرحله مشخص نمود:

بخش مقداردهی اولیه: در این بخش ماتریکس V را چنان می‌سازیم که در آن $V[t][s]$ بیانگر محتمل‌ترین مسیری که در مرحله t به حالت s ختم شود، باشد. از همین جهت مقدار $V[0][s]$ برابر با حاصلضرب توزیع احتمال اولیه یعنی $\Pi=\{\pi_i\}$ در احتمال اولین مشاهده یعنی $[A]_{1j}$ است.

- بخش تکرار: در اینجا به ازای t مرحله (از مرحله اول تا مرحله t) احتمال زیر را حساب می‌کنیم:

$$V[t][state]=max(V[t-1][\text{prev\_state}]\times \text{transition\_prob}[\text{prev\_state}][state]\times \text{emit\_prob}[state][obs[t]])$$

که در آن prev_state شامل تمام مقادیر ممکن حالات پیش می‌شود. به این بخش، بخش پیش‌روی الگوریتم نیز گفته می‌شود.

-بخش عقب‌رو: پس از بخش تکرار حالتی را که در گام t ام بیش‌ترین احتمال را داشت پیدا می‌کنیم. سپس به کمک ماتریکس V ساخته شده، به عقب بر می‌گردیم تا محتمل‌ترین مسیر ممکن را به ازای حالت پایانی پیدا کنیم.

برای مثال مدلی را فرض کنید که در آن دو حالت پنهانی Sunny و Rainy و دو حالت مشاهده Wet و Dry داریم. توزیع احتمال اولیه چنین است: $P('Sunny')=0.8, P('Rainy')=0.2$ همچنین احتمال‌های گذار مفروض است:

$$P('Sunny'\rightarrow'Sunny')=0.7$$
$$P('Sunny'\rightarrow'Rainy')=0.3$$
$$p('Rainy'\rightarrow'Sunny')=0.4$$
$$P('Rainy'\rightarrow'Rainy')=0.6$$

اگر احتمال‌های انتشار چنین باشد: $P('Dry'\mid'Sunny')=0.9, P('Wet'\mid'Sunny')=0.1$
$P('Dry'\mid'Rainy')=0.2, P('Wet'\mid'Rainy')=0.8$

و سری Dry → *Wet* → Dry مشاهده شود، طبق الگوریتم ویتربی داریم:

$$V[0]['Sunny']=0.8*0.9=0.72$$
$$V[0]['Rainy']=0.2*0.2=0.04$$
$$V=\begin{pmatrix}0.72 & 0.04\end{pmatrix}$$

در مرحله تکرار ماتریکس V ساخته می‌شود (از محاسبات فاکتور گرفته شده است):

$$V=\begin{pmatrix}0.72 & 0.04\\ 0.0504 & 0.0216\end{pmatrix}$$

در نهایت از آنجا که در مرحله tام حالتی که بیش‌ترین احتمال را داشت (سطر اول را نگاه می‌کنیم) حالت آفتابی بود، به عقب برمی‌گردیم تا به مرحله اول برسیم (سطر دوم را نگاه می‌کنیم). در این حالت باز محتمل‌ترین حالت آفتابی است. پس محتمل‌ترین حالات پنهان برابر با آفتابی ← آفتابی می‌باشد.

برچسب‌گذاری ادات سخن به زبان فارسی و انگلیسی

در پوشه hmm/pos برنامه مربوط به آموزش یک عامل HMM برای برچسب‌گذاری ادات سخن به زبان فارسی و انگلیسی قرار دارد. برای زبان انگلیسی از دیتاست مربوط به ۸ مگابایت محتوای خبری استفاده شده است در حالی که برای زبان فارسی از دیتاست بی‌جن‌خوان[46] استفاده شده است. هم‌چنین درون فولدر فایل تکستی جهت مشخص نمودن معنای هر نماد pos قرار دارد. هر دو دیتابیس به فرمت خاصی که برای برنامه قابل هضم است تغییر یافته‌اند ولیکن پیش‌پردازش دیتاست بی‌جن‌خوان به شدت پرزحمت‌تر بود؛ چرا که در این دیتاست فارسی‌نویسی به شکل استاندارد نبود. به عنوان مثال به جای نیم‌فاصله از فاصله استفاده شده بود (و گاهی اوقات هم سرهم‌نویسی ترجیح داده شده بود) که همین مورد کوچک موجب تشخیص به اشتباه دو کلمه به جای یک کلمه می‌شد. از دیگر مشکلات این دیتاست در وجود همزه‌ها، ی‌های عربی، استفاده از ا به جای آ و ... می‌توان اشاره کرد. به صورت کلی به علت این که هنوز استانداردنویسی در زبان فارسی رسم نشده‌هاست، تحقیق و پژوهش در زبان فارسی با مشکلات زیادی رو به روست. در همین برنامه تفاوت عمل‌کرد چشم‌گیر عاملی که بر روی دیتاست انگلیسی آموزش دیده است با عاملی که بر روی دیتاست فارسی آموزش دیده شده است بدیهی می‌باشد. بخشی از مشکلاتی که در دیتاست بی‌جن‌خوان یافت می‌شد را گاه به صورت دستی گاه با نوشتن اسکریپت از بین برده‌ام و نتیجه یک دیتاست ۳۷ مگابایتی شده است.

برنامه از ۴ فایل تشکیل شده است. فایل utils.py فایلی است که نهایتا یک فرهنگ واژگان بر اساس دیتاست موردنظر می‌سازد. برنامه ما باید باید حالات ادات سخنی را که پیش از این با آن روبه‌رو نشده بود را برچسب‌گذاری کند. این کار به کمک تابع assign_unkown صورت می‌گیرد که به آن توکن برچسب «ناشناخته» زده می‌شود. منتها برای ما اهمیت دارد که که این برچسب ناشناخته از نوع اسم‌ساز، صفت‌ساز، فعل‌ساز و یا قید ساز است؟ از این جهت و با توجه به دو زبانه بودن برنامه‌مان، لیست‌هایی از پرکاربردترین پسوندهای زبان فارسی و انگلیسی به تفکیک نقش ادایی جمع‌آوری شده است:

```
44      if str(tok).isascii(): #then it's english:
45          noun_suffix = ["action", "age", "ance", "cy", "dom", "ee", "ence",
46          verb_suffix = ["ate", "ify", "ise", "ize"]
47          adj_suffix = ["able", "ese", "ful", "i", "ian", "ible", "ic", "ish"
48          adv_suffix = ["ward", "wards", "wise"]
49      else:
50          noun_suffix = ["اله", "باز", "بان", "تر", "سار", "ستان", "سرا", "
51                         "وار", "گون", "دان"]
52          verb_suffix = ["ام", "اید", "اند", "ایم"]
53          adv_suffix = ["آسا", "آگین", "سان"]
54          adj_suffix = ["انه", "گان", "گون"]
55      # Digits
```

[46] https://dbrg.ut.ac.ir/بیژن%e2%80%8cخان/

چنین لیستی صد البته با خطاهایی رو به رو است زیرا ممکن است بر اساس یک پسوند، نقش دستوری یک واژه را اشتباه حدس بزند (حالتی را در نظر بگیرید که آن پسوند بخش جدانشدنی آن واژه باشد و ما به اشتباه آن را پسوند در نظر گرفته‌ایم).

تابع processing به ازای لیستی از توکن‌ها (واژگان) حالاتی را که توکن خالی است یا ناشناخته است کنترل می‌کند.

تابع build_vocab پس از خواندن فایل دیتاست موردنظر ما و شمردن تکرارها به ازای هر توکن و سپس الحاق برچسب‌های خاص ناشناخته (مثل –unk_adj-- برای صفات نامشخص)، فرهنگ‌واژگان را می‌سازد و پس از مرتب‌سازی آن را برمی‌گرداند.

فایل train.py برای یادگیری عامل HMM است و اولین برنامه‌ای است که باید اجرا شود. در ابتدا دیتاست مورد نظر (فارسی یا انگلیسی) از کاربر پرسیده می‌شود و سپس برنامه فرهنگ‌واژگان را درست می‌کند و در فایل vocab.pkl ذخیره می‌کند و ماتریس‌های گذار و انتشار را محاسبه و در دو فایل ذخیره می‌کند.

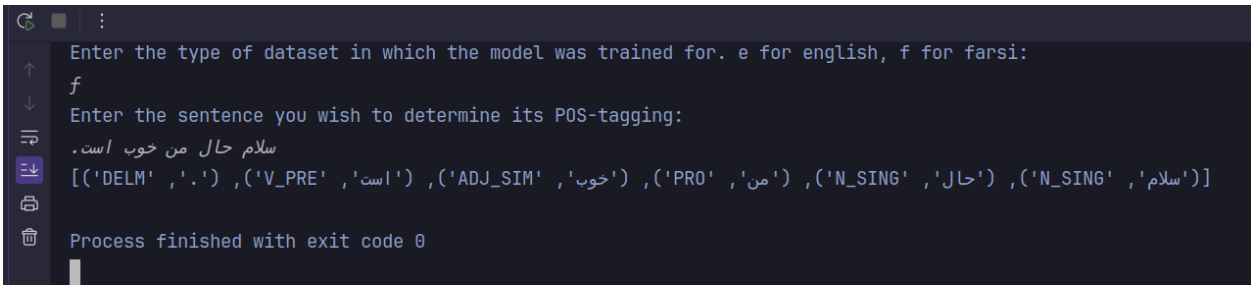
فایل hmm.py مربوط به عامل زنجیره مارکوف پنهان است. در این فایل تابع build_vocab_to_index به کمک فرهنگ‌واژگانی که در utils.py ساخته می‌شود، هر توکن را به اندیس آن متناظر می‌کند.

تابع کمکی create dictionaries با شمردن تکرارها در دیتاست، ساختارهای داده‌ای را برای ذخیرسازی تعداد تکرارتگ‌ها، انتشارها و گذارها می‌سازد و آن‌ها را برمی‌گرداند.

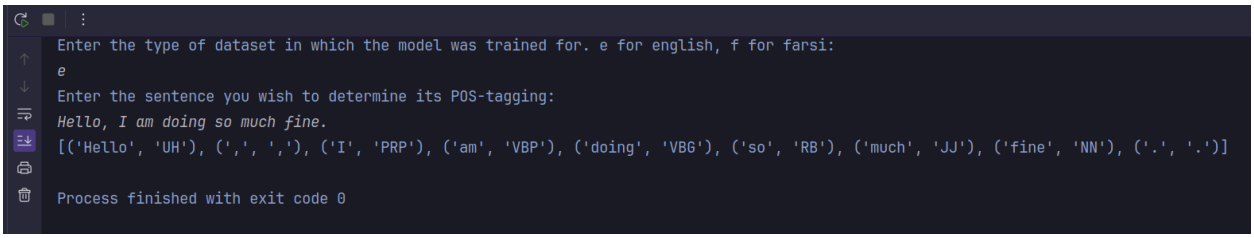
تابع create_transition_matrix و create emission matrix به ترتیب ماتریس گذار و ماتریس انتشار را می‌سازد. مکانیسم عمل هر دو شبیه هم است منتها با این تفاوت که ما در ماتریس گذار تعداد تکرارهای حالت کنونی به شرط حالت قبلی (حالت iام به ازای حالت j) را می‌شماریم ولی در ساخت ماتریس انتشار بررسی می‌کنیم که به ازای هر حالت ادات سخن (هر تگ) چه تعداد توکن (واژه) پدیدار شده است. در نهایت به کمک شمارش‌های به دست آمده و با بهرمجویی از تکنیک laplace smoothing، احتمال‌های مربوط به هر حالت را به دست می‌آوریم. تکنیک laplace smoothing که به آن additive smoothing نیز می‌گویند، روشی برای «صاف»سازی داده‌هاست که در آن با اضافه کردن یک عامل کوچک، از صفر شدن محاسبات جلوگیری می‌کنیم. این تکنیک در آمار و احتمالات به کران استفاده می‌شود.

سه تابع viterbi_forward، initialization و viterbi_backward با هم یک جز واحد را تشکیل می‌دهند. با پیاده‌سازی الگوریتم viterbi، این سه تابع بهترین مسیر و در نهایت بهترین حدس ممکن را به ازای یک سری واژگان مشاهده شده را محاسبه می‌کنند و آن حدس را بر می‌گردانند. این سه تابع برای بخش آزمایش برنامه حیاتی می‌باشند.

فایل pos.py برای آزمایش و جواب گرفتن از عامل hmm است. لازم به ذکر است که این فایل باید پس از فایل train.py اجرا شود. ابتدا دیتاستی که از آن برای آموزش دادن عامل استفاده شده است پرسیده می‌شود و سپس از کاربر جمله‌ای را می‌خواهد. پس از گرفتن جمله برنامه به کمک الگوریتم ویتربی حدس خود را برای نقش‌های دستوری آن جمله چاپ می‌کند.



خروجی برنامه به ازای یک جمله ساده فارسی.



خروجی برنامه به ازای یک جمله ساده انگلیسی.