

**تمرین سوم**

اعضای گروه:

علی درخشش

محراب مرادزاده

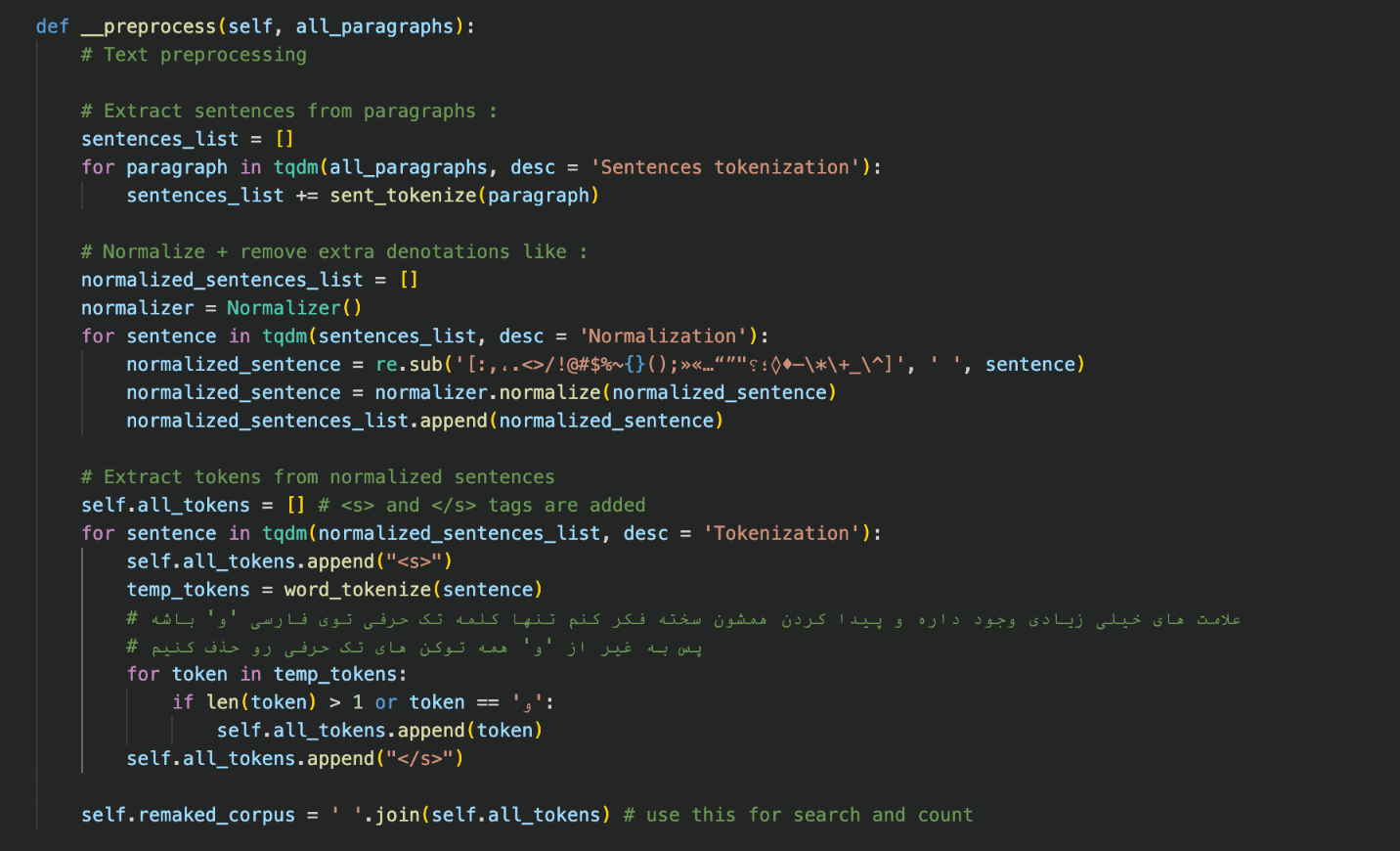
ابوالفضل ملک احمدی

ما در این بخش یک مدل بر پایه مدل زبانی n-gram آموزش دادیم ٬ که این مدل برای پیش بینی کلمه بعد مورد استفاده و آموزش داده شده است .

در ابتدا با درخواست زدن به مخزن مورد نظر داده های مربوطه را جهت شروع کار استخراج میکنیم :



پس از بارگیری مجموعه داده جملات هر پاراگراف را با استفاده از SentenceTokenizer موجود در کتابخانه ی hazm جدا می شوند و سپس جملات نرمال می شوند و نیز چون در داخل جملات یکسری علامت های نگارشی و سایر علائم (از جمله ایموجی و غیره) وجود دارد آنها را نیز حذف میکنیم :



توجه یکی دیگر از کار های که در رشته کد بالا مورد توجه است توکن کردن است زیرا ما باید مرز جملات را مشخص کنیم به خاطر اینکه کلمه اخر یک جمله با کلمه اول جمله بعد را به عنوان یک bigram نگیرد و با هم احتمالش را حساب کند زیرا این دو با هم مرتبط نیست .

پ.ن: برای حذف علائم نگارشی و سایر علائم غیر کاربردی از انجایی که تنها تک حرف داخل فارسی «و» است پس انرا نگه داشته و مابقی را حذف میکنیم.

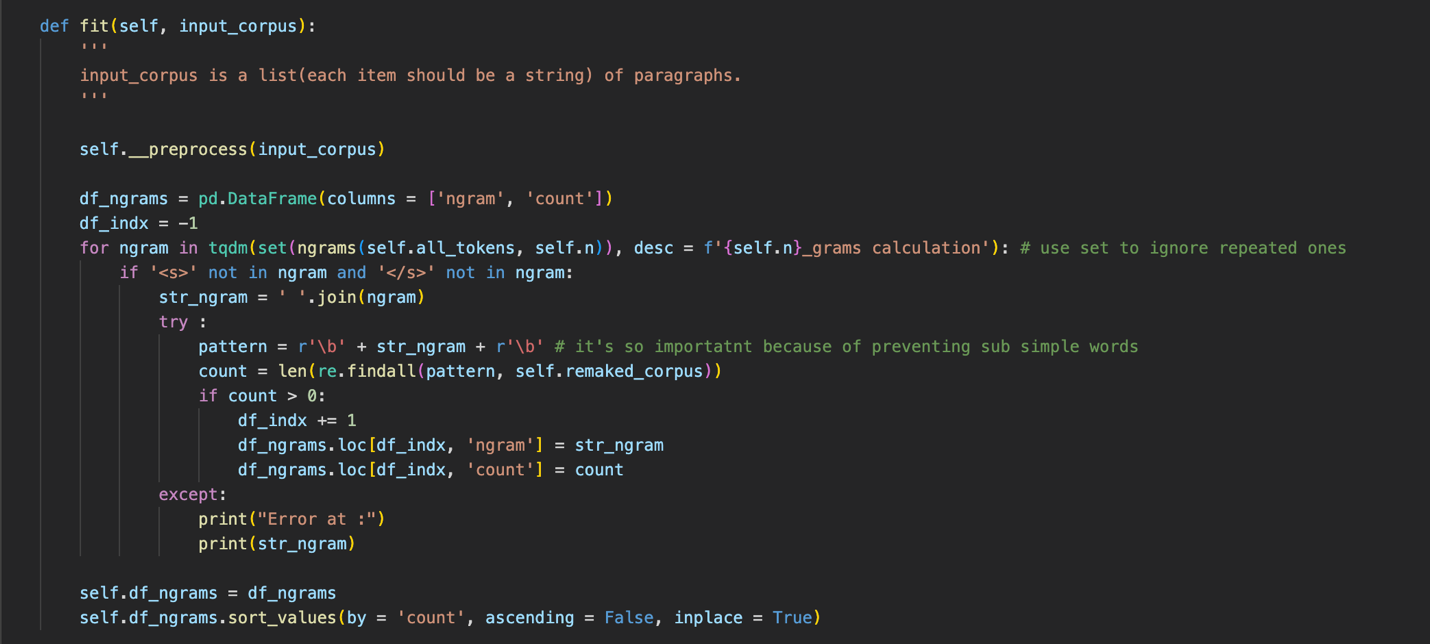
**معرفی مدل**

* تابع fit

در این قسمت ابتدا ما یک dataframe شامل ngram و count میسازیم که داخل این dataframe بسته به این که مقدار n برابر ۱ باشد تعداد 1-gram ها اگر ۲ باشد تعداد 2-gram ها و اگر ۳ باشد تعداد 3-gram ها به همین صورت تا به بالا قرار میگیرد و تعداد هر کدام را شمرده و در ستون count قرار میدهد :

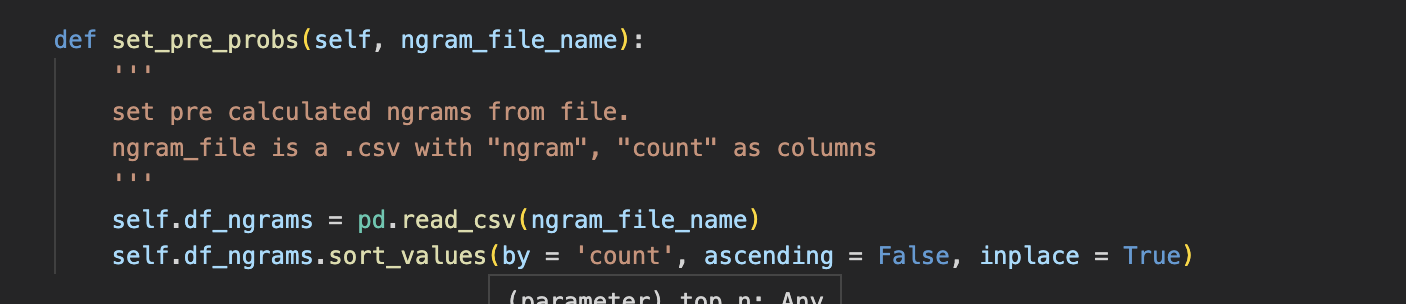
نکته: ۱. چون nltk.ngrams از روی توکن ها ngram های تکراری نیز تولید میکند از set استفاده شده است .

۲. برای محاسبه تعداد هر ngram از تابع findall رجکس استفاده شده است .



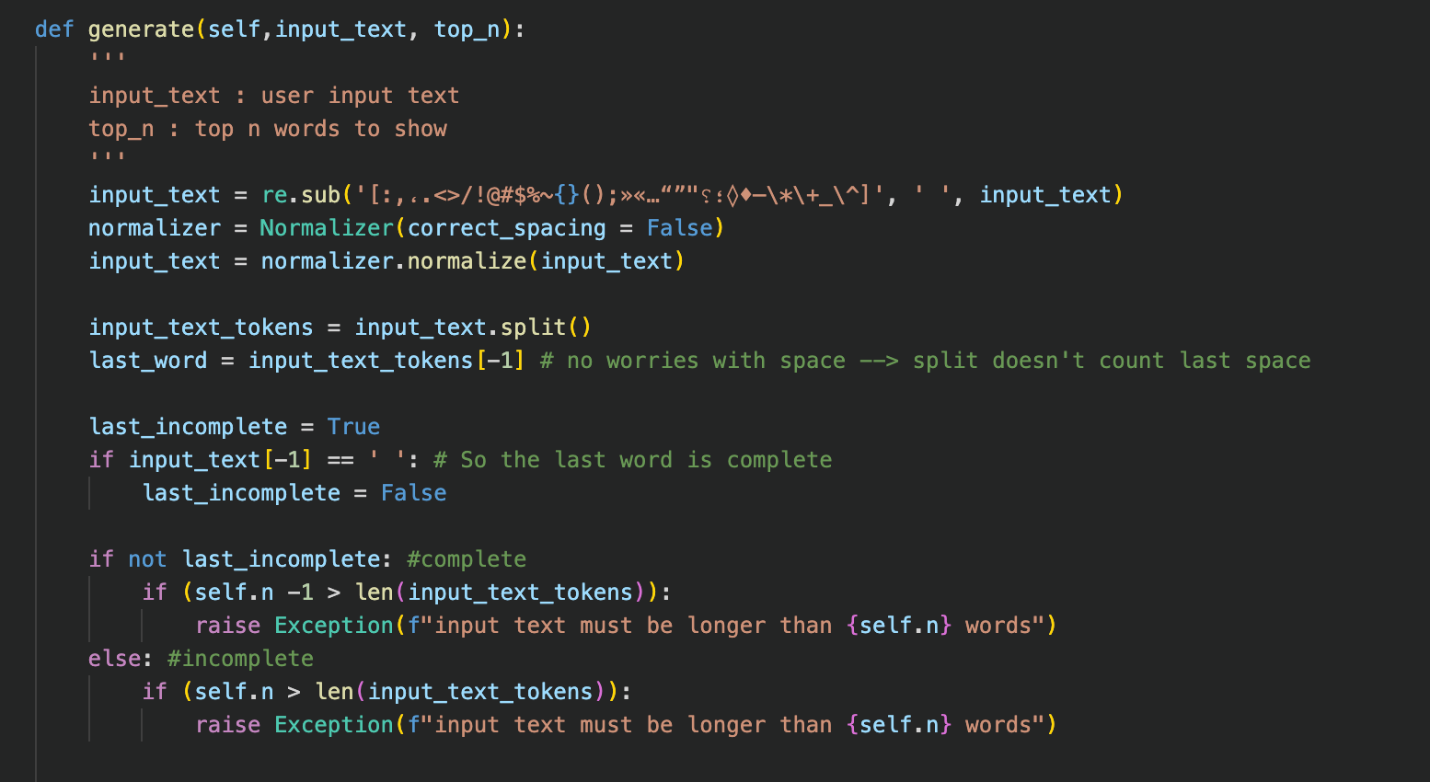
* تابعset\_pre\_probs

توجه این مدل فقط نیاز به یکبار آموزش دیدن دارد و در دفعات بعد کاربر بدون نیاز به آموزش مدل و تنها با داشتن جدول تشکیل شده در بالا میتواند از مدل استفاده کند :

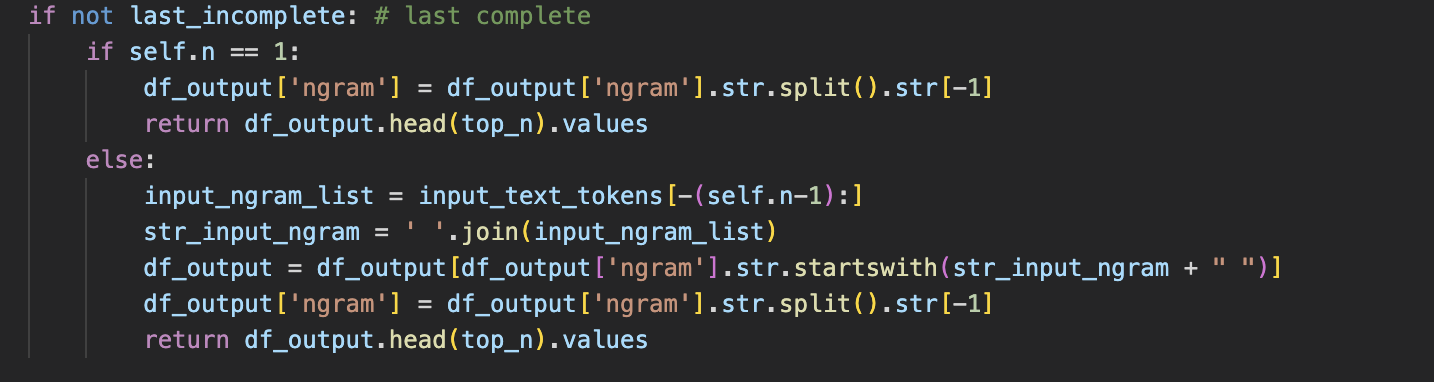


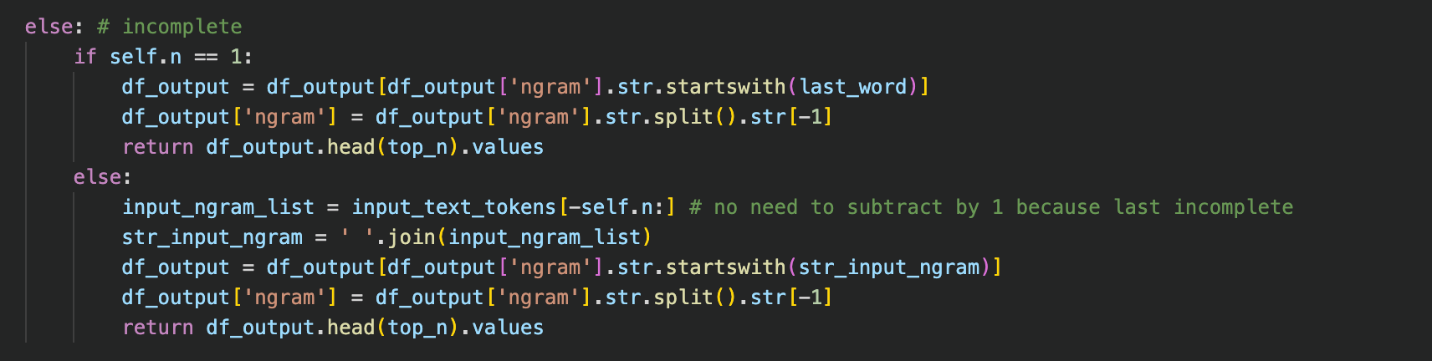
* تابعgenerate

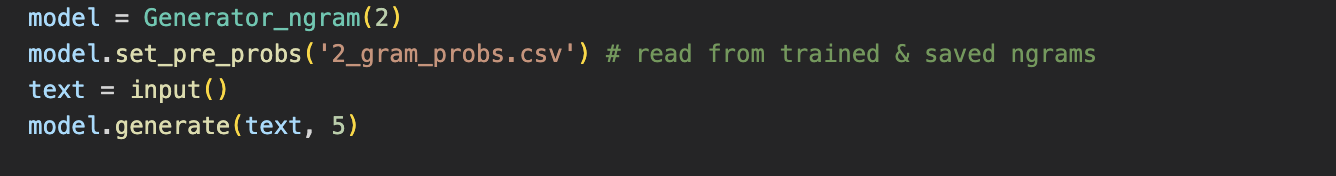
در این تابع ابتدا جمله ورودی پیش پردازش میشود و از انجایی که نرمالایز hazm فاصله ها را حدف میکند و ما برای اینکه بفهمیم که کاربر کلمه اخر خود را کامل داده یا ناقص نیاز است که فاصله ها را داشته باشی زیرا ما با استفاده از فاصله ها تشخیص میدهیم که کلمه اخر کامل یا ناقص است. در نتیجه مقدار correct\_spacing را false در نظر میگیریم و در مرحله بعد کامل بودن یا نبودن کلمه اخر را تشخیص می دهیم



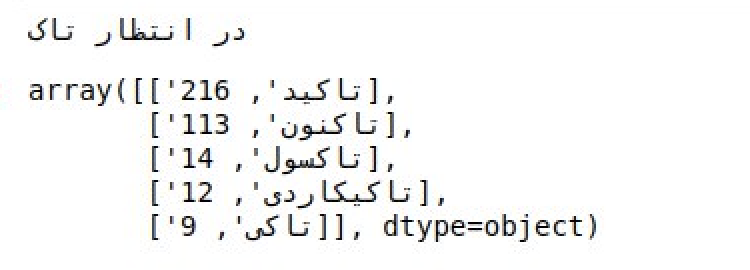
حال اگر کلمه کامل باشد و n برابر ۱ باشد در اینجا باید بیشترین احتمال ها بدون در نظر گرفتن قبلی ها برگردانیم و کلمه ای که بیشترین احتمال را داشته بر میگردانیم و اگر n بیشتر از ۱ بود از آن ترکیب های محاسبه شده(به عنوان مثال ترکیب های دوتایی یا ترکیب های سه تایی محاسبه شده )استفاده میکنیم

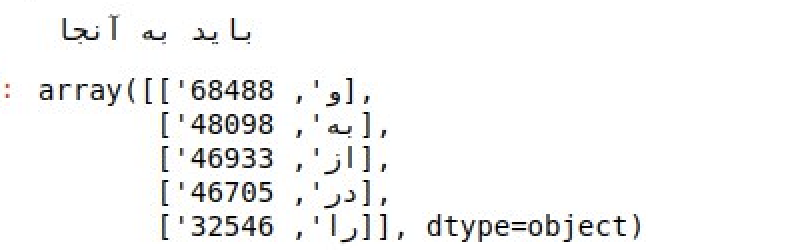


برای اینکه کلمه بعد به صورت ناقص امده باشد روند بالا را طی میکنیم با این تفاوت که فاصله اضافه شده برای کلمات کامل را در این قسمت در نظر نمیگیریم چون میخواهیم ابتدا کلمه ناقص را پیش بینی کنیم 

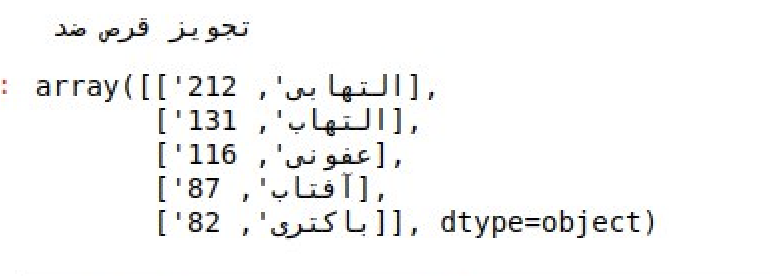
حال برای استفاده از مدل به صورت زیر عمل میکنیم :ابتدا n مورد نظر برای مدل را مشخص کرده که در این جا مقدار ۲ در نظر گرفته شده سپس فایل پیش بینی شده از ترکیب ها که در اینجا ترکیب های دوتایی که محاسبه شده بود خوانده شده حال جمله مورد نظر را به آن میدهیم که نتایج بدست آمده به صورت زیر است :

N=1:

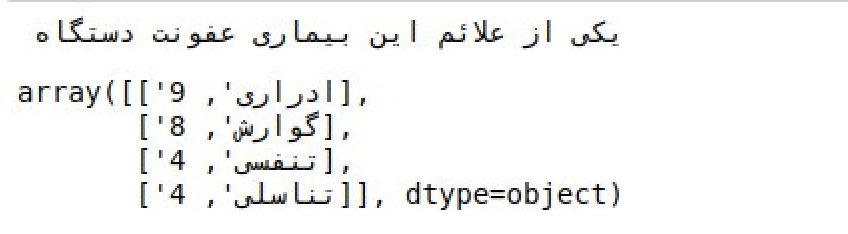




N=2:



N=3:



* اثبات چرایی استفاده از تعداد محاسبه شده به جایی احتمال :

*حالا با استفاده از روابط بالا در مثال زیر فرضیه خود را ثابت میکنیم*

*مثال :*

*x*= *افزایش نرخ تورم سال جاری----افزایش نرخ تورم سال جاری*

*برای محاسبه احتمال جمله به این صورت است که :*

*حالا فرض کنید جاری را نداشته حال برای اینکه این کلمه را پیش بینی کنیم باید کلمه ای را انتخاب کنیم که احتمال کل جمله را بیشینه کند*

*به طور مثال اگر به جای y سه کلمه داشته باشیم مثل جاری ،برای ،شیر حال ما احتمال کلمه جمله را برای تک تک این کلمات حساب میکنیم*

*با توجه به اینکه ترم های قبل ترم اخر یکی است میتوان کل انها را نادیده گرفت و فقط ترم اخر را در نظر گرفت و برای ما مهم ست*

*حال با استفاده از قضیه مارکوف می دانیم که به جای اینکه کل p(x) رو در نظر بگیریم اگر n=1 باشدy مستقل از همه ی قبلی ها میشه و بنابراین در این ترم برابر خود p(y) است پس کافی است مقدار زیر را محاسبه کنیم و کلمه ای را پیدا کنیم که بیشترین احتمال را داشته باشد :*

حال اگر n=2 باشد یعنی مارکوف مرتبطه دوم که یعنی فقط به کلمه قبلی بستگی دارد پس کافیه ما برای مثال بالا عبارت زیر را محاسبه کنیم

که مخرج تمام این کسر ها برابر است که میتوان ان را نادیده گرفت و فقط ترم صورت کسر را باید بیشینه کنیم به صورت زیر :

حال برای اینکه این احتمال ها را به دست بیاوریم باید تعداد مثلا جا های که به عنوان مثلا «سال» در کنار «جاری» امده را بشماریم و بر تعداد کل کلمات داخل جمله تقسیم کنیم که احتمال این ترم را به دست بیاوریم برای مابقی هم به همین صورت که از انجایی که باز هم مخرج سه ترم به یک صورت است فقط نیاز است تعداد جاهای که این کلمات کنار هم امده است را محاسبه کنیم .

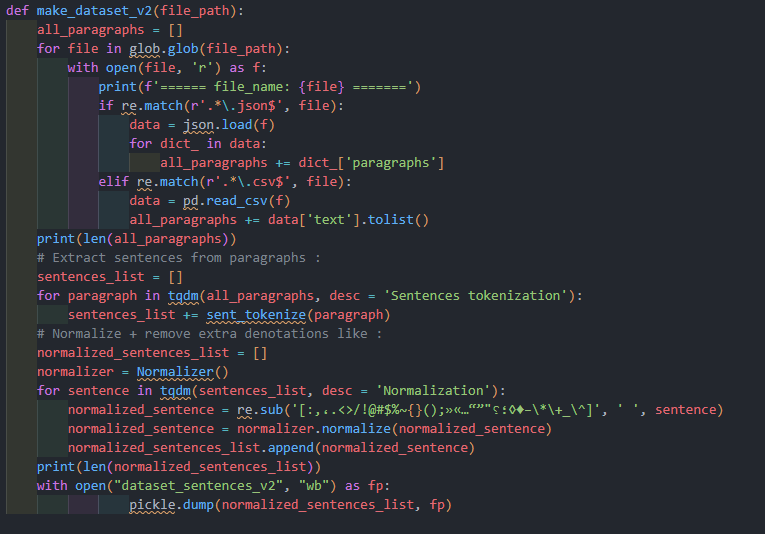
در این بخش مدل عمیق مورد بررسی قرار می‌گیرد.

برای اجرا این بخش ما از vps با مشخصات 72 گیگابایت رم، یک کارت گرافیک Nvidia RTX 3090 و 12هسته پردازشی از HPC دانشگاه شریف استفاده کردیم.

* تابعmake dataset و make dataset v2

برای استفاده از دیتاست خام و آماده کردن آن (توکنایز کردن و نرمال کردن) در ابتدا ما با استفاده از تابعی که دانشجویان ترم‌های گذشته برای این کار تدارک دیده بودند استفاده کردیم. زمان استفاده شده برای بر روی HPC دانشگاه 18 ساعت بود.

در تابع make dataset v2 بهبودهایی بر روی تابع اولیه شکل گرفت و مرتبه زمانی تابع از O(n3) به O(n2) کاهش یافت و زمان اجرا بر روی HPC به کمتر از 3 دقیقه کاهش یافت. همچنین بهینه سازی‌های دیگری نیز در نرمال سازی جملات صورت گرفت مانند حذف علائم نگارشی و ... .



همچنین برای غنی‌تر کردن دیتاست‌های مورد استفاده، داده‌های سایت‌های العربیه و lastsecond کراول شده‌اند که کد‌های کراوال در فایل‌های ژوپیتر lastsecond\_scraper و DataCrawling موجود می‌باشد.

**مدل پایه**

بهترین مدل زبانی موجود در زبان فارسی در حال حاضر مدل GPT2 بلبل‌زبان می‌باشد. همچنین میتوان از مدل‌های زبانی جدیدتر و متن بازی مانند LLaMa هاگینگ فیس و Alpaca نیز استفاده کرد اما به دلیل حجم بالای وزن ها و همچنین

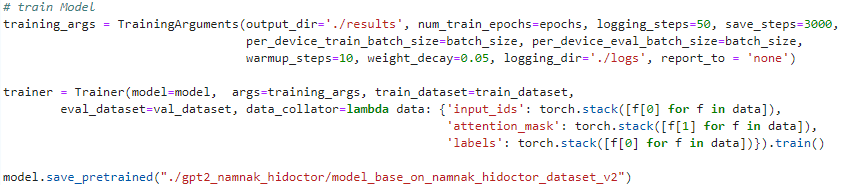
محدودیت سخت افزاری موجود (حتی در صورت استفاده از کارت گرافیک‌های HPC دانشگاه) این امر امکان پذیر نیست. لذا مدل GPT2 بهترین مدل در دسترس است.

**آموزش مدل GPT2 بر روی دیتاست پایه**

در ابتدا ما مدل GPT2 را بر روی دیتاست پایه‌ای که فقط شامل متون سایت های Hidoctor و namnak بود تنظیم دقیق کردیم. این امر در 8 epoch انجام شد.

تنظیم دقیق مدل با استفاده از تابع Trainer و TrainingArguments کتابخانه Transformers انجام شد.

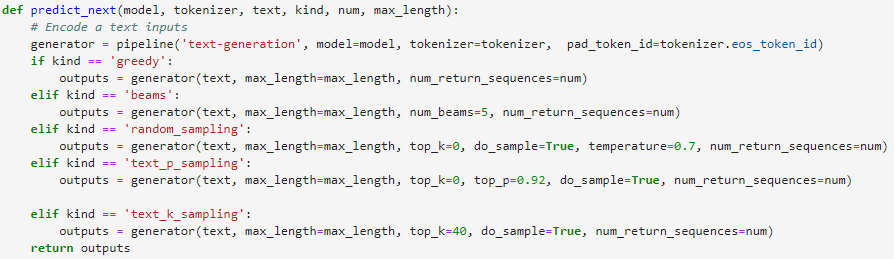
همچنین قابل ذکر است با توجه به استفاده از یک GPU 3090 ما قادر بودیم تا از تمامی جملات دیتاست به طور کامل استفاده کنیم.



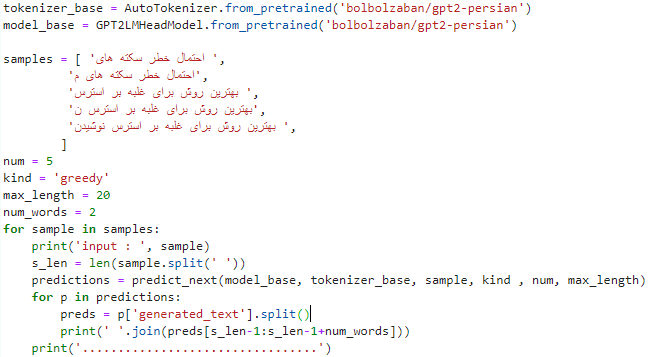
در مرحله بعد مدل توسط دیتاست غنی‌تر شده شامل کلمات پایه و کلمات سایت‌های العربیه و lastsecond یک بار دیگر در 6 epoch تنظیم دقیق شد.

**روش های رمزگشایی و نتایج**

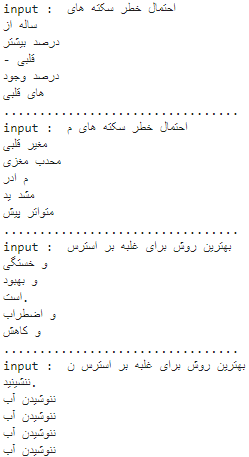
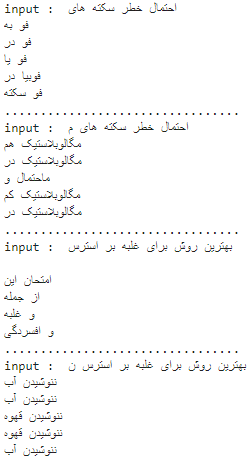
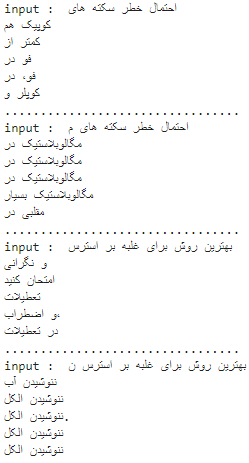
برای بررسی نتایج همان‌طور که در این [سایت](https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation-ba95ee0faadc) اشاره شده بود چندین روش برای دیکودینگ وجود دارد که با استفاده از کتابخانه Transformers این روش‌ها به سادگی قابل پیاده‌سازی هستند.



حال با استفاده از تابع predict\_next می‌توانیم نتایج خروجی مدل‌های تنظیم دقیق شده را مشاهده کنیم.



در زیر خروجی مدل‌های پایه، تنظیم دقیق شده بر روی دیتاست پایه، و دیتاست غنی با استفاده از رمزگشایی greedy را مشاهده می‌کنید:



مدل تنظیم دقیق شده بر روی دیتاست پایه

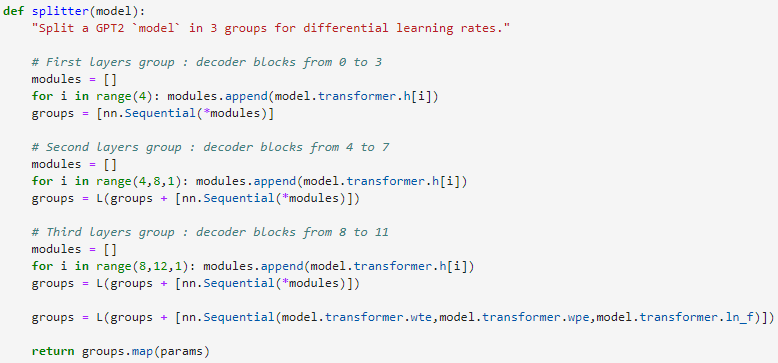
مدل تنظیم دقیق شده بر روی دیتاست غنی

مدل پایه GPT2

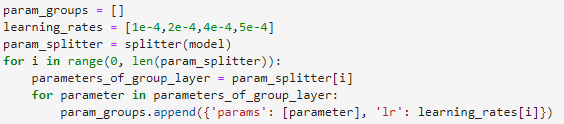
**تنظیم دقیق مدل با استفاده از فریز کردن لایه‌ها**

همان‌طور که در [مقاله](https://aclanthology.org/P18-1031.pdf) آقای Jeremy Howard اشاره شده است با استفاد از فریز کردن تدریجی لایه‌ها و استفاده از نرخ یادگیری متفاوت برای هر لایه می‌توانیم مدل را تنظیم دقیق کنیم.

تابع splitter لایه‌های مدل را به چهار قسمت تقسیم می‌کند:



و نرخ یادگیری را برای هر لایه تنظیم میکنیم، برای لایه‌های ابتدایی نرخ یادگیری کم و برای لایه‌های انتهایی این نرخ افزایش می‌یابد.



**فریز کردن لایه‌‌ها**

روش دیگری که در مقاله اشاره می‌شود فریز کردن مرحله به مرحله تعدادی از لایه‌ها در ایپاک‌‌های مختلف است. در ایپاک اول لایه‌های آغازین فریز میشوند و در ایپاک ‌های بعدی لایه های بعدی فریز می‌گردند.



در نهایت مدل با استفاده از دیتاست غنی در 3 epoch تنظیم دقیق شد و نتایج به در زیر آمده است.

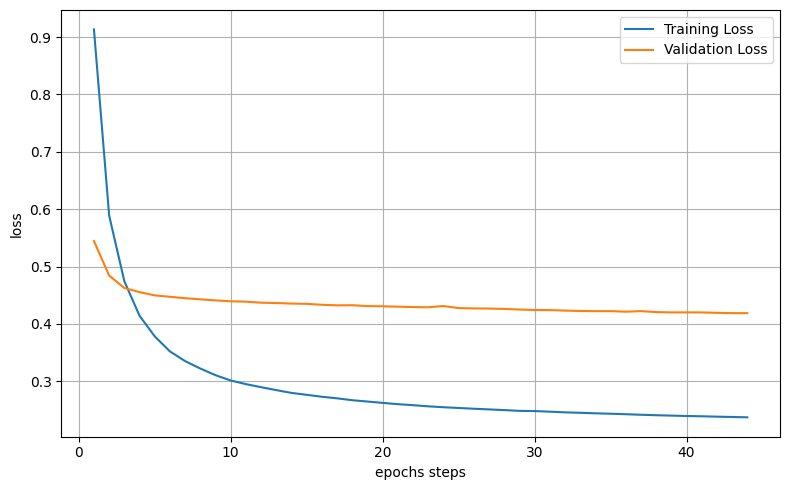
ایپاک اول:

Average training loss: 0.47

Perplexity: 1.61

Validation Loss : 0.42

Perplexity: 1.52



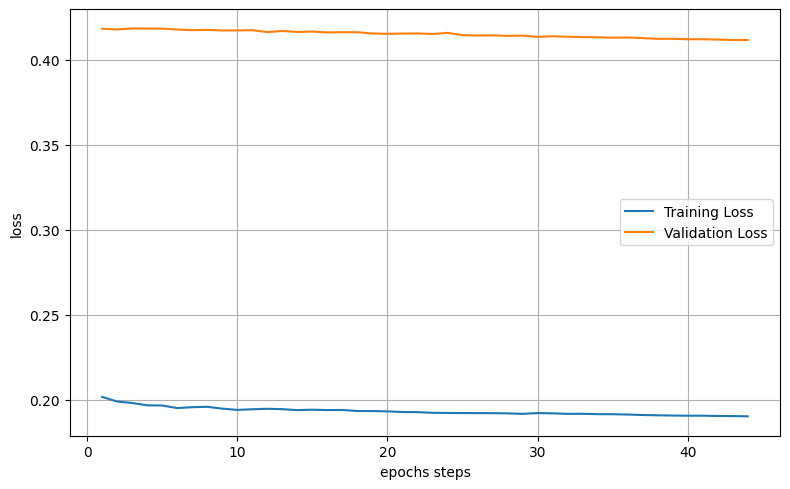
ایپاک دوم:

Average training loss: 0.38

Perplexity: 1.46

Validation Loss : 0.41

Perplexity: 1.51



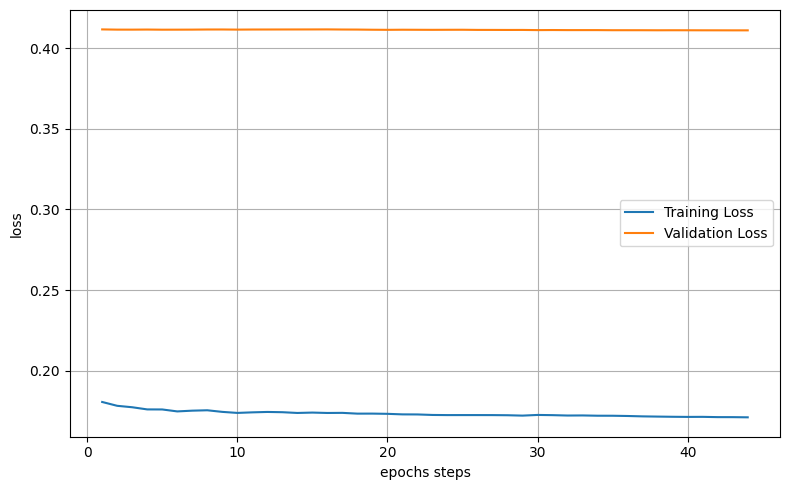
ایپاک سوم:

Average training loss: 0.34

Perplexity: 1.41

Validation Loss : 0.41

Perplexity: 1.51



نتایج خروجی این مدل در زیر قابل مشاهده است:

