## توضیحات مدل و نتایج:

مدلی که روی آن مدل زبانی را آموزش دادم یک LSTM با ۲لایه ۲۵۶تایی از hidden stateها است.
بعد از تمیزسازی و آماده شده داده‌ها میانگین طول جملات حدودا برابر با ۱۶۵ شد و به همین دلیل تعداد hidden stateها را مقداری بیشتر و برابر با ۲۵۶تا درنظر گرفتم و مدل را روی جملات به دست آمده از مجموعه اخبار آموزش دادم.
این مدل تابع هزینه CrossEntropy روی حدود ۶۰۰۰۰۰ جمله با میانگین حدود ۱۶۵ کاراکتر آموزش دیده شده و توی ۸ epoch هم روی مجموعه داده آموزش و هم روی مجموعه داده validation مقدار loss آن به حدود ۱.۳ رسیده است.
این مدل روی ۱۰۰۰تا از جملات مجموعه داده ارزیابی به صورت میانگین به مقدار حدود ۱.۲ برای لگاریتم perplexity و ۰.۶ برای میانگین مقادیر CER می‌رسد.(برای اجراهای متفاوت این مقادیر ممکن است به علت انتخاب بین ۳تا کاراکتر اول در زمان تولید جمله مقدار اندکی متفاوت باشد.)

## توضیحات کد:

- فایل data_cleaner.py: این فایل شامل کلاس TextCleaner است که وظیفه تمیزسازی و آماده کردن مجموعه داده را دارد.
- فایل train.py: این فایل شامل کلاس CharRNN است که از torch.nn.Module ارث‌بری می‌کند و برای تعریف ساختار شبکه عصبی است و با استفاده از مجموعه داده آماده شده در مرحله قبل عملیات train را انجام می‌دهد و در هر گام در صورت بهبود مدل روی مجموعه داده validation آن checkpoint را ذخیره می‌کند.
- فایل language_model.py: این فایل شامل کلاس LanguageModel است که با استفاده از ساختار شبکه عصبی تعریف شده در مرحله قبل و مدل آموزش دیده شده توابع مربوط به مدل زبانی را پیاده‌سازی می‌کند.
  - تابع lm_unit: این تابع وظیفه دارد تا با گرفتن آدرس یک checkpoint مدل آموزش دیده شده را بارگزاری کند.
  - تابع get_next_states_and_output: این تابع یک دنباله از کاراکترهای اندیس‌گذاری شده را به عنوان ورودی دریافت می‌کند و با استفاده از مدل آموزش دیده خروجی‌ها و stateهای حاصل شده را در خروجی برمی‌گرداند.
  - تابع prefix_to_hiddens: این تابع یک دنباله از کاراکترهای اندیس‌گذاری شده را در ورودی می‌گیرد و در خروجی تبدیل شده آن‌ها به stateها را برمی‌گرداند.
  - تابع get_probability: این تابع یک دنباله از کاراکترهای اندیس‌گذاری شده را به عنوان ورودی دریافت می‌کند و با استفاده از تابع get_next_states_and_output کاراکترها به همراه احتمال رخداد آن‌ها با فرض وجود آن prefix را در خروجی برمی‌گرداند.
  - تابع get_overall_probability: این تابع یک دنباله از کاراکترهای اندیس‌گذاری شده را به عنوان ورودی دریافت می‌کند و با استفاده از تابع get_probability احتمال وقوع این جمله را محاسبه می‌کند. بدین شکل که با شروع از ابتدا در هر مرحله یک prefix از این

جمله را به تابع get_probability می‌دهد و احتمال محاسبه شده کاراکتر بعدی در این جمله را ضربدر یک متغیر می‌کند و به همین شکل پیش می‌رود تا جمله به پایان برسد.

- تابع generate_new_sample: این تابع یک دنباله از کاراکترهای اندیس‌گذاری شده را در ورودی می‌گیرد و با استفاده از تابع get_probability تا زمانی که به کاراکتر پایان جمله نرسیده باشیم یا طول عبارت تولید شده به میانگین جملات نرسیده باشد از بین ۳تا کاراکتر با بیشترین احتمال رخداد یکی را به صورت تصادفی به انتهای این prefix اضافه می‌کند.

- فایل evaluation.py: این فایل شامل کلاس LanguageModelEvaluator است که وظیفه ارزیابی مدل زبانی را به عهده دارد.
  - تابع perplexity_log: این تابع یک جمله اندیس‌گذاری شده را به عنوان ورودی دریافت می‌کند و بعد از محاسبه احتمال این جمله با استفاده از تابع get_overall_probability مدل زبانی میزان perplexity برای این جمله را محاسبه می‌کند.
  - تابع char_error_rate: این تابع یه جمله اندیس‌گذاری شده را در ورودی دریافت می‌کند و بعد از تولید یه جمله با استفاده از ۱۰تا کاراکتر اول این جمله با کمک تابع generate_new_sample مدل زبانی مقدار CER این دو جمله را محاسبه و برمی‌گرداند.
  - تابع evaluate_test_set: این تابع داده‌های ارزیابی آماده شده را با کمک توابع perplexity_log و char_error_rate ارزیابی می‌کند و به عنوان خروجی میانگین این مقادیر را برمی‌گرداند.