به نام خدا

تمرین سوم درس پردازش زبان طبیعی

عنوان: مدل زبانی مبتنی بر شبکه‌های عصبی بازگشتی

استاد درس:

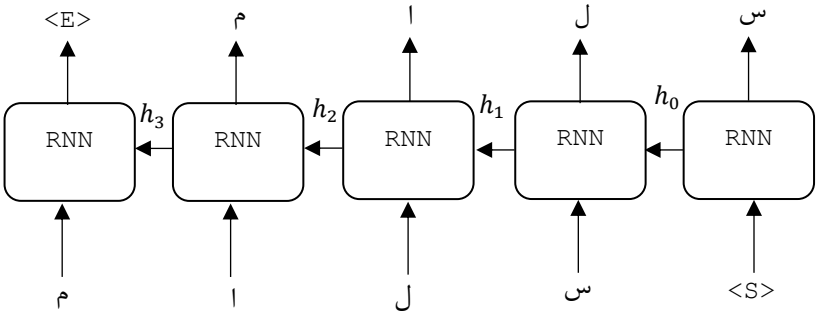
دکتر اکبری

موعد تحویل: ۱۳۹۹/۰۹/۹

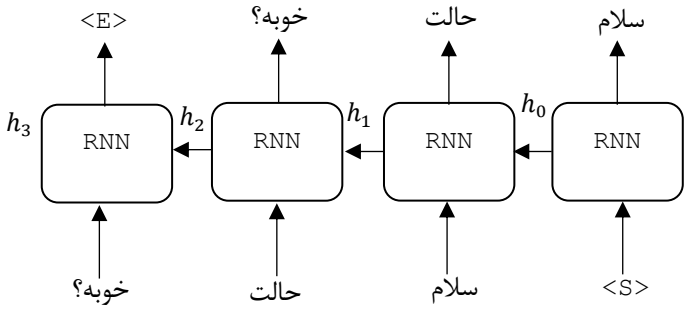
فهرست مطالب

شرح تمرین

هدف از این تمرین طراحی یک مدل زبانی **سطح-کاراکتر**[1] با استفاده از شبکه‌های عصبی بازگشتی[2] (RNN) می‌باشد. بر خلاف مدل‌های زبانی سطح-کلمه[3] در این نوع از مدل‌های زبانی می‌خواهیم با داشتن کاراکترهای قبلی کاراکتر بعدی را پیش‌بینی کنیم. مدل‌های زبانی‌ای که از شبکه‌های عصبی بازگشتی بهره می‌برند از معماری Many-to-Many استفاده می‌کنند (یعنی هم ورودی و هم خروجی دنباله هستند). در این نوع از مدل‌های زبانی خروجی همواره یک یا چند واحد زمانی از ورودی عقب‌تر است (به عبارتی نوع یادگیری self-supervised است). در شکل ۱ و شکل ۲ به ترتیب یک مدل زبانی سطح کاراکتر و سطح کلمه را مشاهده می‌کنید (h_iها بردارهای مخفی را نمایش می‌دهند). به دلیل نیاز شدید به حافظه برای طراحی مدل زبانی سطح-کلمه، در این تمرین از این مدل صرف نظر شده است. توجه کنید که هدف اصلی در این تمرین این است که با مدل زبانی سطح-کاراکتر یک **جمله** را پیش‌بینی کنید. مانند تمرین‌های قبل از مجموعه داده‌ی اخبار باشگاه خبرنگاران جوان و فارس‌نیوز که در سال ۱۳۹۷ تهیه و جمع‌آوری شده است بهره می‌گیریم. در ادامه مجموعه داده، نحوه طراحی مدل و همچنین ورودی و خروجی مدل را تشریح می‌کنیم.



شکل ۱ مدل زبانی سطح-کاراکتر



شکل ۲ مدل زبانی سطح-کلمه

[1] Character-level
[2] Recurrent Neural Network
[3] Word-level

بخش ۱: شناخت داده‌ها

ابتدا فایل `News.rar` را از طریق این لینک دانلود کنید. این فایل متشکل از دو فایل `train.csv` و `test.csv` می‌باشد. این دو فایل حاوی شش ستون می‌باشد که سه ستون `text`، `title` و `category` که به ترتیب نمایش‌دهنده‌ی متن خبر، عنوان خبر و دسته‌بندی خبر می‌باشند، دارای اهمیت هستند. فایل `train.csv` حاوی ۱۱۷۱۹۰ نمونه آموزشی و فایل `test.csv` حاوی ۲۱۱۰۴ نمونه آزمایشی می‌باشند. در ضمن، `encoding` این دو فایل `utf-8` می‌باشد.

توجه کنید که داده آموزشی `train.csv` را با یک نسبت معقول مانند ۷۰ به ۳۰ به دو بخش آموزش و اعتبارسنجی تفکیک کنید. از داده اعتبارسنجی برای یافتن پارامترهای غیر قابل تنظیم[۴] و همچنین یافتن بهترین مدل استفاده کنید.

بخش ۲: نوع شبکه عصبی بازگشتی

همان طور که می‌دانید شبکه‌های عصبی بازگشتی با معماری‌های گوناگونی طراحی شده‌اند. در این تمرین شما مجاز به استفاده از دو نوع از آن‌ها هستید.

الف- `Long Short Term Memory (LSTM)`

ب- `Gated Recurrent Units (GRU)`

توجه کنید که امکان این که چند لایه شبکه عصبی بازگشتی را زیر هم `stack` کنید نیز وجود دارد. همچنین در صورت استفاده از معماری‌های پیشرفته -وابسته به نوع آن- نمره امتیازی تعلق می‌گیرد.

بخش ۳: پیش‌پردازش داده

برای آموزش دسته‌بند و اعمال داده به آن، ابتدا باید داده‌ی مورد نظر پیش‌پردازش شود. با دنبال کردن مراحل زیر می‌توانید داده را پیش‌پردازش کنید.

۱. متن اخبار را پاکسازی کنید. بدین منظور، تمامی کلمات انگلیسی را حذف نمایید. همچنین کاراکترهای خاص (مانند ❋) و علائم نگارشی (به جز نقطه، نقطه ویرگول، علامت سوال) را نیز حذف کنید. در پایان این مرحله، متن هر خبر باید فقط حاوی حروف فارسی، اعداد و سه علامت نگارشی خاص (نقطه و علامت سوال، علامت سوال) باشد.

۲. در متن کلیه اخبار، به جای اعداد، کاراکتر `N` را قرار دهید.

۳. به ابتدا و انتهای کلیه جملاتِ مجموعه داده کاراکترهای **آغاز** و **پایان** را اضافه کنید. برای این منظور از کاراکتر خاص (`\n`) برای آغاز و از کاراکتر خاص (`\t`) برای پایان استفاده کنید.

۴. متن اخبار را به **کاراکترها** تجزیه کرده و تعداد کاراکترها را گزارش کنید. هدف ما نمایش سطح-کاراکتر از یک جمله می‌باشد.

[4] `Non-tunable`

۵. این بخش توکن کردن[5] متن نام دارد. برای این کار ابتدا کاراکترها را در یک لیست براساس حروف الفبا مرتب کنید. سپس دو دیکشنری `char2index` و `index2char` را به گونه‌ای بسازید که هر کاراکتر به یک عدد منحصر به فرد نگاشت شود و بالعکس. اکنون نمایش سطح کاراکتر را با استفاده از `index`ها نمایش دهید (توجه کنید که `\n` و `\t` نیز باید توکن شوند.).

۶. در داده آموزشی میانگین طول اخبار را به دست آورید و اخباری که دارای طولی بیش از طول میانگین می‌باشند را کنار بگذارید. توجه کنید که میانگین‌گیری را بر اساس تعداد کلمات انجام دهید. (البته به جای این کار می‌توانید جملات را به اندازه‌های کوچک‌تر تقسیم کنید. برای تقسیم جملات به اندازه کوچک‌تر می‌توانید آن‌ها را به سایز `n` تقسیم کنید. به این صورت که با یک طول گام مشخص مانند `m` بر روی جمله حرکت کنید و هر بار یک دسته `n`تایی از کلماتِ آن جمله را به عنوان جمله جدید اتخاذ کنید. همچنین اعداد `m` و `n` هر بار می‌توانند در یک بازه مشخص و به صورت تصادفی اختیار شوند)

۷. این بخش بردار کردن[6] متن نام دارد. ساده‌ترین روش نمایش متن استفاده از روش نمایش `One-hot` می‌باشد. در این روش برای نمایش هر توکن بدین صورت عمل می‌شود که یک بردار ۰ و ۱ به طول کل توکن‌ها ساخته می‌شود که تمامی درایه‌های آن به جز داریه‌ای که `index` آن با `index` آن توکن برابر است، صفر است. برای نمایش یک متن می‌توانید تمامی توکن‌ها را به صورت یک بردار درآورید و سپس با کنار هم قرار دادن این بردارها یک ماتریس بسازید که نمایش دهنده کل متن یا جمله می‌باشد.

۸. در صورتی که حافظه شما محدود است و نمی‌توانید کلیه اخبار را در حافظه قرار دهید، داده‌ها را حین فرایند آموزش و به صورت دسته‌ای بردار کنید. برای این کار می‌توانید از `data generator`ها یا `data loader`ها استفاده کنید.

بخش ۴: طراحی مدل زبانی

برای این مدل زبانی یک کلاس به نام `LanguageModel` ایجاد کنید. این کلاس از توابع زیر تشکیل شده است (فرایند آموزش مدل زبانی کاملا از کلاس `LanguageModel` مستقل است و شما باید آن را به صورت مجزا انجام دهید):

- **تابع `__init__`**

  این تابع شامل یک متغیر ورودی به نام `lm_checkpoints` است که مسیر وزن‌های مدل آموزش دیده در دیسک سخت را نشان می‌دهد.

- **تابع `lm_unit`**

  این تابع مدل زبانی را با استفاده از وزن‌های آموزش دیده مجددا بازسازی می‌کند. ورودی‌های این تابع `initi_states` و `weights` است که در حالت عادی برابر `None` هستند. اگر در فرایند آموزش شبکه‌های بازگشتی از حالات مخفی اولیه صفر استفاده کرده‌اید نیازی به `initi_states` نیست. همچنین برای ساخت مدل زبانی، این تابع را در تابع **`__init__`** فراخوانی کنید و وزن‌های مدل را با متغیر `weights` به آن پاس دهید. خروجی تابع یک مدل زبانی آموزش دیده خواهد بود.

- **تابع `get_next_states_and_output`**

  این تابع با داشتن بخش ابتدایی جمله (`Prefix`)، کاراکتر بعدی را حدس می‌زند. مثلا فرض کنید `Prefix` برابر "انتخابات آمریک" باشد. در این صورت کاراکتر بعدی که بیشترین احتمال را دارد "ا" می‌باشد. اما توجه کنید که شما

[5] `Tokenizing`
[6] `Vectorizing`

باید کل `prefix` را با یک یا دو بردار مخفی (به عنوان مثال در LSTM این دو بردار `cell state` و `hidden state` هستند) در زمان `t` نمایش دهید (نه به صورت رشته) که این بردار خود از مدل زبانی حاصل شده است و با دادن آن به تابع، احتمال هر یک از کاراکترها (به صورت یک بردار) و همچنین بردار اولیه در زمان `t+1` را از تابع خروجی بگیرید. توجه کنید که معمولا حالات مخفی در زمان `t=0`، بردار صفر می‌باشد.

- **تابع `prefix_to_hiddens`**

این تابع برخلاف تابع فوق، با داشتن بخش ابتدایی جمله (`Prefix`) به صورت **رشته**، بردار (های) مخفی حاصل شده از شبکه‌ی عصبی بازگشتی را برمی‌گرداند. لذا توجه کنید که ورودی تابع یک رشته و خروجی آن یک (دو) بردار می‌باشد.

- **تابع `generate_new_sample`**

این تابع با گرفتن یک `prefix` با معنی به صورت رشته، یک جمله را تولید می‌کند. خروجی این تابع نیز به صورت یک رشته است که جمله‌ی کامل شده را نمایش می‌دهد. این تابع بعدا در بخش ارزیابی مورد استفاده قرار می‌گیرد.

- **تابع `get_probability`**

این تابع با داشتن یک `prefix` به صورت بردار (بردارهای) اولیه، کاراکتر با بیشترین احتمال را یافته و خود کاراکتر و احتمال آن را به عنوان خروجی برمی‌گرداند.

- **تابع `get_overall_probability`**

این تابع با گرفتن یک جمله به صورت کامل، احتمال رخداد آن را در مدل زبانی محاسبه می‌کند. اگر $c_1, \dots, c_m$ کاراکترهای جمله ورودی باشند می‌توانید از رابطه زیر برای محاسبه احتمال رخداد کل این جمله استفاده کنید (اگر عدد به دست آمده خیلی کوچک بود می‌توانید از لگاریتم احتمال استفاده کنید).

$$P(c_1, \dots, c_m) = P(c_1) \prod_{i=2}^{m} P(c_i | c_1, \dots, c_{i-1})$$

برای سهولت می‌توانید از تابع `get_probability` برای محاسبه این احتمال استفاده کنید.

متناسب با نیاز خود و سبک برنامه‌نویسی که دارید، می‌توانید توابع دیگری را نیز به کلاس اضافه کنید. اما به خاطر داشته باشید توابع فوق حتما باید در کلاس موجود باشند و به درستی کار کنند. پاسخ شما به تمرین بر اساس این توابع و هر گونه عملی که از طریق آن‌ها انجام شود، ارزیابی می‌گردد. **همچنین در صورتی که نیاز داشتید تا ورودی‌ها و خروجی‌های توابع را کم یا زیاد کنید حتما نحوه عملکرد آن توابع را مستند کنید.**

مجددا لازم به ذکر است که فرایندهایی نظیر تمیز کردن داده، اندیس کردن، آموزش مدل و ذخیره‌سازی وزن‌های مدل از کلاس فوق مستقل است و باید به صورت جداگانه انجام پذیرد. به همین دلیل این دو بخش را در دو `notebook` مجزا انجام دهید.

بخش ۵: ارزیابی مدل روی داده‌های آموزش، اعتبارسنجی و آزمایش

برای ارزیابی مدل از معیارها و آنالیزهای زیر استفاده کنید.

- معیار آشفتگی[7]
- نرخ خطای کاراکتر[8] (CER)

در این بخش، باید **یک تابع با نام evaluate** را پیاده سازی کنید. که دو مورد فوق را انجام دهد. این تابع یک جمله را به عنوان ورودی دریافت می‌کند و پس از پیش‌پردازش، داده موردنظر را به مدل اعمال کرده و خروجی را ارزیابی می‌کند.

همچنین با در نظر گرفتن کلمه‌ی اول تمامی جملات مجموعه داده آزمایش به عنوان `prefix` و با استفاده از تابع **generate_new_sample** جملات جدیدی را تولید کرده و در فایل `new_samples.txt` ذخیره کنید. همچنین احتمال کلی رخداد هر نمونه جدید را نیز با استفاده از **get_overall_probability** در فایلی به نام `probabilities.txt` ذخیره کنید.

در بخش آموزش مدل انتظار می‌رود با روش‌هایی نظیر cross validation، grid search و ... بهترین مدل انتخاب شده باشد. توجه کنید که لازم است خروجی به دست آمده از بخش ارزیابی را تحلیل کنید.

شرح مستندسازی

مستندسازی یک تکه کد، به دیگر توسعه‌دهندگان در فهم آن کمک می‌کند. در این تمرین از شما تقاضا داریم یک فایل کوتاه در قالب `pdf` در شرح کدهای خود بنویسید. یک تا دو صفحه کافی است. لطفا مختصر توضیح دهید. برای هر تابعی که نوشته‌اید، به طور مختصر نحوه کارکرد آن را گزارش دهید. همچنین ورودی و خروجی (در صورتی که عینا مطابق تمرین نیست و یا پارامتر اضافه‌ای دارد) را ذکر نمایید. **در مستندسازی حتما نام و نام خانوادگی خود را به همراه شماره دانشجویی‌تان ذکر نمایید.**

تقسیم‌بندی نمرات برای ارزیابی

خواسته تمرین	نمره
تابع `lm_unit`	۱۵
تابع `get_next_states_and_output`	۱۵
تابع `generate_new_sample`	۱۰
تابع `get_probability`	۱۰
تابع `get_overall_probability`	۱۰
پاکسازی و آموزش مدل زبانی	۱۵
ارزیابی مدل و تحلیل خروجی	۱۵
مستندسازی	۱۰
مجموع	۱۰۰

[7] Perplexity
[8] Character Error Rate

نحوه ارسال پاسخ

پاسخ شما به این تمرین باید در قالب یک فایل فشرده (zip) باشد که در سامانه courses بارگذاری می‌گردد. این فایل شامل موارد زیر است:

- فایل‌هایی با پسوند py. و یا ipynb. که شامل کد مربوط به پیاده‌سازی توابع هستند. لازم است به وضوح مشخص شود که هر بخش از کد شما مربوط به پاسخ کدام بخش از تمرین است. برای این کار، لطفا به یکی از روش‌های زیر عمل نمایید:

- درج comment در فایل py.
- درج کدهای markdown در notebook
- استفاده از فایل‌های جدا برای هر بخش از تمرین

توجه: کد برنامه شما باید به زبان پایتون ۳ نوشته شود.

یک notebook با نام Preparing_and_training.pynb که شامل کدهای آماده‌سازی داده و آموزش مدل زبانی می‌باشد.

فایل LanguageModel.py که حاوی کلاس مدل زبانی است.

وزن‌های مدل زبانی آموزش دیده (وابسته به کتابخانه مورد استفاده، پسوند فایل متفاوت است).

فایل‌های probabilities.txt و new_samples.txt

دو فایل pickle. که دیکشنری‌های char2index و index2char را در خود جای داده است.

یک فایل با نام docs.pdf که در آن مستندسازی توابع قرار دارد.

- هر گونه فایل دیگری که برای بارگذاری مدل شما مورد نیاز است. (مجموعه داده را دوباره بارگذاری نکنید)

لطفا در صورت وجود هر گونه سوال از طریق ایمیل زیر آن را مطرح بفرمایید:

sadeghi.hamidreza1400@gmail.com

توجه: مهلت ارسال تمرین تا ساعت ۲۴ روز یکشنبه ۱۳۹۹/۰۹/۹ می‌باشد و پاسخ به تمرین پس از این زمان پذیرفته نیست.

با آرزوی موفقیت

حمیدرضا صادقی