به نام خدا

۱ شبکه عصبی پیچشی

عملگر کانولشن نقش blur کردن اطلاعات را دارد یعنی می‌تواند یک فیلتری را روی پنجره خاصی اعمال کند و همسایگی‌های متن را خلاصه کند. بدین صورت که پنجره‌ای به طول c داریم که روی داده سر می‌خورد و در فیلتری ضرب می‌شود و خروجی آن را به عنوان خلاصه‌ای از همسایگی‌های متن داریم. خروجی امبدینگ جمله به طول N با پنجره‌ای به طول c به صورت زیر است:

$$h_i = g(W^T x_{i:(i+c-1)} + b)$$

$$S = [h_1, ..., h_{N-c+1}] \in \mathbb{R}^{N-c+1}$$

۲ شبکه عصبی بازگشتی

۱.۲ traditional RNN

شبکه‌های عصبی بازگشتی (RNN) خانواده‌ای از شبکه‌های عصبی هستند که بطور ویژه جهت پردازش داده‌های سری (یا دنباله‌ها) طراحی شده‌اند. زمانی که می‌خواهیم ترتیب یا همان زبان را در مدل‌مان اثر دهیم از آن‌ها استفاده می‌کنیم. این مدل‌ها می‌توانند اطلاعات قبلی را در کنار اطلاعات جدید داشته باشند و بتوانند براساس این اطلاعات خروجی مدنظر را به ما بدهند.

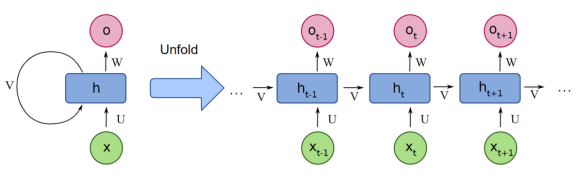
زمانی که می‌خواهیم سیر را خلاصه‌سازی کنیم، یعنی داده زمانی داریم و می‌خواهیم تاریخچه ترتیبی که وجود داشته را نگه داریم، از این شبکه عصبی استفاده می‌کنیم.

از یک لایه ورودی، یک لایه پنهان و یک لایه خروجی تشکیل شده است. لایه پنهان شامل اتصالات مکرر است، به این معنی که خروجی لایه پنهان در زمان t به لایه پنهان در زمان $t+1$ بازگردانده می‌شود. این به شبکه اجازه می‌دهد تا یک حالت داخلی را حفظ کند که می‌تواند اطلاعات مراحل زمانی قبلی را بگیرد. ورودی در هر مرحله زمانی با حالت پنهان قبلی ترکیب می شود تا یک حالت پنهان جدید ایجاد شود، که سپس برای تولید خروجی آن مرحله زمانی استفاده می‌شود. این فرآیند برای هر مرحله زمانی در ترتیب ورودی تکرار می شود.

$$h_t = g(W^{hh} h_{t-1} + x_t^T W_{xh} + b_h)$$

$$y_t = g'(h_t^T W_{hy} + b_y)$$

لازم به ذکر است که back propagation در طول زمان رخ می‌دهد. از آنجا که قاعده زنجیری وابسته به ماتریس W می‌شود در صورتی که مقدار تکین این ماتریس بزرگتر از یک باشد به علت ضرب‌های پی‌درپی ممکن است گرادیانی که محاسبه می‌شود بسیار بزرگ شود و اصطلاحا gradient exploding داریم و در صورتی که مقدار تکین این ماتریس کوچکتر از یک باشد به علت ضرب‌های پی‌درپی ممکن است گرادیانی که محاسبه می‌شود بسیار کوچک شود و اصطلاحا vanishing gradient داریم.



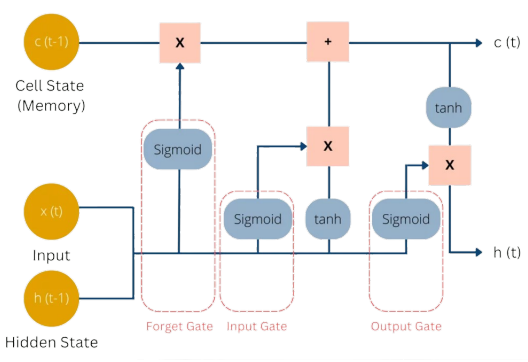
شکل ۱: RNN architecture

۲.۲ LSTM

یک شبکه LSTM یا Long Short Term Memory یک نوع شبکه عصبی بازگشتی است که به خاطر حافظه کوتاه مدت و بلند مدت خود، قادر به حفظ اطلاعات قبلی و استفاده از آن‌ها در آینده است. LSTM ها در بسیاری از مسائل پردازش زبان طبیعی مانند تشخیص احساسات و ترجمه ماشینی استفاده می‌شوند.

از input gate ، output gate ، forget gate ، cell state تشکیل شده است. cell state مقادیر را در بازه‌های زمانی دلخواه به خاطر می‌آورد و سه گیت جریان اطلاعات را به داخل و خارج از cell state تنظیم می‌کنند. forget gate ها تصمیم می‌گیرند چه اطلاعاتی را از وضعیت قبلی حذف کنند. input gate ها تصمیم می‌گیرند که کدام بخش از اطلاعات جدید را در وضعیت فعلی ذخیره کنند (با استفاده از سیستم مشابه forget gate ها). output gate ها با در نظر گرفتن حالت‌های قبلی و فعلی، کنترل می‌کنند که کدام بخش از اطلاعات در وضعیت فعلی خروجی داده شود.

مزیت اصلی LSTM دو جهته (BiLSTM) نسبت به vanilla LSTM توانایی آن در استفاده از اطلاعات از past context و future context برای یادگیری بازنمایی بهتر تک کلمات است. به عبارت دیگر، BiLSTM یک جمله را به جلو و عقب می‌خواند تا اطلاعات بیشتری به دست آورد.



شکل ۲: LSTM architecture

۳ امبدینگ‌های سیاق‌محور

سه تفاوت concat pretrained LM و skipgram این است که در concat pretrained LM جملات را طولانی‌تر لحاظ می‌کنیم و بهتر است. همچنین skipgram همزمان به چپ و راست کلمه نگاه می‌کند اما در concat pretrained LM به صورت جداگانه به چپ و راست می‌پردازیم که بهتر است. تفاوت دیگر این است که در skipgram برای کلمه یکسان در جملات متفاوت امبدینگ یکسان داریم اما در concat pretrained LM اینطور نیست زیرا سیاق‌محور هستند.

۱.۳ TagLM

یک رویکرد نیمه نظارت شده است که با concat کردن امبدینگ‌های از پیش آموزش‌دیده از مدل‌های زبان دوطرفه برای هر توکن در دنباله ورودی در تسک sequence tagging استفاده می‌شود.

۲.۳ ELMO

یک deep contextualized word representations است که بر اساس یک مدل زبان دو جهته عمیق (biLM) است. biLM بر روی یک مجموعه متن بزرگ از قبل آموزش داده شده است و از دو لایه تشکیل شده است که هر یک دارای یک forward pass و backward pass است. (معمولا از دو LSTM استفاده می‌شود) نمایش هر توکن مانند رابطه زیر به صورت جمع وزن‌دار بردار اولیه با بردار مرحله یک و دو به دست می‌آید. در تسک‌های مربوط به syntax وزن لایه پایین بیشتر می‌شود و در تسک‌های مربوط به semantic وزن لایه بالا بیشتر می‌شود.

$$ELMO_k^{task} = \gamma^t ask \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

شکل ۳: ELMO architecture

۳.۳ ULMFit

شامل یک معماری سه لایه است. شامل سه مرحله است: مرحله اول بدین صورت است که از یک مدل زبان از پیش آموزش دیده شده استفاده می‌کنیم. در مرحله دوم این مدل زبانی را برای دامنه مد نظرمان fine tune می‌کنیم. در مرحله سوم با classifier را برای دامنه مد نظرمان fine tune می‌کنیم. در هنگام fine tune کردن learning rate لایه‌های پایین کمتر است زیرا کمتر آپدیت می‌شوند. همچنین اعمال learning rate بر حسب زمان باید متفاوت باشد در ابتدا به صورت خطی افزایش می‌یابد تا به نقطه اوجی برسد و سپس به صورت خطی کاهش می‌یابد زیرا در ابتدا می‌خواهیم به سرعت به یک منطقه مناسب از فضای پارامتر همگرا شود و سپس با کاهش نرخ یادگیری، پارامترهای آن را اصلاح کند.

۴ امبدینگ‌هاي سیاق‌محور بدون RNN

در این بخش با این خط فکری شروع می‌کنیم که می‌خواهیم بدون RNN برای هر کلمه که امبدینگ آن موجود است، امبدینگ سیاق‌محور آن را در محاسبه کنیم. برای این کار یک پنجره در نظر می‌گیریم و امبدینگ‌های کلمات اطراف آن کلمه و حاضر در پنجره را در خود آن لحاظ می‌کنیم. مشکلات این ایده این است که ممکن است اطراف آن فقط stopword باشد و کلمات بی‌ربط در اطراف آن باشد که راهکار آن این است که به کلمات وزن بدهیم یعنی برای امبدینگ هر کلمه، میانگین وزن‌دار تمام کلمات را در نظر بگیریم که این وزن عملا میزان ارتباط کلمه مورد نظر با کلمه دیگر می‌شود. این ایده مقدمه‌ای بر attention است.