به نام خدا

یادگیری عمیق، تکلیف چهارم
مهدی کافی ۹۹۲۱۰۷۵۳

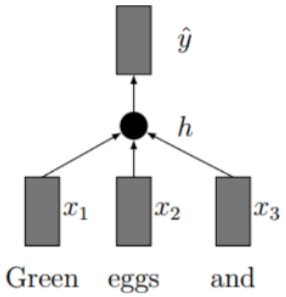
مسئله‌ی ۱. Neural network Language Models(NNLM)

برای آموزش بردارهای کلمات [۱] می‌توانیم از معماری مدل‌های زبانی شبکه عصبی [۲] استفاده کنیم. این مدل سعی می‌کند با توجه به N کلمه قبلی(کلمات زمینه [۳]) ، کلمه بعد از آن‌ها را(کلمه هدف [۴]) پیش‌بینی کند. برای این‌کار بردارهای کلمات قبلی را باهم متصل [۵] می‌کنیم و حاصل را از یک لایه مخفی با تابع فعال‌ساز Tanh رد می‌کنیم و در آخر خروجی را به یک لایه Softmax می‌دهیم تا کلمه هدف را پیش‌بینی کند. با در نظر گرفتن فرض های زیر، به سوالات پاسخ دهید.

- سایز لایه مخفی برابر با H و تعداد لغات منحصر به فرد [۶] داخل پایگاه داده V است.
- تابع هزینه مورد استفاده برای آموزش مدل، Cross Entropy است.
- بردار کلمه‌ی N لغات زمینه برابر با ماتریس x است که هر کدام از x_i ها یک بردار D بعدی هستند.
- y بردار Onehot کلمه هدف است.
- ابعاد پارامترها و متغیرها برابر است با:

$$x \in \mathbb{R}^{(N \cdot D)}, W \in \mathbb{R}^{H \times (N \cdot D)}, b \in \mathbb{R}^{H}, h \in \mathbb{R}^{V \times H}, d \in \mathbb{R}^{V}, \hat{y} \in \mathbb{R}^{V}$$

- روابط ریاضی در شبکه مورد نظر:



$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

$$\mathbf{h} = \tanh(W\mathbf{x} + \mathbf{b})$$

$$\hat{\mathbf{y}} = \text{softmax}(U\mathbf{h} + \mathbf{d})$$

$$J = \text{CE}(\mathbf{y}, \hat{\mathbf{y}})$$

$$CE = -\sum_i y_i \log(\hat{y}_i).$$

(آ) دو تفاوت اصلی و عمده بین NNLM و CBOW که در درس آموخته‌اید را بیان کنید و برای هرکدام یک مثال (به زبان فارسی یا انگلیسی) بزنید. (۹ نمره)

(ب) پیچیدگی محاسباتی را در بدست آوردن خروجی شبکه NNLM با روش انتشار رو به جلو برای یک عدد داده آموزشی محاسبه کنید. توجه کنید که برای گرفتن نمره کامل این بخش باید مرحله به مرحله پیچیدگی محاسباتی را حساب کنید و سپس تجمیع آن‌ها را بدست آورید. (۷ نمره)

(ج) اگر بتوانیم تابع هزینه را تغییر دهیم، یک روش ارائه کنید تا بتوانیم پیچیدگی محاسباتی بخش قبل را کاهش دهیم. نام بردن روش به صورت مختصر کافی است و نیازی به توضیح نیست. (۲ نمره)

(د) گرادیان J نسبت به x را بدست آورید. (۵ نمره)

آ. مدل‌های زبانی شبکه عصبی (NNLM)، به طور کلی سعی می‌کنند احتمال آمدن کلمه w_t به شرط داشتن $n-1$ کلمه قبلی یا عبارت زیر را تخمین بزنند.

$$p(w_t|w_{t-1},\dots,w_{t-n+1})$$

اولین لایه در این شبکه‌ها معمولا نقش بردار کلمات را بازی می‌کند و می‌توان از وزن‌های این لایه به عنوان بردار کلمات استفاده کرد.

روش CBOW یکی از روش‌های word embedding است که سعی می‌کند، روابط معنایی بین کلمات را آموزش ببیند و همینطور کلمات را به فضای با بعد کمتر ببرد، به نحوی که پنجره‌ای به طول مشخص از کلمات اطراف کلمه هدف را به شبکه می‌دهد و شبکه سعی می‌کند که کلمه هدف را تخمین بزند.

تفاوت اصلی این دو روش، که یکی، بردار کلمات یک محصول فرعی آن است و دیگری هدف اصلی‌اش ایجاد کردن بردار کلمات است، در پیچیدگی محاسباتی آنها است. ساخت بردار کلمات با یک شبکه عمیق برای یک مجموعه لغات بزرگ، توسط NNLM پیچیدگی محاسباتی بسیار زیادی دارد.
تفاوت دیگر، در هدف اصلی آموزش این شبکه‌ها است؛ روش CBOW هدف اصلی‌اش، پیدا کردن روابط معنایی کلی بین لغات است که می‌تواند در بسیاری از فرآیندهای بعدی مفید و سودمند باشد. در مقابل، روش NNLM بردار لغاتی مختص همان هدف و داده‌ای که برای آنها آموزش دیده است، ایجاد می‌کند که برای کاربردهای دیگر مناسب نخواهد بود.

اگر فرض کنیم که جمله She is eating the green apple را داریم. روش NNLM سعی می‌کند به طور مثال با دیدن ۲ کلمه قبلی کلمه بعدی را تخمین بزند. مثلا در ابتدا با داشتن دو کلمه (She, is) سعی در تخمین کلمه eating دارد، سپس با کلمات (is,) eating) سعی می‌کند کلمه the را تخمین بزند و این کار را تا انتهای جمله ادامه می‌دهد. روش CBOW به اینصورت عمل می‌کند که به طور مثال اگر پنجره‌ای به طول یک کلمه قبل و یک کلمه بعد داشته باشد در ابتدا با داشتن کلمه‌های (She, eating) سعی در تخمین کلمه is دارد و سپس با کلمات (is, the) به عنوان کلمات زمینه سعی می‌کند کلمه eating را به عنوان کلمه هدف تخمین بزند و این روش نیز همین کار را تا انتهای جمله ادامه می‌دهد و بردار کلمات رو تولید می‌کند.

ب. برای یک داده آموزشی نیاز به N کلمه زمینه داریم و هر یک از این کلمات را باید از فضای embedding استخراج کنیم و هر کدام از آنها D بعدی هستند و در نهایت این N بردار D بعدی را با یکدیگر concatenate می‌کنیم و یک بردار N.D بعدی می‌سازیم. این مرحله پیچیدگی محاسباتی $N \times D$ خواهد داشت. سپس این بردار را در بردار وزن‌های لایه مخفی ضرب می‌کنیم که ضرب یک بردار N.D بعدی در یک ماتریس H در N.D بعدی است و این ضرب پیچیدگی محاسباتی $N \times D \times H$ خواهد داشت. در نهایت یک بردار H بعدی داریم که نیاز داریم در یک ماتریس H در V بعدی ضربش کنیم برای محاسبه مقادیر لایه

خروجی قبل از اعمال تابع softmax و این مرحله نیز پیچیدگی محاسباتی $H \times V$ خواهد داشت. در نتیجه پیچیدگی محاسباتی کل برابر با مجموع پیچیدگی‌های هر مرحله و برابر با عبارت زیر است.

$$Computational\ Complexity = N \times D + N \times D \times H + H \times V$$

ج. به هنگام محاسبه تابع softmax نیاز داریم که تمام لغات موجود در vocabulary را پردازش کنیم و این محاسبات بسیار زمان گیر است. بنابراین از روش‌های hierarchical softmax و یا negative sampling استفاده می‌کنیم.

د. برای محاسبه گرادیان نیاز داریم که در ابتدا گرادیان تابع هزینه یعنی Cross-Entropy را نسبت به خروجی شبکه یعنی $\hat{y}$ محاسبه کنیم و سپس با توجه به مشتق‌های زنجیره‌ای، این گرادیان را در گرادیان خروجی شبکه $(\hat{y})$ نسبت به خروجی لایه آخر قبل از تابع softmax، ضرب کنیم. در کلاس درس اثبات شده‌است که حاصل ضرب این دو گرادیان برابر با عبارت زیر است.

$$\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o} = \hat{y} - y$$

سپس باید گرادیان o که خروجی لایه آخر قبل از تابع softmax است را نسبت به ورودی آن محاسبه کنیم. این مقدار به صورت زیر خواهد بود.

$$\frac{\partial o}{\partial h} = U^T$$

سپس باید گرادیان h را نسبت به z که خروجی لایه مخفی قبل از تابع tanh است را محاسبه کنیم.

$$\frac{\partial h}{\partial z} = 1 - tanh(z)^2$$

و در آخر گراداین z را نسبت به x محاسبه میکنیم.

$$\frac{\partial z}{\partial x} = W^T$$

با توجه به مشتق‌های زنجیره‌ای، برای محاسبه گرادیان تابع هزینه نسبت به x نیاز داریم که تمامی گرادیان‌های محاسبه شده را در یکدیگر ضرب کنیم.

$$\frac{\partial L}{\partial x} = (\hat{y} - y) \times U^T \times (1 - tanh(Wx + b)^2) \times W^T$$

مسئله‌ی ۲. Word2Vec and Glove

به سوالات زیر پاسخ دهید.

(آ) یکی دیگر از روش‌های یادگیری بردار کلمات، روش‌های وقوع همزمان مبتنی بر شمارش[۸] است. دو مزیت و دو عیب در استفاده از روش Word2Vec نسبت به این روش را بیان کنید. (۴ نمره)

(ب) دو نفر می‌خواهند از روش Word2Vec برای بدست آوردن تعبیه کلمات در یک Vocabulary یکسان با سایز V استفاده کنند. به طور دقیق‌تر نفر اول بردار کلمات زمینه u_w^A و بردار کلمه هدف v_w^A را برای هر $w \in V$ و نفر دوم بردار کلمات زمینه u_w^B و بردار کلمه هدف v_w^B را برای هر $w \in V$ می‌خواهند به وسیله این روش، بدست آورند. فرض کنید برای هر جفت کلمه $w, w' \in V$ ضرب داخلی بردار کلمات در مدل هر دو نفر یکسان باشد یعنی

$$(u_w^A)^T v_{w'}{}^A = (u_w^B)^T v_{w'}{}^B$$

آیا می‌توان نتیجه گرفت که برای هر کلمه $w \in V$ داریم $v_w^A = v_w^B$ ؟ چرا؟ (۴ نمره)

(ج) چرا استفاده از تابع Softmax برای Word2Vec کار درستی نیست؟ راهکار پیشنهادی شما برای حل این مشکل چیست؟ (۳ نمره)

(د) مقدار حافظه مصرفی الگوریتم های Word2Vec و Glove را با ذکر دلیل مقایسه کنید. (۲ نمره)

(ه) دو مورد از نواقص مشترک Word2Vec و Glove را نام ببرید. (۴ نمره)

(و) در تابع هزینه زیر که متعلق به الگوریتم Glove است، اگر ماتریس های R و $\tilde{R}$ تعبیه‌های کلمات $\{r_i\}, \{\tilde{r}_j\}$ را در خود داشته باشند، نشان دهید که این تابع هزینه محدب[۹] نیست. برای حل این سوال، نیازی به اثبات ریاضی نیست و فقط کافی است توضیح منطقی‌ای ارائه کنید که محدب نبودن را تصدیق کند.

$$J(R, \tilde{R}) = \sum_{i,j} f(x_{ij})(r_i^T \tilde{r}_j - \log x_{ij})^۲$$

راهنمایی: از مفاهیم Swap-inveriance و Permutation-invariance استفاده کنید. (۷ نمره)

آ. مزیت‌های روش Word2Vec

۱. روش Word2Vec به حافظه بسیار کمتری نسبت به روش‌های وقوع همزمان مبتنی بر شمارش نیاز دارد و فضای تعبیه آن کوچک و ثابت است بر خلاف روش‌های وقوع همزمان مبتنی بر شمارش که فضای تعبیه متناسب با سایز کلمات است.

۲. می‌تواند روابط معنایی و دستور زبانی بین کلمات را پیدا کند و به طور مثال می‌تواند با کم و زیاد کردن بردارهای تعبیه لغات به بردارهای تعبیه‌ای برسد که در معنا هم این بردارها درست هستند به طور مثال نتیجه $king - man + woman$ برداری است که بسیار به بردار $queen$ نزدیک است.

معایب روش Word2Vec

۱. برای ساخت بردارهای تعبیه تنها به اطلاعات محلی توجه می‌کند. بازنمایی معنایی یک لغت تنها به کلمات همسایه آن بستگی دارد که می‌تواند غیر بهینه باشد.

۲. برای آموزش بردارهای تعبیه به زمان بیشتر و corpus بزرگتری نیاز دارد زیرا که باید بردارهای تعبیه با فرایند آموزش شبکه ایجاد شوند.

ب. برای رد کردن این نتیجه گیری یک مثال نقض می‌آوریم. به طور مثال بردارهای دو بعدی زیر بردارهای تعبیه کلمات car و cat برای دو فرد A و B هستند.

A	V (target)	U (context)
car	(3, 1)	(5, 2)
cat	(1, 2)	(2, 3)

B	V (target)	U (context)
car	(1, 4)	(2, 1)
cat	(4, 1)	(1, 2)

حال اگر w کلمه car و w' کلمه cat باشند. خواهیم داشت.

$$[5, 2].[1, 2] = [2, 1].[4, 1]$$

عبارت بالا درست است و حال اگر w کلمه cat و w' کلمه car باشد خواهیم داشت.

$$[2, 3].[3, 1] = [1, 2].[1, 4]$$

این عبارت نیز برقرار است. حال با توجه به نتیجه گیری صورت سوال باید مقادیر $[3, 1]$ و $[1, 4]$ با یکدیگر و مقادیر $[1, 2]$ و $[4, 1]$ نیز با یکدیگر برابر باشند که اینطور نیست.

ج. تابع softmax در لایه آخر نیاز دارد که روی تمام لغات vocabulary عملات softmax را اجرا کند. بنابراین برای محاسبه مخرج کسر softmax نیاز دارد که exp تمامی مقادیر لایه آخر که به تعداد کلمات vocabulary هستند را با یکدیگر جمع کند که این کار بسیار زمانبر است. روش‌های پیشنهادی برای جلوگیری از این زمان بسیار زیاد، روش‌های hierarchical softmax و negative sampling هستند که در زمان کمتر سعی در تخمین مقادیر softmax دارند.

د. روش GloVe یک روش hybrid و ترکیبی از روش‌های co-occurrence و Word2Vec است و سعی می‌کند مزایای هر دو روش را ترکیب کند. این روش علاوه بر وزن‌های شبکه به ماتریس وقوع همزمان نیاز دارد و به همین دلیل حافظه مصرفی آن از حافظه مصرفی Word2Vec که روشی مبتنی بر تخمین است و فقط به حافظه‌ای برای وزن‌های شبکه‌اش نیاز دارد، بیشتر است.

ه. دو مورد از نواقص مشترک روش‌های Word2Vec و GloVe به صورت زیر هستند.

۱. هر دو این روش‌ها نمی‌توانند برای کلمات با چند معنی بردار تعبیه خوبی تولید کنند زیرا که کلمات و بردار تعبیه آنها یک به یک هستند.

۲. هر دو این روش‌ها نمی‌توانند برای کلمات خارج از corpusای که از روی آن بردارهای تعبیه را تولید کرده‌اند، بردار تعبیه خوبی ارائه دهند.

و. در مراحلی که این تابع هزینه ایجاد می‌شود، تابع Fای وجود دارد که این تابع خاصیت symmetry دارد و اگر ورودی‌های آنرا جا به جا کنیم باز هم خروجی آن یکسان می‌شود. فرض می‌کنیم که این تابع F تابع expاست و سپس با لاگ گرفتن از دو طرف

رابطه، لگاریتم در عبارت ظاهر می‌شود. حال اگر r_i و $\widetilde{r}_j$ را جابجا کنیم باز هم مقدار تابع هزینه یکسان می‌شود. نشان دادیم که به ازای مقادیر متفاوت ورودی تابع هزینه مقدار آن یکسان می‌شود و باعث می‌شودکه این تابع محدب نباشد.

مسئله‌ی ۳. Transformers and Attention models

به سوالات زیر پاسخ دهید.

۱. یکی از مراحل مقدماتی در پیش پردازش داده‌ها برای حوزه پردازش متن، توکن‌سازی [۱۰] از جمله‌های پایگاه داده است. یکی از چالش‌های این مرحله، این است که ترتیب کلمات در جمله با این‌کار از بین می‌رود. برای حل این مشکل رویکرد Positional Embedding مطرح می‌شود. به صورت کلی و کوتاه توضیح دهید که چگونه این رویکرد می‌تواند مشکل را برطرف کند. (۳ نمره)

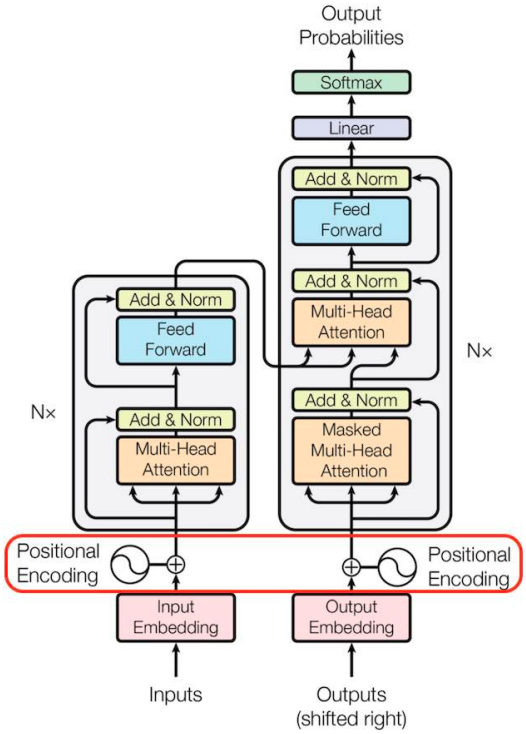
۲. یکی از کاربردهای رویکرد گفته شده در قسمت قبل، استفاده از Transformer ها است. این معماری چگونه مشکل گفته شده(بهم خوردن ترتیب کلمات) را برطرف می‌کند؟ (۴ نمره)

۳. محدودیت معماری Encoder-Decoder ای که از مکانیزم توجه استفاده نمی‌کند را در حوزه ترجمه ماشینی بیان کنید و سپس به صورت خلاصه توضیح دهید که چگونه مکانیزم توجه می‌تواند این مشکل را برطرف کند. (۶ نمره)

۱. در transformerها داده‌های ورودی به صورت موازی پردازش می‌شوند و به همین دلیل اطلاعات مربوط به موقعیت داده‌ها از بین می‌رود. راه حلی که برای این مسئله مطرح شده‌است positional embedding نام دارد. این روش به این صورت عمل می‌کند که برای هر داده علاوه بر بردار معنایی آن یک بردار مشخص کننده موقعیت نیز تعریف می‌شود که می‌توان این بردار را با بردار اصلی جمع و یا به آن concat کرد. هدف این بردار نیز این است که اطلاعات مکانی، در بردارهای ورودی self-attention در transformerها تعبیه شوند و سعی کنند که با وجود پردازش موازی داده‌های ترتیبی، اطلاعات مکانی آنها حفظ شوند. این بردارها باید چند ویژگی داشته‌باشند، یکی اینکه فارغ از تعداد داده‌ها و محتوای آنها مقادیر یکسانی برای هر جایگاه تولید کنند و دیگری اینکه مقادیر این بردار محدود باشند و با بزرگ شدن بیش از حد، باعث نشوند که به دلیل جمع کردن این بردار جایگاه با بردار معنایی، اطلاعات معنایی از بین بروند. روشی که می‌توان از آن استفاده کرد این است که هر یک از المان‌های بردار مکانی را با توابع sin و cos با فرکانس‌های متفاوت تولید کنیم که باعث می‌شوند هم مقادیر این بردار مکانی بین یک و منفی یک باقی بمانند و هم اینکه این توابع برای مقادیر بسیار بزرگ دامنه نیز variationهای خوبی دارند و فرکانس‌های متفاوت نیز باعث می‌شوند که فرکانس‌های پایینتر می‌توانند مشخص کنند که داده در ابتدا یا انتهای رشته قرار دارند یا خیر و هر چه فرکانس بیشتر می‌شود به طور دقیقتری مکان داده در رشته مشخص می‌شود. روش‌های دیگری که می‌توان استفاده کرد، یادگیری بردار مکان در حین آموزش شبکه است. بنابراین به طور کلی هدف positinoal encoding این است که بردارهایی تولید کند که در کنار بردارهای معنایی، کمک کند اطلاعات موقعیت از بین نروند.

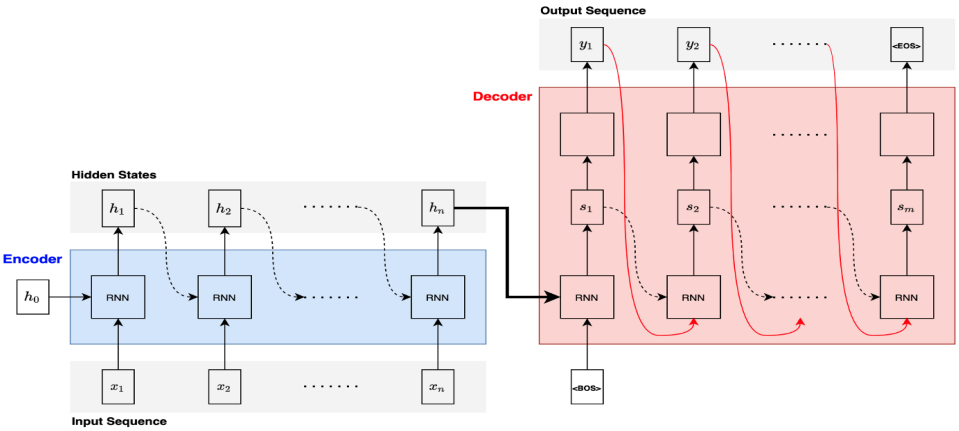
---

۲. در transformerها کلمات به صورت موازی پردازش می‌شوند بنابراین سرعت بسیار بیشتری نسبت به شبکه‌های RNN شبیه به LSTM دارند اما این پردازش موازی باعث می‌شود که اطلاعات مکانی از بین بروند و این تغییر در مکان کلمات می‌تواند معنای جلمه را به طور کلی تغییر دهد. برای حل این مشکل سعی می‌کنیم با کمک positional embedding این اطلاعات مکانی از بین رفته را به بردارهای تعبیه شده از کلمات بازگردانیم و سپس بردارها را مانند شکل زیر به لایه self-attention وارد کنیم.

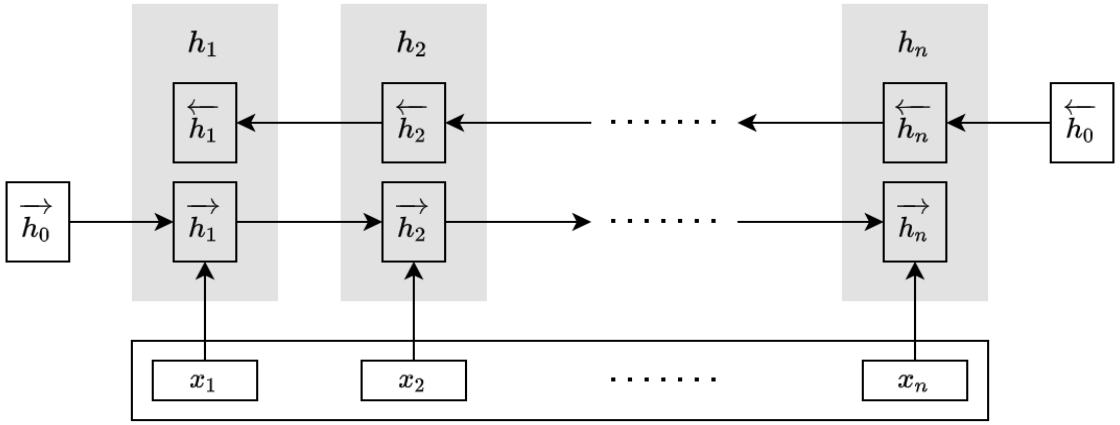


---

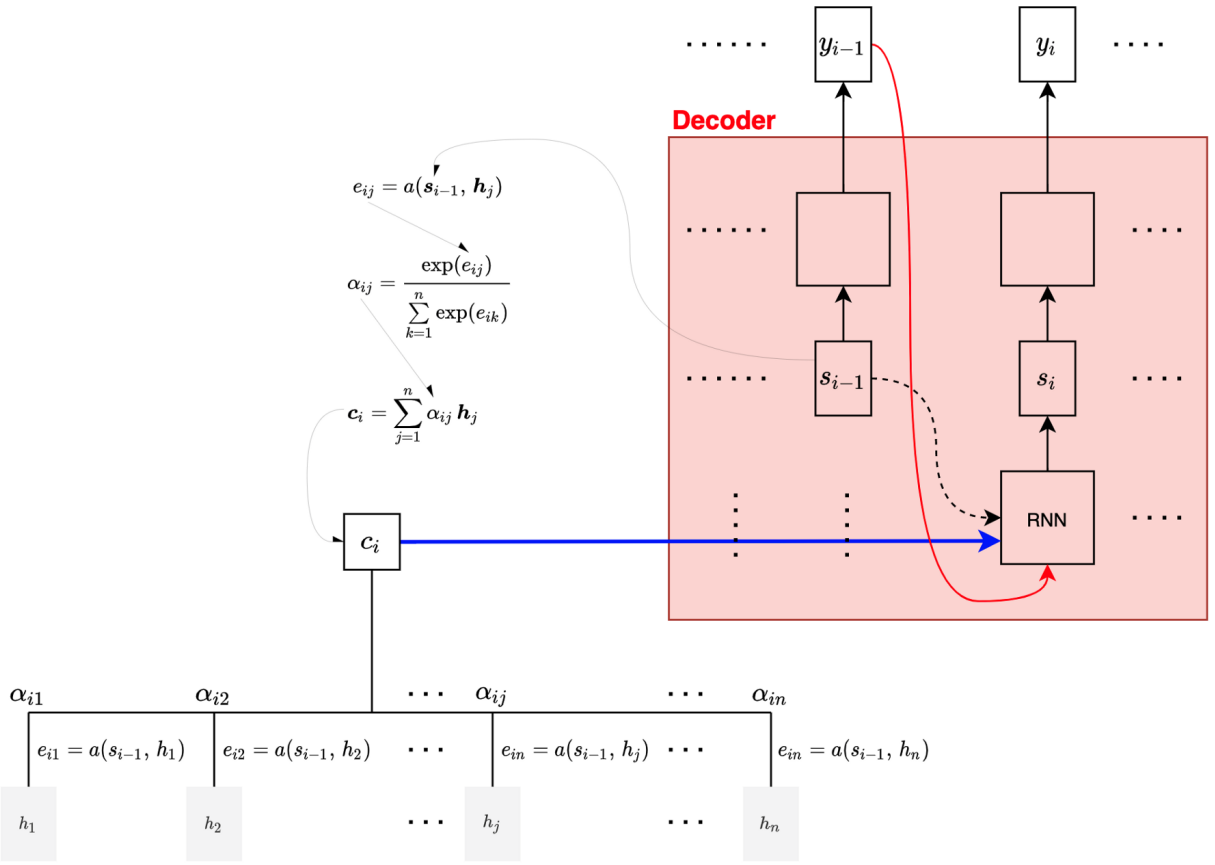
۳. شبکه‌های encoder-decoder که برای ترجمه ماشینی استفاده می‌شوند به این صورت عمل می‌کنند که بخش encoder سعی می‌کند که فارغ از طول جمله ورودی، تمام اطلاعات این جمله را در یک بردار با طول ثابت و از پیش تعیین شده، فشرده کند و بخش decoder نیز با گرفتن این بردار به عنوان ورودی سعی در تولید جمله خروجی دارد. اما این روش برای جملات با طول زیاد، مناسب نیست و باعث می‌شود که پدیده‌ای با عنوان فراموشی ایجاد شود و به طور مثال اگر کلمات انتهایی جمله مقصد به کاملات ابتدایی جمله مبدا مرتبط باشند، این ارتباط به دلیل فاصله زیاد این کلمات (به دلیل طول بلند جمله) و ثابت بودن طول بردار context، فراموش شود. معماری شبکه‌های encoder-decoder به طور کلی به صورت زیر است.



مکانیزم توجه به بخش decoder این اجازه را می‌دهد که به هنگام تولید رشته خروجی، به جای دسترسی داشتن فقط به یک بردار نهایی خروجی از encoder به تمام hidden-stateهای تمام مراحل زمانی encoder دسترسی داشته‌باشد. این رویکرد از پدیده فراموشی جلوگیری می‌کند. در بخش encoder نیز از شبکه‌های RNN دو جهته مانند شکل زیر استفاده می‌شود.



این شبکه‌ها باعث می‌شوند که بخش encoder، بردارهای context غنی‌تری تولید کند که این بردارها concatenationای از بردارهای hidden-state رفت و برگشت در هر مرحله زمانی هستند.
بخش decoder نیز به هنگام تولید رشته خروجی در هر مرحله زمانی، به تمام این بردارهای annotation (کانکت بردارهای hidden-state رفت و برگشت encoder) دسترسی دارد و می‌تواند با جستجو بر روی این بردارها، کلمه یا کلمات مناسب برای تولید کلمه خروجی در آن مرحله زمانی را پیدا کرده و در نهایت خروجی بهتری نسبت به معماری‌های encoder-decoder بدون مکانیزم توجه تولید کند.



در نمودار زیر عملکرد معماری‌های encoder-decoder با مکانیزم توجه و بدون مکانیزم توجه آورده شده‌است. مدلهای RNNsearch-50 و RNNsearch-30 مجهز به مکانیزم توجه و مدل‌های RNNenc-50 و RNNenc-30 بدون مکانیزم توجه هستند. اعداد ۵۰ و ۳۰ نیز به معنای این هستند که در هنگام آموزش شبکه طول جملات ورودی حداکثر ۵۰ و یا ۳۰ کلمه

بوده‌اند. همانطور که در نمودار دیده می‌شود مدل RNNsearch-50 عملکرد بهتری نسبت به سایر معماری‌ها به خصوص برای جملات با طول بیشتر از ۲۰ کلمه دارد.