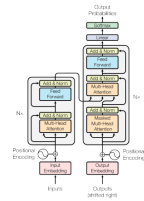




تمرین سوم نگاشت واژگان

تاریخ بارگذاری تمرین: ۱۴۰۲/۰۳/۲۲

تاریخ تحویل تمرین: ۱۴۰۲/۰۴/۲۰



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^۱ از یکدیگر فاصله‌ای حداقل برابر با α داشته باشند. بازیکن^۲ 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 خصوصیتی را باید داشته باشد؟ (در این راستا می‌توانید نگاشت E_2 مشخص شده در بخش ۳.۲.۲ را مطالعه کنید و مطالبی مانند چگونگی محاسبه وزن‌ها را کامل‌تر توضیح دهید)

۲ برنامه‌نویسی

۱.۲ شناسایی پرسش‌ها

مجموعه جملات مندرج در فهرست زیر را در نظر بگیرید، برخی از آن‌ها جملاتی خبری و برخی دیگر جملات پرسشی هستند. برنامه‌ای بنویسید که درخت تجزیه‌ی سازواری^۳ هر جمله را رسم کند و سپس تشخیص دهید جمله مورد نظر پرسشی است یا خیر. برای این کار می‌توانید از پکیج StanfordParser از مجموعه ابزار Nltk استفاده کنید.

- Do androids dream of electric sheep?
- How many androids escaped this time?
- Will have difficulty understanding how you happened to let us void your testing apparatus before the test began.
- This time was different.
- This has happened before.

¹Embedding space

²Player

³Constituency parse tree

مشخص کنید باتوجه به اطلاعات ارائه شده توسط درخت مورد نظر، چگونه می‌توان پرسشی بودن جملات را مشخص کرد؟

۲.۲ دسته‌بندی

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

۱.۲.۲ مجموعه‌داده

مجموعه‌داده IMDB از ۵۰۰۰۰۰ 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')
>>> 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 صرفاً 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^۴ (خط دوم) را حذف نمایید. ممکن است اجرای کد بالا با خطا روبه‌رو شود، در این صورت کد را یک‌بار دیگر اجرا کنید.

۲.۲.۲ 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^۵ را از مجموعه‌داده حذف کنید و تمامی حروف را با حروف کوچک جایگزین کنید.

^۵Stop words

۳.۲.۲ 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 ارائه بردارهای نظیر واژگان به صورت عددی دارند. یکی از این نگاشت‌ها، تک‌نمود^۶ نام دارد که مجموعه واژگان اتفاق افتاده در سند را در نظر می‌گیرد و به ازای هر کدام از آن‌ها برداری تماماً صفر را ایجاد می‌کند و تنها درایه‌ی هم‌شماره با آن واژه را برابر با یک قرار می‌دهد، این نگاشت را E_1 می‌نامیم.

$$E_1(w_i) = [x_n | x_n = 1 \text{ for } n = i \text{ and } x_n = 0 \text{ for } n \neq i, \quad n = 1, \dots, N] \quad \text{for } w_i \in \text{Dict} \quad (1)$$

بر اساس E_1 می‌توان بسته واژگان^۷ هر جمله را به دست آورد. برای این کار بردارهای واژگان مختلف یک جمله که توسط نگاشت تک‌نمود به دست آمده‌اند با یکدیگر جمع می‌شوند:

$$BOW(s_i) = E_1(w_1) + \dots + E_1(w_N) \quad \text{for } s_i \in \text{Dataset} \quad (2)$$

خروجی این نگاشت، هر جمله را به صورت برداری نمایش می‌دهد. بر اساس نگاشت تک‌نمود و با استفاده از یک شبکه‌ی عصبی می‌توان بردارهای معادل دیگری برای واژگان هر جمله به دست 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 به دست آید. برای این کار از یک شبکه عصبی با یک لایه‌ی مخفی^۸ خطی استفاده می‌شود (تعداد نوروں^۹های ورودی برابر با تعداد درایه‌های بردار تک‌نمود واژگان و تعداد نوروں‌های 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 بر اساس مجاورت دیگر واژگان در یک پنجره با سائز مشخص نسبت به واژه ورودی صورت می‌گیرد:

$$E_2(w_i) = [\theta_1, \dots, \theta_h], \quad y = f(E_1(w_i), \theta) \quad \text{for } w_i \in \text{Dict} \quad (3)$$

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 بر اساس معیاری مانند ضرب نقطه^{۱۰} ای، میزان شباهت بین واژگان مختلف مشخص شود.

$$\text{Cosine similarity} = \frac{E_2(w_i)E_2(w_j)}{\|E_2(w_i)\| \|E_2(w_j)\|} \quad (4)$$

۴.۲.۲ خوشه‌بندی

بردار به دست آمده توسط BOW برای هر جمله را در نظر بگیرید، سپس بر اساس الگوریتم k means مجموعه جملات آموزش را به k خوشه^{۱۱} تقسیم کنید:

^۶One hot

^۷Bag of words

^۸Hidden layer

^۹Neuron

^{۱۰}Dot product

^{۱۱}Cluster

• 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 مجموعه داده مورد نظر است؟ چرا؟ چگونه می‌توان در این مورد تصمیم گرفت؟ الگوریتمی بنویسید که بتواند تعداد خوشه مناسب را به دست آورد. (اختیاری)

• خوشه‌های به دست 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^{۱۲} انتخاب کنید. بر چه اساسی m_i را برای هر خوشه محاسبه می‌کنید؟

$$\text{New total number of sentences in the dataset : } M = \sum_{i=1}^k m_i \quad (5)$$

۵.۲.۲ دسته‌بندی

پس از کاهش تعداد جملات مجموعه آموزش، حال اقدام به دسته‌بندی می‌کنیم. مدل Word2vec آموزش داده شده توسط گوگل روی مجموعه دادگان 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')
```

توجه کنید که این مدل بیش از یک گیگابایت حجم دارد، به همین خاطر برای دانلود و استفاده از آن حتماً 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 با توجه به این موضوع و عددی بودن بردارهای حاصل شده، می‌توان روی آن‌ها عملیات‌های ریاضیاتی انجام داد و بردارهای جدیدی به دست آورد:

$$E_2(\text{queen}) = E_2(\text{king}) - E_2(\text{man}) + E_2(\text{woman}) \quad (6)$$

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 آموزش داده شده گوگل جایگزین کنید و شبکه‌ی عصبی براساس مدل‌های زیر آموزش دهید (انتخاب ابرپارامتر^{۱۳}های شبکه به عهده دانشجو می‌باشد):

• شبکه‌ی عصبی بازگشتی^{۱۴}

• شبکه عصبی بازگشتی دوطرفه^{۱۵} (اختیاری)

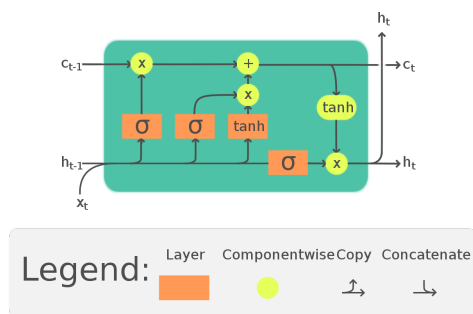
¹²Uniform

¹³Hyperparameter

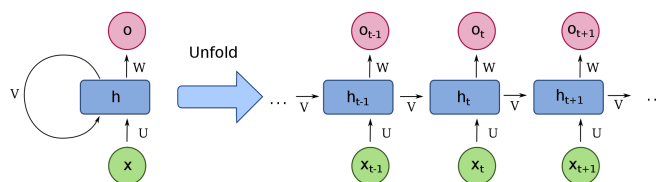
¹⁴Recurrent neural network (RNN)

¹⁵Bidirectional RNN

• حافظه طولانی کوتاه مدت^{۱۶} (اختیاری)



(ب)



(ا)

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

۳ پرسش‌هایی از گذشته

تاکنون تمرین‌های مختلفی را به انجام رسانده‌اید. حال که هر یک از آن‌ها را به خوبی بررسی کرده‌اید و به پاسخ‌های مختلفی برای آن‌ها دست پیدا کرده‌اید، در این بخش مجموعه پرسش‌هایی در رابطه با روش‌ها و نکات مطرح شده‌ی این تمرین‌ها مطرح می‌گردد (این پرسش‌ها سعی دارند تا مطالب عمیق‌تری را به خواننده ارائه نمایند):

۱.۳ تمرین صفرم

۱. برخی از دلایلی که در ارزیابی مدل‌های زبانی از F-score استفاده می‌شود را بیان کنید.

۲.۳ تمرین یکم

۱. مدل خالق^{۱۷} طراحی شده از جنس مدل خودهم‌بسته^{۱۸} می‌باشد. برخی از محدودیت‌های این نوع از مدل را مشخص کنید (برای مطالعه دقیق‌تر می‌توانید مقاله Limitations of autoregressive models and their alternatives [۴] را در نظر بگیرید).

۳.۳ تمرین دوم

۱. اثبات کنید گراف دوبخشی^{۱۹} ساخته شده، یک تناظر دوسویه^{۲۰} نیست. (اختیاری)

۲. در صورتی که محدودیت دوبخشی بودن از روی گراف مورد نظر برداشته شود، چه روابط دیگری به گراف می‌توان اضافه کرد؟ این روابط چه اطلاعاتی را فراهم می‌کنند؟

¹⁶Long short-term memory(LSTM)

¹⁷Generative

¹⁸Autoregressive

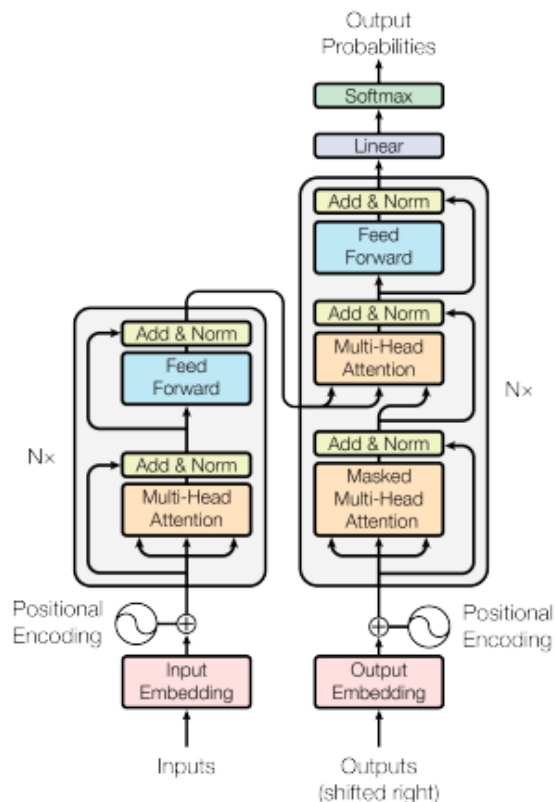
¹⁹Bipartite

²⁰Bijective

۴ خوانش مقاله

مقاله‌ی مشهور «Attention is all you need» [۵] را مورد مطالعه قرار دهید و به پرسش‌های زیر پاسخ دهید:

۱. تصویر یکم در مقاله «Attention is all you need» [۵] را در نظر بگیرید (شکل ۱). این تصویر ساختار ترنسفورمر^{۲۱} 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» [۶] مشخص کنید.



شکل ۲: نمایه‌ای از ساختار کلی ترنسفورمر

۵ 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 داکيومنت‌ها و فایل توضیحات می‌باشد.
۳. لازم به ذکر است که رعایت قوانین نگارشی حائز اهمیت خواهد بود.

²¹Transformer

- [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] C.-C. Lin, A. Jaech, X. Li, M. R. Gormley, and J. Eisner, “Limitations of autoregressive models and their alternatives,” *arXiv preprint arXiv:2010.11939*, 2020.
- [5] 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.
- [6] 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.