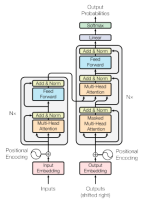


تمرین
نگاشت واژگان و دسته‌بندی متن

۱ پرسش‌ها

۱. برنامه‌ای را تصور کنید که یک بازی ساده براساس واژگان را پیاده‌سازی می‌کند. این بازی در هر مرحله، دو واژه را پیشنهاد می‌دهد، نام این دو را خاص‌واژه می‌گذاریم. خاص‌واژگان، می‌بایست در فضای تعبیه[1] از یکدیگر فاصله‌ای حداقل برابر با α داشته باشند. بازیکن[2] یک واژه را پیشنهاد می‌دهد، این واژه در صورتی مورد قبول واقع می‌شود که فاصله‌ای کم‌تر از β را با حداقل یکی از واژگان موجود داشته باشد. این روند تا جایی ادامه می‌یابد که فاصله‌ی بین تمامی واژگان موجود کم‌تر از β باشد، بنابراین ممکن است با واژه‌ی اول معرفی شده توسط بازیکن، بازی به پایان برسد. یا در حالت کلی‌تر مجموعه‌ای از واژگان، به ترتیب توسط بازیکن پیشنهاد می‌شود تا در نهایت فاصله‌ی بین تمامی واژگان موجود کم‌تر از β گردد. برای نگاشت واژگان به فضای تعبیه چه روشی را پیشنهاد می‌دهید؟ چرا؟ این روش چه خصوصیاتی را باید داشته باشد؟ (در این راستا می‌توانید نگاشت E_2 مشخص شده در بخش ۳.۱.۲ را مطالعه کنید و مطالبی مانند چگونگی محاسبه وزن‌ها را کامل‌تر توضیح دهید)

۲ برنامه‌نویسی

۱.۲ دسته‌بندی

نکته: برای جلوگیری از مشکلات سخت‌افزاری حتما برنامه را روی کولب اجرا کنید.

۱.۱.۲ مجموعه‌داده

مجموعه‌داده IMDB از ۵۰۰۰۰ نظر در مورد فیلم‌ها، که به دو دسته‌ی مثبت و منفی تقسیم می‌شود تشکیل شده است. برای دست‌یابی به این مجموعه‌داده می‌توان از پکیج‌های مختلفی استفاده کرد. با استفاده از pytorch می‌توان آن را به شکل زیر دریافت کرد (توجه داشته باشید که خروجی پکیج‌های دیگر می‌تواند متفاوت باشد - در این حالت دادگان به دو قسمت مساوی برای آموزش و آزمون تقسیم شده‌اند).

```
// Getting IMDB dataset
>>> pip install 'portalocker > = 2 . 0 . 0' # remove spaces in quotation

>>> from torchtext.datasets import IMDB
>>> train_iter = IMDB(split='train')
```

[1] Embedding space
[2] Player

```
>>> test_iter = IMDB(split='test')
>>> for label, line in train_iter:
>>> ... print('comment:' + line + ' its corresponding label: ' + label)
```

نکات: به صورت تصادفی نصف مجموعه‌داده را حذف کنید (این مسئله صرفا برای ساده‌تر کردن اجراها بیان شده است، در صورتی که محدودیت سخت‌افزاری ندارید، مجموعه‌داده را به همین شکل نگه دارید).
به هنگام اجرای کد بالا فضاهای خالی داخل نقل‌قول[۳] (خط دوم) را حذف نمایید.
ممکن است اجرای کد بالا با خطا روبه‌رو شود، در این‌صورت کد را یک‌بار دیگر اجرا کنید.

۲.۱.۲ پیش‌پردازش

براساس هدف مسئله (دسته‌بندی) مجموعه پیش‌پردازش‌های لازم را روی مجموعه‌داده انجام دهید.

- آیا حذف کردن واژگانی که تنها در یک نظر اتفاق افتاده‌اند می‌تواند به پروسه‌ی دسته‌بندی کمک کند (این پرسش را با در نظر گرفتن مراحل بعدی پاسخ دهید)؟
- واژگان توقف[۴] را از مجموعه‌داده حذف کنید و تمامی حروف را با حروف کوچک جایگزین کنید.

۳.۱.۲ نگاشت به فضای تعبیه

شبکه‌های عصبی بر اساس ورودی‌هایی عددی کار می‌کنند، این مسئله باعث می‌شود تا نتوان دادگان غیرعددی مانند جملات را به صورت مستقیم به عنوان ورودی آن‌ها داد. برای این کار نگاشت‌های متعددی معرفی شده است که سعی در ارائه‌ی بردارهای نظیر واژگان به صورت عددی دارند. یکی از این نگاشت‌ها، تک‌نمود[۵] نام دارد که مجموعه واژگان اتفاق افتاده در سند را در نظر می‌گیرد و به ازای هر کدام از آن‌ها برداری تماما صفر را ایجاد می‌کند و تنها درایه‌ی هم‌شماره با آن واژه را برابر با یک قرار می‌دهد، این نگاشت را E_1 می‌نامیم.

$$E_1(w_i) = [x_n | x_n = 1 \ for \ n = i \ and \ x_n = 0 \ for \ n \neq i, \ \ n = 1, ..., N] \quad for \ w_i \in Dict \qquad (۱)$$

براساس E_1 می‌توان بسته واژگان[۶] هر جمله را به‌دست آورد. برای این کار بردارهای واژگان مختلف یک جمله که توسط نگاشت تک‌نمود به‌دست آمده‌اند با یکدیگر جمع می‌شوند:

$$BOW(s_i) = E_1(w_1) + ... + E_1(w_N) \ for \ s_i \in Dataset \qquad (۲)$$

خروجی این نگاشت، هر جمله را به صورت برداری نمایش می‌دهد. براساس نگاشت تک‌نمود و با استفاده از یک شبکه‌ی عصبی می‌توان بردارهای معادل دیگری برای واژگان هر جمله به‌دست آورد، این نگاشت word2vec نام دارد[۱] [۲] [۳]. در این نگاشت از یک شبکه‌ی عصبی به گونه‌ای استفاده می‌شود که احتمال رخداد یک واژه خاص در مجاورت واژگان دیگر سند در قالب وزن‌های شبکه به‌دست آید. برای این کار از یک شبکه عصبی با یک لایه‌ی مخفی[۷] خطی استفاده می‌شود (تعداد نورون[۸]های ورودی برابر با تعداد درایه‌های

[4] Stop words
[5] One hot
[6] Bag of words
[7] Hidden layer
[8] Neuron

بردار تک‌نمود واژگان و تعداد نورون‌های لایه‌ی مخفی برابر با بعد بردار نگاشت خواسته شده است). به هنگام آموزش این شبکه یک لایه‌ی خروجی با تابع فعال‌سازی Softmax خواهد داشت که هر واژه ورودی را به واژه دیگری در سند نگاشت می‌کند، این نگاشت براساس مجاورت دیگر واژگان در یک پنجره با سایز مشخص نسبت به واژه ورودی صورت می‌گیرد:

$$E_2(w_i) = [\theta_1, ..., \theta_h],\ y = f(E_1(w_i), \boldsymbol{\theta})\ for\ w_i \in Dict \qquad (۳)$$

از آن‌جایی که در نگاشت E_2 به واژگان مجاور یک واژه در سندهای مختلف توجه می‌شود، و از طرفی واژگانی که با یکدیگر شباهت معنایی دارند در محتوای یکسانی قرار می‌گیرند، می‌توان انتظار داشت که براساس معیاری مانند ضرب نقطه[9]ای، میزان شباهت بین واژگان مختلف مشخص شود.

$$Cosine\ similarity = \frac{E_2(w_i)E_2(w_j)}{||E_2(w_i)||\ ||E_2(w_j)||} \qquad (۴)$$

۴.۱.۲ خوشه‌بندی

بردار به‌دست‌آمده توسط BOW برای هر جمله را در نظر بگیرید، سپس براساس الگوریتم k means مجموعه جملات آموزش را به k خوشه[10] تقسیم کنید:

- تصمیم بگیرید که خوشه‌بندی باید روی دادگان آموزش و آزمایش جداگانه صورت گیرد یا در کنار هم؟ چرا؟
- تعداد خوشه‌ها k، را برابر با هفت در نظر بگیرید.
- آیا این عدد، بهترین انتخاب برای تعداد خوشه‌های مجموعه‌داده مورد نظر است؟ چرا؟ چگونه می‌توان در این مورد تصمیم گرفت؟ الگوریتمی بنویسید که بتواند تعداد خوشه مناسب را به‌دست آورد. **(اختیاری)**
- خوشه‌های به‌دست آمده را رسم کنید، برای این کار می‌توانید از PCA یا t-sne استفاده کنید.
- برای کاهش حجم دادگان از هر خوشه m_i جمله را به صورت یکنواخت[11] انتخاب کنید. بر چه اساسی m_i را برای هر خوشه محاسبه می‌کنید؟

$$New\ total\ number\ of\ sentences\ in\ the\ dataset: \quad M = \sum_{i=1}^{k} m_i \qquad (۵)$$

۵.۱.۲ دسته‌بندی

پس از کاهش تعداد جملات مجموعه آموزش، حال اقدام به دسته‌بندی می‌کنیم. مدل Word2vec آموزش داده شده توسط گوگل روی مجموعه‌دادگان خبری را در نظر بگیرید، برای این کار می‌توانید از پکیج gensim به شکل زیر استفاده کنید:

```
// Getting IMDB dataset
>>> import gensim.downloader as api
>>> wv = api.load('word2vec-google-news-300')
```

[9] Dot product
[10] Cluster
[11] Uniform

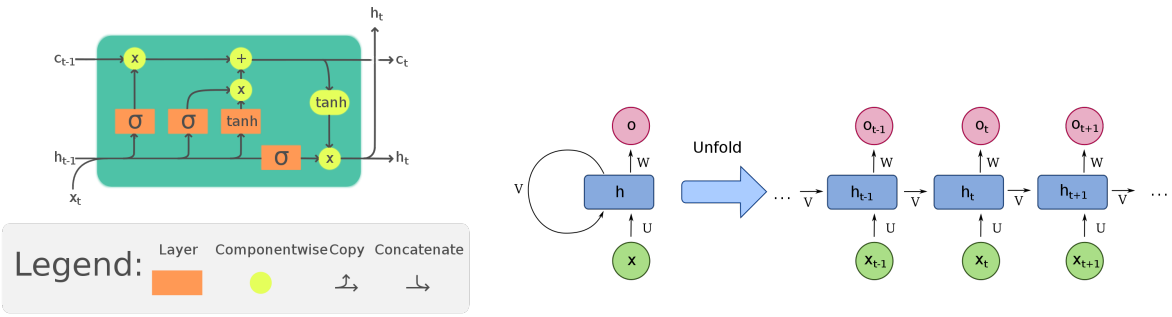
توجه کنید که این مدل بیش از یک گیگابایت حجم دارد، به همین خاطر برای دانلود و استفاده از آن حتما از کولب استفاده کنید. برای اطلاعات بیش‌تر می‌توانید به مستندات رسمی gensim مراجعه کنید. همان‌طور که پیش‌تر اشاره شده، بردارهای حاصل از نگاشت Word2vec را می‌توان با یکدیگر مقایسه نمود. نکتهی جالب توجه در این زمینه این است، که باتوجه به این موضوع و عددی بودن بردارهای حاصل شده، می‌توان روی آن‌ها عملیات‌های ریاضیاتی انجام داد و بردارهای جدیدی به‌دست آورد:

$$E_2(queen) = E_2(king) - E_2(man) + E_2(woman) \quad (۶)$$

برای مثال این مسئله با کم کردن بردار واژه man از king و اضافه کردن بردار woman به آن اتفاق می‌افتد، در نهایت برداری که در مجموعه بردارها به بردار حاصل شده نزدیک‌تر باشد بازگردانده می‌شود(بردار queen). این مورد را با استفاده از پکیج مذکور نشان دهید.

حال واژگان اتفاق افتاده در جملات را با بردارهای مدل آموزش‌داده شده گوگل جایگزین کنید و شبکه‌ی عصبی براساس مدل‌های زیر آموزش دهید(انتخاب ابرپارامتر[12]های شبکه به عهده دانشجو می‌باشد):

- شبکه عصبی کانولوشن [13] یک بُعدی
- شبکه‌ی عصبی بازگشتی[14]
- ترکیب شبکه بازگشتی و کانولوشن یک بعدی (**پیاده‌سازی اختیاری است، اما در گزارش خود می‌بایست نحوه‌ی ترکیب این دو شبکه را در قالب یک دسته‌بند توضیح دهید**)



(آ) (ب)

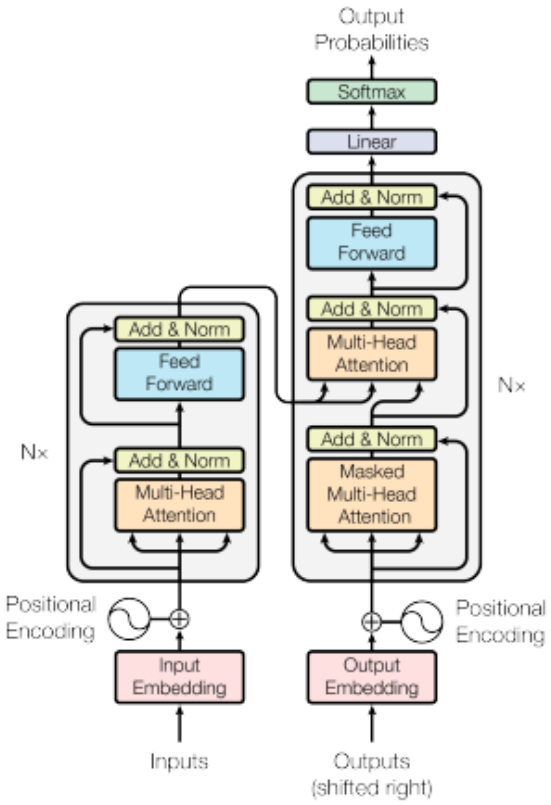
شکل ۱: نمایه‌ای از (آ) شبکه عصبی بازگشتی (ب) حافظه طولانی کوتاه مدت

۳ خوانش مقاله

مقاله‌ی مشهور «Attention is all you need» [۴] را مورد مطالعه قرار دهید و به پرسش‌های زیر پاسخ دهید:

[12] Hyperparameter
[13] Convolutional neural network
[14] Recurrent neural network (RNN)

۱. تصویر یکم در مقاله «Attention is all you need» [۴] را در نظر بگیرید (شکل ۱). این تصویر ساختار ترنسفورمر[15] را نشان می‌دهد که در مقاله توضیح داده شده است. تفاوت و تاثیر Post-LN با Pre-LN را در ترنسفورمر براساس مقاله‌ی «On Layer Normalization in the Transformer Architecture» [۵] مشخص کنید.



شکل ۲: نمایه‌ای از ساختار کلی ترنسفورمر

۴ نکات تحویل

۱. پاسخ خود را تحت پوشه‌ای به اسم NLP_NAME_ID و در قالب zip بارگذاری نمایید.

۲. این پوشه می‌بایست حاوی موارد زیر باشد:

- پوشه‌ای با نام code باشد که شامل برنامه‌ی نوشته/تغییر داده شده است.
- پوشه‌ای با نام doc که حاوی داکیومنت‌ها و فایل توضیحات می‌باشد.

۳. لازم به ذکر است که رعایت قوانین نگارشی حائز اهمیت خواهد بود.

[15] Transformer

مراجع

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol.26, 2013.

[3] Y. Goldberg and O. Levy, “word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method,” *arXiv preprint arXiv:1402.3722*, 2014.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol.30, 2017.

[5] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, “On layer normalization in the transformer architecture,” in *International Conference on Machine Learning*, pp.10524–10533, PMLR, 2020.