به نام خدا

تمرین اول درس پردازش زبان طبیعی

عنوان: آموزش مدل‌های زبانی

استاد درس:

دکتر اکبری

موعد تحویل: ۱۳۹۹/۰۷/۲۶

فهرست مطالب

شرح تمرین

در این تمرین از شما می‌خواهیم با دریافت یک پیکره از اخبار باشگاه خبرنگاران جوان و فارس‌نیوز در سال ۱۳۹۷، یک مدل زبانی فارسی را طراحی نمایید. در ادامه، ابتدا در مورد پیکره توضیحاتی ارائه می‌شود؛ سپس مراحل حل تمرین و خروجی مورد انتظار در هر بخش ذکر می‌گردد.

بخش ۱: شناخت داده‌ها و آماده‌سازی

در این بخش، می‌خواهیم با بررسی داده‌ها، آن‌ها را برای آموزش مدل آماده کنیم. بدین منظور، ابتدا فایل `corpus.zip` را از طریق لینک زیر دانلود کنید:

https://drive.google.com/file/d/1gRDsuHlc6WDUpAAQ1pkHGmIlKdf-5Bzl/view?usp=sharing

در این فایل دو فایل فشرده دیگر با نام‌های `train.rar` و `valid.rar` موجود است که هر کدام حاوی یک فایل `json` هستند. از این فایل‌ها به ترتیب به عنوان پیکره آموزشی[1] و پیکره اعتبارسنجی[2] استفاده خواهید کرد. هر فایل `json` حاوی لیستی از اخبار است که به شکل رشته ذخیره شده‌اند. برای بارگذاری اخبار، لازم است در هنگام خواندن فایل‌ها، مقدار پارامتر `encoding` را برابر `utf-8` قرار دهید. پس از بارگذاری اخبار، شما باید مراحل زیر را انجام دهید:

۱. متن اخبار موجود در `train.json` را نرمال کنید. بدین منظور، تمامی کلمات انگلیسی را حذف نمایید. همچنین کاراکترهای خاص (مانند *) و علائم نگارشی (به جز نقطه، علامت سوال) را نیز حذف کنید. در پایان این مرحله، متن هر خبر باید فقط حاوی حروف فارسی، اعداد و دو علامت نگارشی خاص (نقطه و علامت سوال) باشد.

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

۳. متون اخبار را به توکن‌ها تجزیه کنید. **تعداد کل توکن‌ها** و **تعداد کل توکن‌های مجزا (کلمات)** را بدست آورید و گزارش نمایید.

۴. میزان رخداد هر کلمه در کلیه اخبار را محاسبه نمایید و **۱۰۰۰۰ کلمه با بیشترین میزان رخداد** را در یک فایل به نام `most_frequent.txt` ذخیره نمایید. هر خط فایل شامل یک کلمه باشد؛ به طوری که کلمه‌ای که بیشترین تکرار را دارد، در خط اول و کلمه‌ای که کمترین تکرار را دارد در خط آخر نوشته شود.

۵. پس از محاسبه ۱۰۰۰۰ کلمه پرتکرار، میزان پوشش کل توکن‌ها توسط این کلمات را بدست آورید؛ یعنی محاسبه کنید که این کلمات چند درصد توکن‌ها را تشکیل می‌دهند.

۶. در کلیه جملات، به جای هر کلمه که جزء ۱۰۰۰۰ کلمه پرتکرار نیست، نماد `UNK` را قرار دهید. (توجه کنید که اعداد تا پایان این مرحله به شکل `N` نمایش داده شده و حذف نمی‌شوند).

۷. با توجه به رخداد کاراکترهای نقطه و علامت سوال، جملات متن هر خبر را تفکیک کنید. جملات کلیه اخبار را در یک فایل متنی ذخیره کنید؛ به نحوی که هر جمله در یک خط آن فایل نوشته شده باشد.

---

[1] `Training`
[2] `Validation`

۸. عده‌ای از پژوهشگران بر این باور هستند که توزیع رخداد کلمات از توزیع power law پیروی می‌کند؛ این بدان معنی است که اگر یک نمودار رسم کنیم که محور افقی آن لگاریتم رخداد کلمات و محور عمودی لگاریتم رتبه از نظر تعداد رخداد باشد، نمودار تقریبا خطی خواهد بود. با رسم نمودار مذکور برای پیکره آموزشی، این موضوع را بررسی نمایید. (برای این کار فقط ۵۰۰۰ کلمه اول پرتکرار را در نظر بگیرید).

بخش ۲: طراحی مدل زبانی

در این بخش، با استفاده از داده‌هایی که در بخش گذشته آماده‌سازی نموده‌اید، باید دو مدل زبانی در سطح کلمه[3] طراحی نمایید. این مدل‌ها باید در قالب یک کلاس پیاده‌سازی شوند. یک شیء از این کلاس باید یک پیکره را در قالب یک فایل متنی که هر خط آن حاوی یک جمله است، دریافت نماید (مانند خروجی بخش ۱) و بر اساس پارامترهای دیگری که دریافت می‌کند، یک مدل زبانی را آموزش دهد. در ادامه جزئیات توابع این کلاس ذکر می‌گردد.

- **تابع `__init__`**

این تابع برای دریافت پارامترهای موردنظر کاربر استفاده خواهد شد. این پارامترها عبارتند از:

- `n`: یک عدد صحیح. اگر ۱ باشد مدل `unigram` آموزش داده شود و اگر ۲ باشد `bigram`

  امتیازی ۱: اگر `n` برابر ۳ باشد، مدل `trigram` پیاده‌سازی شود.
- `smoothing`: اگر برابر `False` باشد، مدل `smoothing` انجام ندهد. اگر برابر `laplace` باشد، مدل `laplace smoothing` را پیاده‌سازی کند.

  امتیازی ۲: اگر برابر `kneser-ney` باشد، آن را پیاده‌سازی کند.
- `corpus_dir`: یک متغیر از نوع رشته که آدرس فایل پیکره در آن قرار می‌گیرد.

- **تابع `train`**

این تابع ابتدا پیکره را می‌خواند. سپس متناسب با مقدار پارامتر `n` – که `unigram` یا `bigram` بودن مدل را تعیین می‌کند – مقادیر مربوط به رخداد کلمات را ذخیره می‌نماید. توجه کنید که بعد از آموزش مدل، دیگر به پیکره دسترسی نخواهید داشت. لذا مدل باید پس از آموزش بتواند احتمالات را بر اساس آنچه قبلا ذخیره کرده است، محاسبه نماید. (این محاسبات در تابع `prob` پیاده‌سازی می‌شود). به عبارت دیگر باید به نحوی تعداد رخداد کلمات به تنهایی و در کنار یکدیگر را ذخیره نماید که در صورت لزوم، با سرعت مناسب به آن‌ها دسترسی داشته باشد.

- **تابع `prob`**

پس از آموزش مدل، باید بتوان این تابع را فراخوانی کرد. کاربر باید بتواند متناسب با اینکه مدل `unigram` یا `bigram` است، احتمال رخداد یک دنباله از کلمات (جمله) را محاسبه نماید. ورودی این تابع، یک رشته است. لذا تابع باید با توجه به رخداد کاراکتر فاصله، دنباله کلمات آن رشته را بدست آورد.

- **تابع `generate`**

پس از آموزش مدل، می‌توانید احتمال رخداد هر کلمه بر اساس کلمات قبل از آن در جمله را محاسبه نمایید. تابع `generate` از این قابلیت استفاده می‌کند تا متن تولید نماید. برای این منظور، کافی است به این تابع یک دنباله از کلمات داده شود. تابع کلمه بعدی را تحویل می‌دهد. مثلا فرض کنید دنباله «من می‌خواهم به مدرسه» به صورت یک

[3] `Word-level language model`

رشته به تابع داده شود. تابع این رشته را به شکل یک لیست از کلمات در می‌آورد (من، می‌خواهم، به، مدرسه). سپس بر اساس این لیست، احتمال رخداد هر کلمه بعد از این دنباله را محاسبه می‌کند. فرض کنید احتمال رخداد «بروم» به شرط رخداد «من می‌خواهم به مدرسه» بیشتر از احتمال رخداد هر کلمه دیگری باشد. در این صورت تابع `generate` کلمه «بروم» را برمی‌گرداند. بر اساس فرض `markov` و با توجه به نحوه محاسبه احتمال شرطی در مدل‌های `unigram` و `bigram`، این تابع را پیاده‌سازی نمایید.

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.

در نهایت پس از ایجاد کلاس، حداقل یک نمونه از نحوه فراخوانی هر تابع آن را در کد قرار دهید. این قسمت نمره ندارد و فقط برای درک بهتر کد شما مورد استفاده قرار خواهد گرفت.

بخش ۳: ارزیابی مدل‌ها روی داده‌های اعتبارسنجی

در این بخش، باید **یک تابع با نام `evaluate`** به کلاسی که در بخش ۲ نوشته‌اید، اضافه کنید. این تابع یک فایل متنی – مانند فایلی که در بخش ۱ آماده نمودید – دریافت می‌کند که در هر خط آن، یک جمله نوشته شده است. جملات این فایل باید تنها حاوی توکن‌هایی باشند که مدل شما آن‌ها را به رسمیت می‌شناسد؛ یعنی ۱۰۰۰۰ کلمه پرتکرار، `UNK` برای کلمات ناشناخته و `N` برای اعداد. با این فرض، تابع `evaluate` از تابع `generate` استفاده می‌کند تا با دریافت هر کلمه، روند بصری‌سازی شانون را برای آن انجام دهد و یک جمله تولید کند. بدیهی است مدل شما دقیقا همان جمله اصلی را تولید نخواهد کرد. با توجه به این موضوع، از تابع `evaluate` انتظار داریم برای هر یک از مدل‌ها، میزان `word-error rate (WER)` برای هر نمونه (جمله) در فایل دریافتی را محاسبه نماید و سپس با استفاده از آن مقدار میانگین این خطا (`average WER`) را نیز بدست آورد.

پس از پیاده‌سازی تابع `evaluate`، اخبار موجود در فایل `valid.json` را شکل یک فایل متنی متناسب با تابع `evaluate` ذخیره نمایید. سپس میزان `average WER` مربوط به آن را گزارش نمایید. توجه کنید که در آماده‌سازی فایل `valid.json` کلمات پرتکرار دوباره محاسبه نمی‌شوند؛ اما بقیه مراحل بخش ۱ عینا تکرار می‌گردد.

برای این بخش از شما انتظار داریم علاوه بر پیاده‌سازی، میزان `average WER` را برای مدل‌های `unigram` ، `bigram` و `trigram` (در صورت پیاده‌سازی قسمت امتیازی ۱) گزارش نمایید.

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	نوع
بخش ۱: شناخت و آماده‌سازی داده‌ها	۲۰	اصلی
تابع `train` (مدل `unigram`)	۱۰	اصلی
تابع `train` (مدل `bigram`)	۲۰	اصلی
تابع `train` (مدل `trigram`)	۱۰	امتیازی
`Laplace smoothing`	۱۰	اصلی
`Kneser-ney smoothing`	۱۵	امتیازی
تابع `prob`	۱۰	اصلی
تابع `generate`	۱۰	اصلی
ارزیابی مدل‌ها روی داده‌های اعتبارسنجی	۱۵	اصلی
مستندسازی	۵	اصلی
مجموع اصلی	۱۰۰	
مجموع امتیازی	۲۵	

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 فایل متنی با نام `most_frequent.txt` که حاوی ۱۰۰۰۰ کلمه با بیشترین میزان رخداد است.

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

- هر گونه فایل دیگری که برای بارگذاری مدل شما موردنیاز است. (پیکره را دوباره بارگذاری نکنید)