دانشکده مهندسی کامپیوتر

دکتر بهروز مینایی

بهار ۱۴۰۱

تمرین سری چهارم

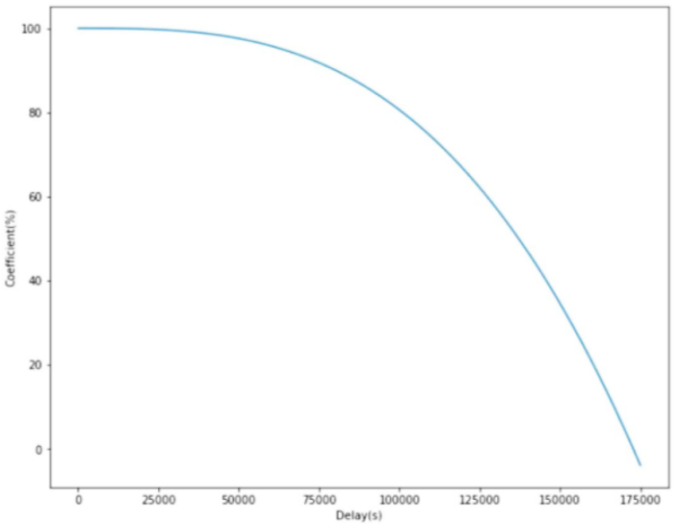
پردازش زبان و گفتار

محمد یارمقدم- مهسا انوریان - امیرحسین امینی مهر

تاریخ تحویل: ۱ تیر ۱۴۰۱ ساعت ۲۳:۵۹:۵۹

قوانین:

- سوالات این تمرین از مبحث «دسته‌بندی متن» می‌باشد و برای پاسخ به سوالات آن نیاز به دانش نسبی درمورد این مبحث دارید.
- این تمرین شامل ۳ سوال می‌باشد.۱ سوال تئوری و نوشتاری هستند و ۲ سوال عملی و شامل پیاده‌سازی هستند.
- درصورت وجود هرگونه سوال، در کلاس درس و یا در گروه تلگرامی درس بپرسید. (لطفا پی‌وی پیام ندهید.)
- هرگونه ایده گرفتن از تمرین دیگران و کدهای موجود در اینترنت که موجب تشابه غیرعادی و بالای کد شما با دیگری شود، تقلب محسوب می‌شود. درصورت مشاهده‌ی تقلب، نمره‌ی تمرین برای هر دو دانشجوی متخلف **صفر** منظور خواهد شد.
- لطفا برای انجام تمرین، زمان مناسب اختصاص دهید و انجام آن را به روزهای پایانی موکول نکنید.
- پاسخ ارسالی شما باید علاوه بر کدهای مربوط به هر سوال، شامل یک گزارش در قالب یک فایل PDF باشد که محتوای گزارش مربوطه توضیحات تکمیلی شما درخصوص هر سوال و اسکرین‌شات از نتیجه اجرای کدهای شما باشد.
- تمامی فایل‌های موردنیاز برای تمرین را به صورت یک فایل ZIP با فرمت شماره دانشجویی_نام و نام خانوادگی_HW4 نام‌گذاری کرده و ارسال کنید. (برای مثال HW4_NameFamily_98000000)
- تاخیر در ارسال تمرین‌ها بر اساس نمودار زیر محاسبه خواهد شد. محور افقی نمودار، مقدار تاخیر به ثانیه و محور عمودی، ضریب اعمالی در نمره تمرین است.



سوالات تئوری:

۱. تحقیق کنید که چگونه LSTM به جلوگیری از مشکل exploding/ vanishing گرادیان در شبکه‌های عصبی RNN کمک می‌کند.

سوالات عملی:

۱. دو سری زمانی در فایل‌های csv.train و csv.test در اختیار شما قرار داده شده است. فایل مربوط به csv.train شامل اطلاعات مربوط به ۱۰۰۰ روز است و فایل csv.test شامل اطلاعات مربوط به ۴۶۰ روز بعد است. با استفاده از لایه (های) simpleRNN و (لایه‌های دیگر) شبکه‌ای پیاده‌سازی کنید که با گرفتن مقادیر ۲۰ روز متوالی مقدار روز ۲۱ام را پیشبینی کند. از داده‌های موجود در csv.train برای آموزش شبکه و داده‌های csv.test برای ارزیابی شبکه استفاده کنید. خطایmean absolute error را بر روی مجموعه داده‌های آموزش و ارزیابی گزارش دهید. برای مجموعه‌داده ارزیابی (برای تمام ۴۴۰ روز ممکن) مقادیر پیش‌بینی شده و مقادیر واقعی را در یک نمودار نشان دهید . با تنظیم هایپرپارامترهای مدل، تلاش کنید خطای کمی بدست بیاورید.

۲. در این تمرین هدف یادگیری مدل های شبکه عصبی بر پایه attention برای ترجمه است. در این تمرین وظیفه شما ترجمه کلمات انگلیسی به کاستومایز شده latin است. در طی اینکار شما تجربه کار به GRU و attention را خواهد آموخت.

این زبان کاستومایزشده latin یک دگرگونی ساده از زبان انگلیسی با قوانین زیر است:

- اگر حرف اول کلمه صامت باشد، آنگاه حرف به انتهای کلمه منتقل می‌شود و حروف "ay" به انتهای آن اضافه میشود.

Team → eamtay

- اگر حرف اول کلمه صدادار باشد، کلمه اصلی بدون تغییر می‌ماند و حروف "way" به انتهای آن اضافه می‌شود.

Impress → impressway

- در نهایت اگر زوج "sh" در کلمه باشد، با آن به عنوان بلاک برخورد می‌شود و به انتهای کلمه منتقل می‌شود.

Shopping → oppingshay

برای ترجمه یک جمله از انگلیسی به این زبان این قواعد را در کنار هم قرار می‌دهیم:

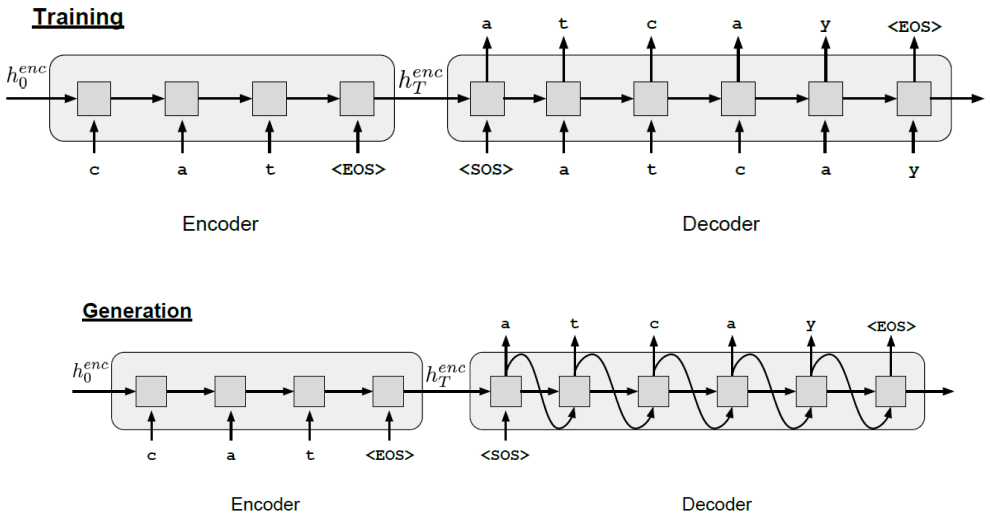
I went shopping → iway entway oppingshay

ما می‌خواهیم یک مدل ترجمه ماشینی عصبی برای یادگیری قواعد latin به طور ضمنی، از جفت کلمات بیاموزیم. از آنجایی که ترجمه به Latin شامل جابجایی کاراکترها در یک رشته است، ما از شبکه های عصبی بازگشتی در سطح کاراکتر برای مدل خود استفاده خواهیم کرد. از آنجایی که انگلیسی و Latin از نظر ساختار بسیار مشابه هستند، کار ترجمه تقریباً یک کار کپی است. مدل باید هر کاراکتر در ورودی را به خاطر بسپارد و کاراکترها را به ترتیب خاصی برای تولید خروجی به یاد بیاورد.

دیتای این تسک شامل زوج کلمات $\{(s^{(i)}, t^{(i)})\}_{i=1}^{N}$ است که منبع $s^{(i)}$ یک کلمه انگلیسی و هدف $t^{(i)}$ ترجمه آن در زبان موردنظر است. دیتاست شامل کلمات یکتا است. واژگان شامل ۲۹ توکن است. که از تعداد ۲۶ حرف استاندارد زبان است که همه کوچک هستند و ۳ توکن دیگر دش(-) و <SOS> و <EOS> است که بیانگر آغاز و پایان رشته است. نمونه های از دیتا را در قسمت زیر مشاهده می‌کنید:

{ (the, ethay), (family, amilyfay), (of, ofway), ... }

کار ترجمه در اینجا با سیستم RNN انجام میشود که از دو جز encoder و decoder تشکیل شده است. Encoder رشته ورودی را در یک بردار با طول ثابت فشرده می‌کند که با ht نمایش داده می‌شود. Decoder روی یک این بردار شرط می‌گذارد تا ترجمه را کاراکتر به کاراکتر تولید کند.



"کار شما"

قسمت اول:

یک گیت GRU طبق روابط زیر تعریف می‌شود:

$$r_t = \sigma(W_{ir}x_t + W_{hr}h_{t-1} + b_r) \quad (1)$$
$$z_t = \sigma(W_{iz}x_t + W_{hz}h_{t-1} + b_z) \quad (2)$$
$$g_t = \tanh(W_{in}x_t + r_t \odot (W_{hn}h_{t-1} + b_g)) \quad (3)$$
$$h_t = (1 - z) \odot g_t + z \odot h_{t-1}, \quad (4)$$

اگرچه pytorch یک پیاده‌سازی برای GRU دارد، ما قصد پیاده‌سازی آن را برای فهم بهتر داریم.

برای این کار سلول GRU را در نوت‌بوک یافته و قسمت __init__ و متد forward آن را طبق تعاریف بالا کامل کنید.

سپس GRU RNN را در قسمت Training - RNN decoder آموزش دهید. به طور پیش فرض این اسکریپت در ۱۰۰ ایپاک ران می‌شود. در انتهای این قسمت خطای آموزش و ولیدیشن نمایش داده میشود و ترجمه برای یک جمله ثابت نمایش داده میشود.

سپس توسط این مدل لغات سلول بعد را با استفاده از تابع translate_sentence ترجمه کنید.

قسمت دوم:

پیاده سازی attention به روش additive با یک MLP دو لایه

برای یکسان سازی رابط بین ماژول های مختلف توجه، ما attention را به عنوان تابعی در نظر می گیریم که ورودی های آن سه گانه هستند (queries, keys, values) که با (Q, K, V) مشخص می شود.

در additive attention، ما تابع f را یادگیری خواهیم کرد که به عنوان یک شبکه دولایه کاملا متصل با تابع فعالسازی relu پارامترسازی شده است.

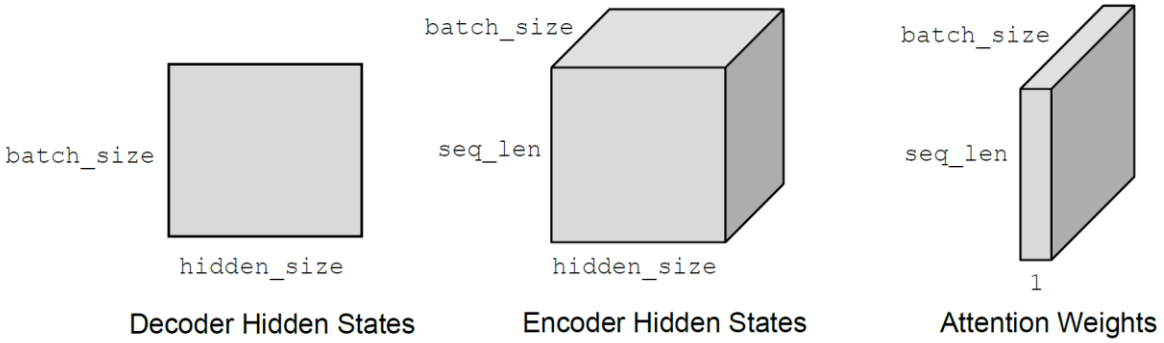
این شبکه وزن های نرمالایز نشده $\tilde{\alpha}_i^{(t)}$ که برای محاسبه context vector نهایی استفاده میشوند را تولید میکند.

$$\tilde{\alpha}_i^{(t)} = f(Q_t, K_i) = W_2(\max(0, W_1[Q_t; K_i] + b_1)) + b_2,$$
$$\alpha_i^{(t)} = \text{softmax}(\tilde{\alpha}^{(t)})_i,$$
$$c_t = \sum_{i=1}^{T} \alpha_i^{(t)} V_i.$$

در اینجا $[Q_t; K_i]$ نشان دهنده کنارهم قرارگیری بردار های Q_t و K_i است. برای اینکه وزن های attention بین ۰ و ۱ قرار گیرد ما تابع softmax را به attention نرمالایز نشده اضافه کردیم.

هنگامی که وزن های attention را داشته باشیم، یک بردار زمینه ct به عنوان ترکیبی خطی از encoder hidden states، با ضرایب داده شده توسط وزن ها، محاسبه می شود.

تابع forward کلاس AdditiveAttention را پر کنید و از softmax برای نرمالایزسازی وزن ها استفاده کنید.



برای forward pass دسته ای از query مرحله زمانی فعلی به شما داده میشود که دارای ابعاد batch_size x hidden_size است و دسته ای از key ها و value های هر مرحله زمانی از دنباله ورودی به شما داده میشود که هر دو دارای ابعاد batch_size x seq_len x hidden_size هستند.

هدف بدست آوردن context vector است.

برای درک مراحل به توضیحات زیر توجه نمایید:

We first compute the function $f(Q_t, K)$ for each query in the batch and all corresponding keys K_i, where i ranges over seq_len different values. You must do this in a vectorized fashion.

Since $f(Q_t, K_i)$ is a scalar, the resulting tensor of attention weights should have dimension batch_size x seq_len x 1.

Some of the important tensor dimensions in the AdditiveAttention module are visualized in above Figure.

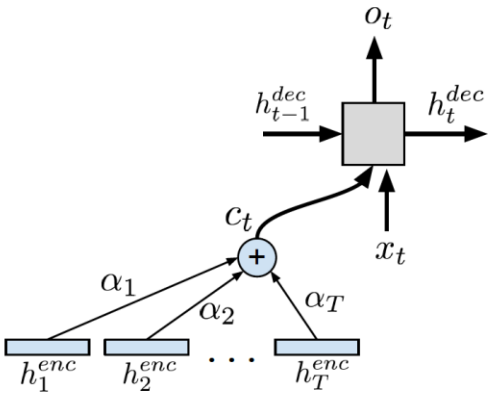
The AdditiveAttention module should return both the context vector batch_size x 1 x hidden_size and the attention weights batch_size x seq_len x 1.

We have provided a template for the forward method of the AdditiveAttention class. You are free to use the template, or code it from scratch, as long as the output is correct.

We will now apply the AdditiveAttention module to the RNN decoder. You are given a batch of decoder hidden states as the query, h_{t-1}^{dec}, for time t - 1, which has dimension batch_size x hidden_size, and a batch of encoder hidden states as the keys and values, $h^{enc} = [h_1^{enc}, \ldots, h_i^{enc}, \ldots]$ for each timestep in the input sequence, which has dimension batch_size x seq_len x hidden_size.

$$Q_t \leftarrow h_{t-1}^{dec}, \quad K \leftarrow h^{enc}, \quad V \leftarrow h^{enc}$$

We will use these as the inputs to the self.attention to obtain the context. The output context vector is concatenated with the input vector and passed into the decoder GRU cell at each time step, as shown in below Figure.



تابع forward در RNNAttentionDecoder را طبق شکل بالا کامل نمایید. برای این کار به موارد زیر نیاز دارید:

- محاسبه context vector و وزن های attention بوسیله self.attention
- Concat کردن context vector با ورودی کنونی decoder
- دادن مقدار بالایی به سلول Decoder GRU برای بدست آوردن hidden state جدید

در مرحله نهایی Attention RNN در قسمت Training - RNN attention decoder را آموزش دهید و نتایج آن را با حالت بدون وجود Attention مقایسه کنید.

موفق باشید.