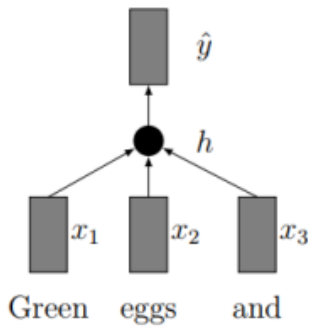
مسئله‌ی ۱. Neural network Language Models(NNLM)

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:



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

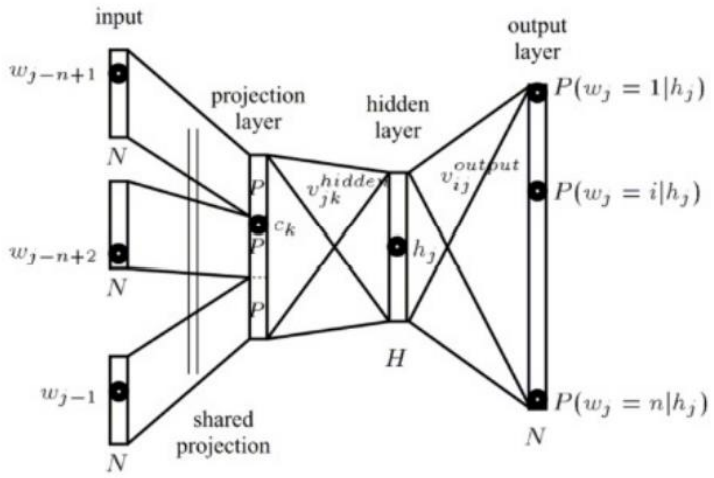
$$\mathbf{h} = \tanh(W\mathbf{x} + \mathbf{b})$$

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

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

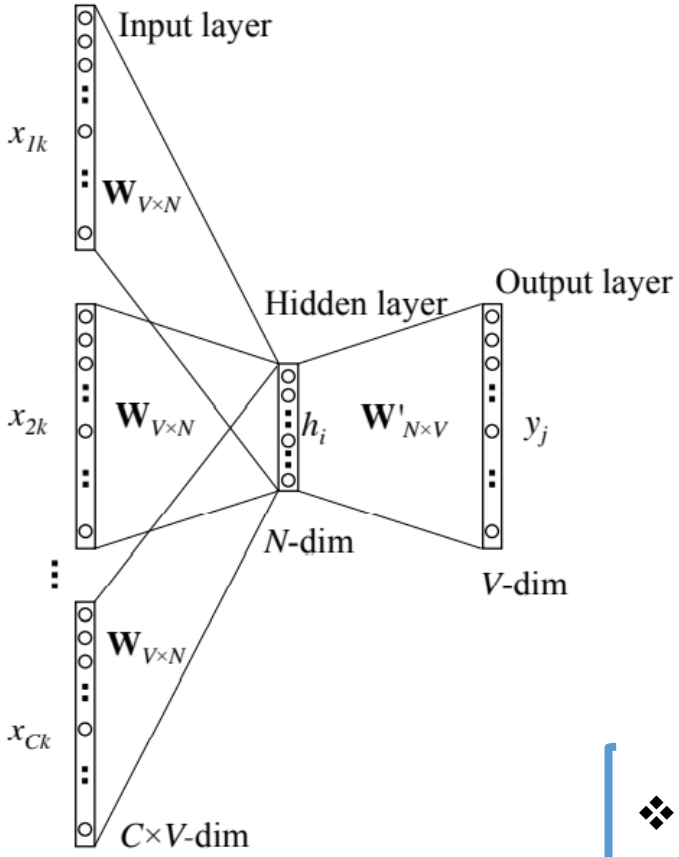
$$CE = -\sum_i y_i \log(\hat{y}_i).$$

(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)



$$p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

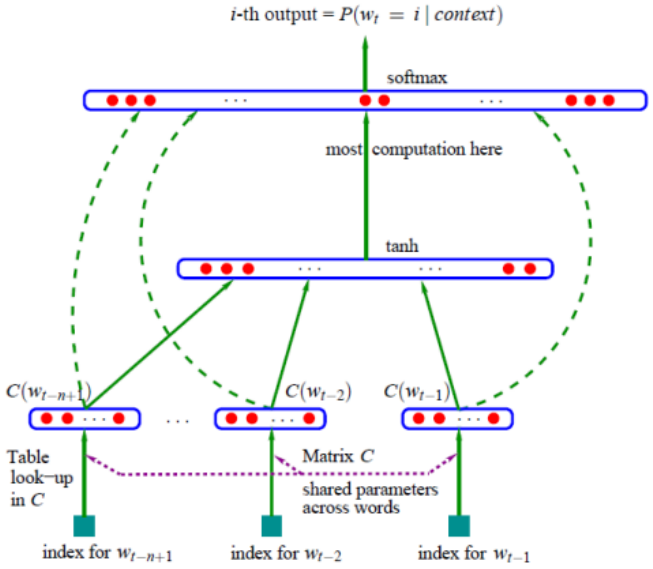
ایده ی NNLM بر این اساس است که ما بر اساس n تا کلمه ای که قبل از یک کلمه آمده است کلمه ی n+1 را پیش بینی می کنیم. بر این اساس که ما تمامی n کلمه ی قبلی را به صورت one-hot به شبکه می دهیم و پس از یک لایه projection داده ها را به یک لایه ی مخفی می دهیم و سپس احتمال تمامی کلمات دیکشنری به شرطی که این n تا کلمه ی قبل از آن آمده باشند را محاسبه می کنیم بنابراین خروجی این شبکه یک دیکشنری است که برای هر کلمه یک احتمال در نظرگفته می شود و بزرگ ترین احتمال را به عنوان کلمه ی بعدی (بعد از n کلمه) معرفی می کند. و تمامی این مرحل مجددا برای کلمات بعدی انجام می شود.



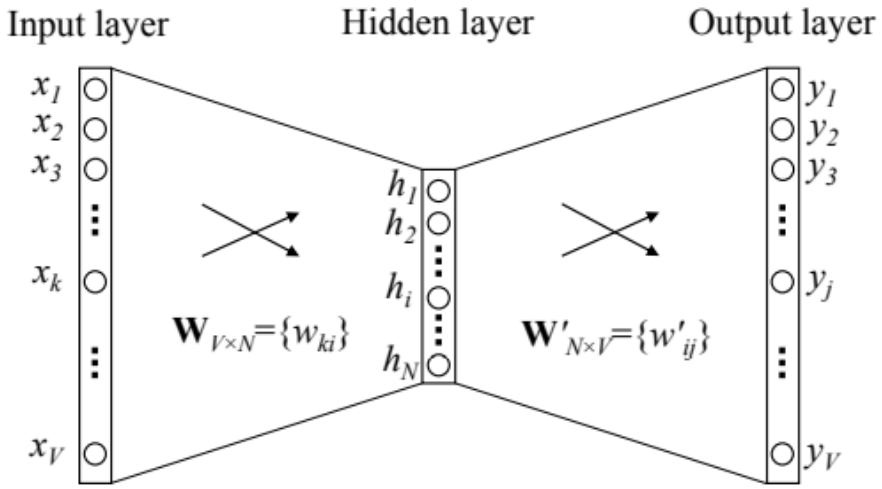
روش CBOW بر این اساس که ما یک پنجره در نظر می گیریم که در هر بار تعداد از کلمات یک جمله درون این پنجره قرار می گیرد و کلمه (کلمه ی وسطی معمولا) به عنوان target در نظر گرفته می شود. بنابراین ما برای پیش بینی کلمه ی target علاوه بر کلمات قبلی کلمات target کلمات بعدی را نیز در نظر می گیریم (این در حالی است که NNLM فقط کلمات قبلی را در نظر می گرفت.)معماری این شبکه بر کاملا بر این اساس است که ما می توانیم شبیه اتوانکدرها یک بردار latent همانطور که در شکل مقابل می بینیم استخراج کنیم که این بردار بازنمایی ان جمله (word2vec) است. معماری کلی CBOW بر این اساس که ما یک لایه مخفی داریم و دو مجموعه وزن w و w' که یکی ورودی لایه ی مخفی را تولید می کند و دیگری کلمه‌ی traget. این معماری بسیاری بهینه تر از روش NNLM است و پیچیدگی های (complexity) موجود در NNLM را بر طرف کرد. یکی از ویژکی های موجود در word2vec این است که برای عملیات ضرب ماتریسی عملا به دلیل اینکه بردارهای ورودی onehot هستند از ماتریس w فقط نیاز داریم یک سطر را ضرب کنیم و برای ماتریس w' صرفا یک ستون.

- Ali is a good boy
- Ali is a bad boy

مثلا اگر به مثال روبه رو توجه کنیم در روش NNLM وقتی جمله ی Ali is a وارد شبکه شود شبکه فرقی بین کلمه ی good و bad در نظر نمی گیرد این در حالی است که روش CBOW به دلیل اینکه در طرفین یک کلمه دیده می شود (یک پنجره در نظر گرفته می شود) می تواند بهتر تواند این دو کلمه Good و Bad را تشخیص دهد.



یکی دیگر از تفاوت هایی که بین دو روش NNLM و CBOW وجود دارد این است که در روش NNLM بردارهای Embedding باهم Concatenate می شوند ولی این در حالی است که در روش CBOW برداهای Embedding باهم جمع می‌شوند و این مسئله سبب می شود وقتی که ما که ترتیب بین کلمات حفظ نشود و این مسئله می تواند سبب شود که جملات با کلمات مشابه ولی با معنای متفاوت یکی در نظر گرفته شوند.

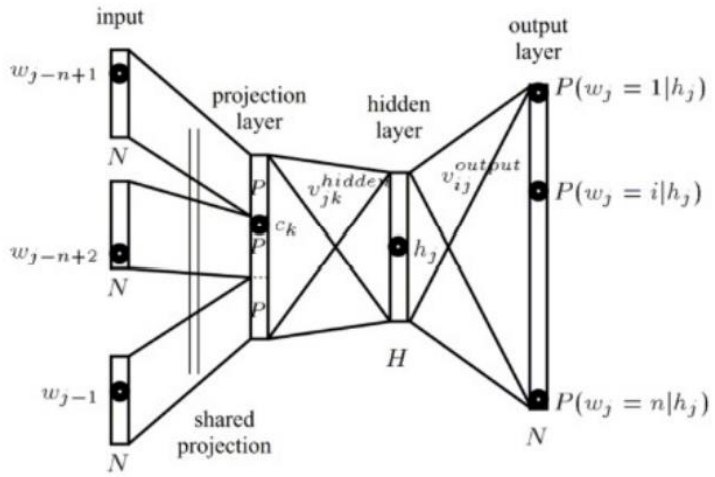


همانطور که دیده می شود در این حالت در معنای دو جمله متفاوت است اما CBOW این دو جمله را یکشان در نظر می گیرد.

- علی پدر حسین است
- پدر حسین علی است

(ب) پیچیدگی محاسباتی را در بدست آوردن خروجی شبکه NNLM با روش انتشار رو به جلو [۷] 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)

همانطور که در بخش قبلی گفته شد معماری NNLM به شکل مقابل است. همانطور که می بینیم این معماری از سه بخش اصلی تشکیل شده است: ۱. لایه ی Projection ۲. لایه ی Hidden ۳. لایه ی خروجی



$$p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

- برای محاسبه ی لایه ی projection داریم:

N*P

که در اینجا N تعداد کلمات قبلی است که مورد استفاده قرار می گیرد و P ابعاد لایه ی Projection است.

- همچنین برای محاسبه ی لایه ی Hidden داریم:

N*P*H

که در اینجا N تعداد کلمات قبلی است که مورد استفاده قرار می گیرد و P ابعاد لایه ی Projection است و H ابعاد لایه‌ی Hidden است.

- همچنین برای پیش بینی کلمه ی خروجی داریم:

H*V

که در اینجا H ابعاد لایه ی Hidden است و V ابعاد Vocabulary است. بنابراین برای پیچیدگی محاسباتی خواهیم داشت.

$$O(NNLM) = N * P + N * P * H + H * V$$

$$O(NNLM) = N * P + N * P * H + H * V$$

به دلیل اینکه حاصل ضرب $H * V$ بسیار بزرگ تر از $N * P$ و $N * P * H$ است (به دلیل اینکه V بسیار بزرگ است و عملا یک دیکشنری است) برای همین پیچیدگی زمانی NNLM می توانیم برابر H*V در نظر می گیریم.

(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)

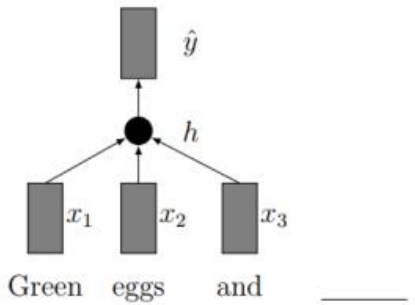
برای حل مشکل پیچیدگی محاسباتی $O(NNLM) = H * V$ موجود در NNLM که در قسمت قبل معرفی شد دو الکوریتم معرفی شد که پیچیدگی محساباتی محاسبه ی خروجی را کاهش دهد. این روش ها عبات اند از :

- Hierarchical Softmax

- Negative Sampling

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

- روابط ریاضی در شبکه مورد نظر:



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

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

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

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

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

طبق فرضیات مسئله داریم:

$$a_1 = Wx + b$$

$$h = \tanh(a_1)$$

$$a_2 = Uh + d$$

$$y_{pred} = softmax(a_2)$$

$$J = CrossEntropy = (y, y_{pred})$$

$$CrossEntropy = -\sum_i y_i \log(y_{predi})$$

$$a_1 = Wx + b$$
$$h = \tanh(a_1)$$
$$a_2 = Uh + d$$
$$y_{pred} = softmax(a_2)$$
$$J = CrossEntropy = (y, y_{pred})$$
$$CrossEntropy = -\sum_i y_i \log(y_{predi})$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y_{pred}} \times \frac{\partial y_{pred}}{\partial a_2} \times \frac{\partial a_2}{\partial h} \times \frac{\partial h}{\partial z_1} \times \frac{\partial z_1}{\partial x}$$

❖ $\frac{\partial J}{\partial y_{pred}} \times \frac{\partial y_{pred}}{\partial a_2}$ → $\frac{\partial J}{\partial y_{pred}} \times \frac{\partial y_{pred}}{\partial a_2} = y - y_{pred}$ →

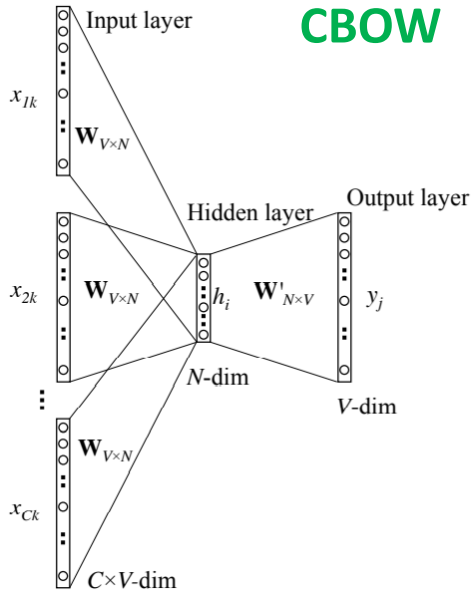
- ❖ $\frac{\partial a_2}{\partial h} = U$
- ❖ $\frac{\partial h}{\partial z_1} = 1 - \tanh^2(z_1)$
- ❖ $\frac{\partial z_1}{\partial x} = W$

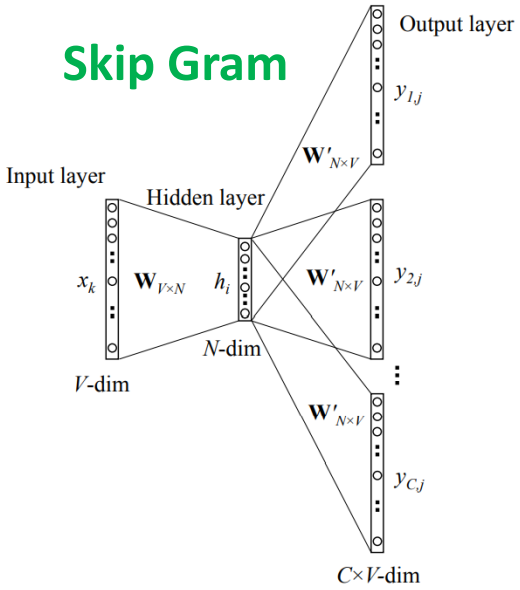
→ $$\frac{\partial J}{\partial x} = (y - y_{pred}) * U * 1 - \tanh^2(z_1) * W$$

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)





همانطور که گفته شد یکی از روش های تولید بردار از متون روش word2vec است. که این روش خود دو نوع دارد CBOW و Skip Gram که در روش CBOW ما بر اساس کلمات پیرامون کلمه ی Target سعی می کنیم که کلمه ی Target را پیش بینی کنیم بنابراین با استفاده از این روش می توانیم یک بردار بازنمایی از هر کلمات ایجاد کنیم. روش Skip Gram منطقی بر عکس CBOW دارد بر این اساس که ما کلمه ی target را به عنوان ورودی در نظر می گیریم و سعی می کنیم کلمات پیرامون کلمه ی Target را پیش بینی کنیم.

روش های وقوع همزمانی مبتنی بر شمارش (Cooccurence Count-based Methods) یک سری روش آماری و احتمالی هستند که به دنبال پیدا کردن شباهت بین کلمات هستند (کلماتی که در کنار هم می آیند) مثل n-gram یا GLOVe این روش ها بر این اساس هستند که یک ماتریس تشکیل می دهند که وقوع کلمات موجود در دیکشنری را پیش بینی می کند. مثلا در مثال زیر ما ماتریس وقوع دو کلمه ی a و penny با کلمات موجود در دیکشنری فرضی در در نظر می گیریم.

penny wise and pound foolish

a penny saved is a penny earned

	a	and	earned	foolish	is	penny	pound	saved	wise
a	0	0	0	0	0	2	0	0	0
penny	0	0	1	0	0	0	0	1	1

با توجه به نکات گفته شده اگر بخواهیم مزایا و معایب word2vec نسبت به روش های Cooccurence Count-based داریم:

مزایا

- روش های Cooccurence Count-based همانطور که گفته شد پیوسته به دنبال پیدا کردن کلماتی هستند در کنار هم می آیند اما word2vec روابط ویژگی ها و رابط پیچیده تری را استخراج می کند. و به نوعی می توان گفت word2vec اطالعات بدست آمده در occurence Count-based را نیز در خود دارد. بنابراین ویژگی Word2vec این است که ویژگی ها و روابط موجود در بین کلمات را به خوبی استخراج می کند
- روش های occurence Count-based همانطور که گفته شد مبتنی بر ماتریس هستند که از کلمات دیکشنری ایجاد می شود. این ماتریس بسیار بزرگ $(V \times V)$ و sparse است و این در حالی است که روش های word2vec بر اساس سایز ورودی ها (corpus) بسیار scalable هستند و از این نظر مشکلی ندارند.

معایب

- روش های occurence Count-based همانطور که گفته شد صرفا مبتنی بر ایجاد ماتریس هستند بنابریان بسیار سریعتر از Word2vec هستند
- روش های occurence Count-based همانطور که گفته شد صرفا مبتنی بر ایجاد ماتریس هستند بنابراین پیچیدگی محاسبتی بسیار کمی نسبت به فرآیند آموزش شبکه های word2vec دارند.

(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)

خیر این حرف لزوما درست نیست زیرا همانطور که در از صورت سوال مشخص است ما با یک ضرب Inner Product (ضرب داخلی مواجه هستیم). در این ضرب مثلا اگر یکی از المان (مثلا u_w^A) در یک ضریبی مثل λ ضرب شود و المان دیگر (مثلا v_w^A،) در عکس این ضریب یعنی $\frac{1}{\lambda}$ باز هم ضرب داخل آنها یکسان می شود . بنابراین نمی توان نتیجه گرفت که هرکدام از برداهای v_w^A وv_w^B باهم هستند. یعنی اگر ضرب داخلی آنها باهم برابر باشد به این معنا نیست که تک به تک آنها برابر باشند. زیرا مثلا در همین مثالی که گفتیم بردارها عملا در یک ضریبی هر کدام ضرب شده اما همچنان ضرب داخلی ما یکسان است.

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

زیرا همانطور که بخش های قبل گفته شد به دلیل اینکه در آخرین لایه ما باید احتمال انتخاب همه کلمات موجود در دیکشنری V را محاسبه کنمی و این عملیات بسیار هزینه بر است بنابراین به صورت مستقیم از Softmax استفاده نمی کنیم. بنابراین می توان همانطور که در قسمت های قبل گفته شد از روش هایی مثل :

- Hierarchical Softmax
- Negative Sampling

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

همانطور که گفته شد روش های وقوع همزمانی مبتنی بر شمارش (Cooccurence Count-based Methods) یک سری روش آماری و احتمالی هستند که به دنبال پیدا کردن شباهت بین کلمات هستند (کلماتی که در کنار هم می آیند) مثل n-gram یا GLOVE این روش ها بر این اساس هستند که یک ماتریس تشکیل می دهند که وقوع کلمات موجود در دیکشنری را پیش بینی می کند. بنابراین ماتریسی که برای Glove ایجاد می شود آموزش می بیند حافظه ی بسیار زیادی را اشغال می کند همانطور که در اسلایدها قبلی گفته شد در صورتی که در روش Word2vec اینطور نیست و محاسبات ما حافظه ی بسیار کمتری استفاده می کنند در برابر ماتریس های Cooccurence (که بر مبنای دیکشنری که ابعاد بزرگی دارد ایجاد می شود). بنابراین از بین روش‌های Glove و Word2vec، Glove حافظه ی بیشتری اشغال می کند.

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

- یکی از مشکلاتی که هم در روش Word2vec و Glove وجود دارد این است که دو روش کلمات با چندین معنا را از هم تشخیص نمی دهند. زیرا کلمات و بردارها روابط یک به یک دارند.
- یکی دیگر از مشکلاتی که این دو روش دارند این است که گاهی دو کلمه با دو معنای خلاف هم در یک مجموعه (فضای حالت یکسان قرار می گیرند) برای مثال کلمه ی زشت و زیبا یا راست و چپ از نظر معنایی کاملا متفاوت هستند و تضاد هم هستند اما در یک فضای برداری قرار می گیرند.
- مشکل دیگری که برای این دو روش وجود دارد این است که این دو روش به شدت وابسته به کلمات دیکشنری هستند بنابراین اگر کلمه ای خارج از دیکشنری بیاید عملا این دو روش نمی تواند بردار مطلوبی برای آن در نظر بگیرند و به صورت random اختصاص می دهند.

(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.

$$J(R, \tilde{R}) = \sum_{i,j} f(x_{ij})(r_i^T \tilde{r}_j - \log x_{ij})^2$$

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

همانطور که در بخش ب سوال ۲ دیدیم ثابت کردیم که زمانی که ما دو ضرب داخلی داریم و حاصل آن ها باهم برابر است لزوما مقدار هر یک از المان ها هم با یگدیگر برابر نیست.

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

$$v_w^A \neq v_w^B$$

همچنین برای اینکه ثابت کنیم که تابع هزینه ی Glove محدب نیست باید ثابت کنیم که تابع هزینه در برخی از نقاط هزینه ی یکسانی دارد یا هزینه بیشتر می شود.

همچنین برای permutation invariance می دانیم:

$$f((x_1, x_2, x_3)) = f((x_2, x_1, x_3)) = f((x_3, x_1, x_2))$$

$$J(R, \tilde{R}) = \sum_{i,j} f(x_{ij})(r_i^T \tilde{r}_j - \log x_{ij})^2$$

حال با توجه به فرضیات گفته شد و تابع هزینه Glove (در قسمت بالا) و با توجه به فرضیات مسئله اینکه گفته شده ماتریس های R و $\tilde{R}$ تعبیه کلمات r و $\tilde{r}$ در خود دارند. بنابراین اگر هریک از R و $\tilde{R}$ را عوض کنیم آنگاه مقدار تابع هزینه همچنان یکسان باقی خواهد ماند. اگر بیایم مثلا میانگین آنها را در نظر بگیریم آنگاه یکسان خواهند شد و این مسئله نیز باعث می شود که تابع هزینه مقدارش بیشتر شود (زیرا دو مقدار که برابر هستند بیشترین ضرب داخلی را دارند).

مسئله‌ی ۳. Transformers and Attention models

به سوالات زیر پاسخ دهید.

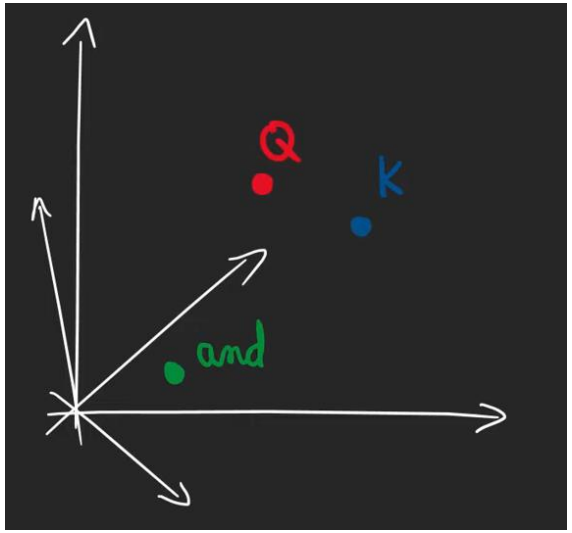
۱. یکی از مراحل مقدماتی در پیش پردازش داده‌ها برای حوزه پردازش متن، توکن‌سازی [۱۰] از جمله‌های پایگاه داده است. یکی از چالش‌های این مرحله، این است که ترتیب کلمات در جمله با این‌کار از بین می‌رود. برای حل این مشکل رویکرد Positional Embedding مطرح می‌شود. به صورت کلی و کوتاه توضیح دهید که چگونه این رویکرد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

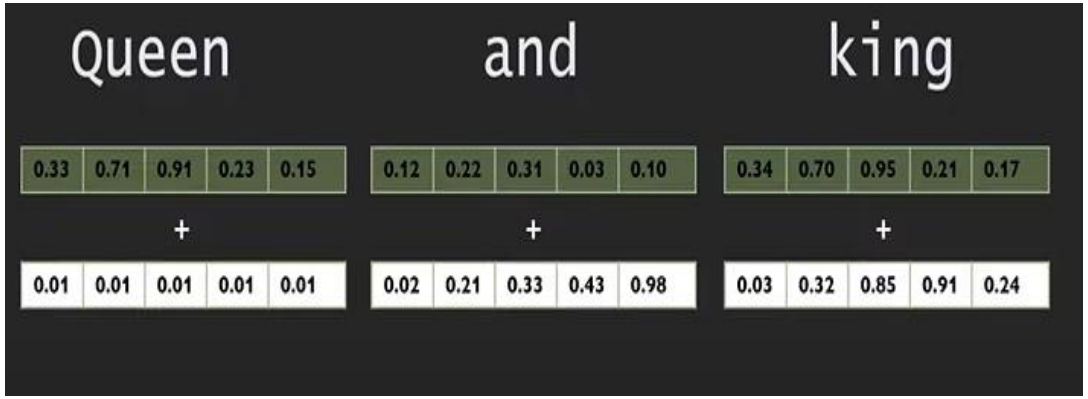
همانطور که گفته شد یکی از مشکلات در مسیر تبدیل دنباله ها (جملات) بهتوکن این است که ترتیب کلمات بهم بخورد و این مسپله می تواند در پیش بینی خروجی ها مارا دچار مشکل کند. برا حل این مشکل ایده ای که مطرح شد این بود که می توانیم برای هر کلمه یک اندیس به عنوان نشان دهنده ی جایگاه (position) در نظر بگیریم. که مسئله می تواند نشان دهنده ی ترتیب کلمات باشد. مثلا فرض می کنیم که ما جمله ی زیر را داریم:

King and Queen

در این حالت ما بردار های بازنمایی هر یک از کلمات را استخراج می کنیم اما همانطور که گفته شد ما پس از توکن سازی از جملات ورودی ما ترتیب عملا می تواند بهم بخورد و این مسئله می توان مطلوب نباشد و ما را دچار مشکل کند. بنابراین ما یکسری اندیس ای پوزیشن به عنوان برای هر کلمه به صورت منحصر به فرد در نظر می گیریم.

ما به دنبال این هستیم که برای embedding هر کلمه یک بردار هم اندازه با آن در نظر بگیریم که نشان دهنده ی جایگاه (پوزیشن) هر بردار است و ما وقتی فضای embedding هر کلمه را بدست آوردیم با اضافه کردن بردار position به هر کدام مقداری بردار embedding هر کلمه را شبفت بدهیم که آن کلمه در مجموعه ی (کلاستر) کلمات با اندیس مشترک قرار بگیرد. اما برای این کار باید ملاحظاتی را نیز در نظر بگیریم از جمله اینکه embedding کلمات در فضای مختصات ارتباط معنایی باهم دارند مثلا قطعا طبق مثال مورد نظر کلمه ی queen و king بهم نزدیک تر هستند تا کلمه ی and بنابراین اگر مثلا مقدار بردار position خیلی بزرگ باشد می تواند انقدری این کلمات را شیف بدهد که عملا روابط معنایی بین آنها را تحت تاثیر قرار دهد. بنابراین ما به دنبال آن هستیم که این شیف دادن این مشکلات را به وجود نیاورد. همچنین باید این اندیس ها طوری تعیین شوند که برای کلمات هم جایگاه یکسان باشند و اگر کلمات یک دنباله تغییر کردند این بردارهای position تغییر نکنند.





برای تعیین این اندیس ها ما چند رویکرد داریم:

- یکی از رویکرد ها بر این اساس است که ما می توانیم از یک شروع کنیم و به ترتیب برای هر کلمه یک اندیس در نظر بگیریم. این ایده نسبتا خوب است و برای کلمات هم جایگاه بردار position یکسان ایجاد می کند. اما مشکلی که دارد این است که عملا اگر طول جملات زیاد باشد می تواند مشکل ساز باشد. این مشکل به این علت است که ما وقتی داریم توزیع کلمات را در فضای مختصات محاسبه می کنیم کلمات با معنی یکسان در این توزیع نزدیک به هم هستن و ارتباط معنایی آن ها با فاصله ای که در این فضای مختصات باهم دارند رابطه ی مستقیم دارد. بنابراین وقتی طول جملات بزرگ باشد و ما یک اندیس بزرگ (که نشان دهنده پوزیشن است) را به بردار embedding هر کلمه اضافه کنیم ممکن است این ارتباط معنایی را از بین ببریم.است.

- برای حل این مشکل ایده ی دیگری ارائه شد که اندیس ها را از بین یک بازه ی معین انتخاب کنیم که این بازه ی معین مثلا بین صفر و یک باشد یعنی برای اولین کلمه اندیس صفر و برای آخرین کلمه عدد یک استفاده کنیم. اما مشکلی که در این روش وجود دارد این است که عملا ما تضمینی وجود ندارد که که کلمات وجود در و جمله با یک جایگاه اندیس یکسان در یافت کنند.

- ایده ی بعدی که مطرح شد استفاده از توابع پریودیک که در بازه ی خاص هستند بود مثل sin و cos یه این صورت که برای پوزیشن های فرد و زوج بر اساس توابع سینوس و کسینوس یک اندیس اختصاص دهیم.

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

❖ ایده ی بعدی که مطرح شد استفاده از توابع پریودیک که در بازه ی خاص هستند بود مثل sin و cos یه این صورت که برای پوزیشن های فرد و زوج بر اساس توابع سینوس و کسینوس یک اندیس اختصاص دهیم.

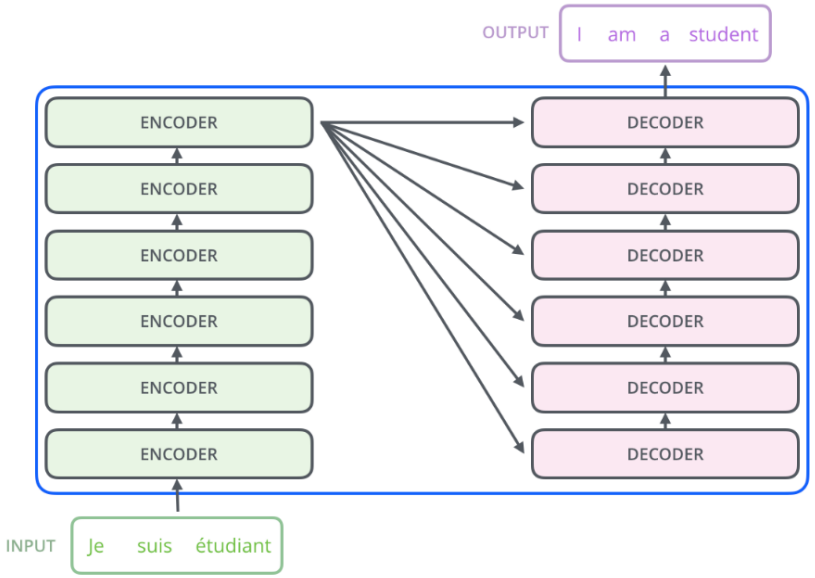
$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

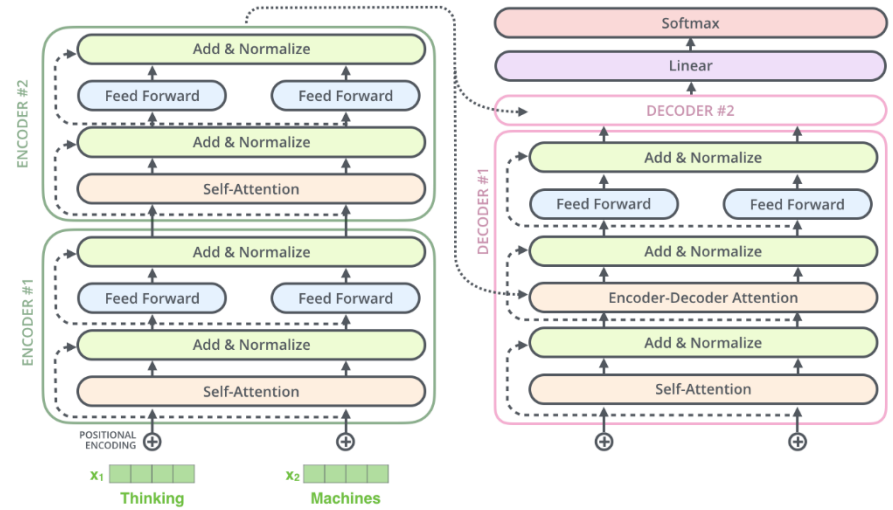
$$\omega_k = \frac{1}{10000^{2k/d}}$$

استفاده از این توابع برای در نظر گرفتن پوزیشن برای جایگاه هر کلمه بسیار مناسب بود زیرا در این حالت ما مقدار اندیس ها را در یک بازه ی معین قرار داده ایم. همچنین این بازه سبب نمی شود که دو اندیس یکسان برای جایگاه های مختلف در نظر گرفته شود. همجنین ما عملا با شیف دادن بردار ها عملا مشکلی برای ارتباط معنایی بین داده ها به وجود نمی آوریم. بنابراین ما در نظر گرفتن بردارهایی به عنوان پوزیشن کلمات و جمع کردن آنها با بردار هر کلمه ی ورودی است که در این صورت ما می توانیم با این رویکرد که positional embedding نام دارد با مسئله ای که به عنوان به هم خوردن ترتیب جایگاه کلمات است مقابله کنیم زیرا عملا ما هر کلمه را با توجه به جایگاه آن برایش برداری در نظر گرفته ایم.

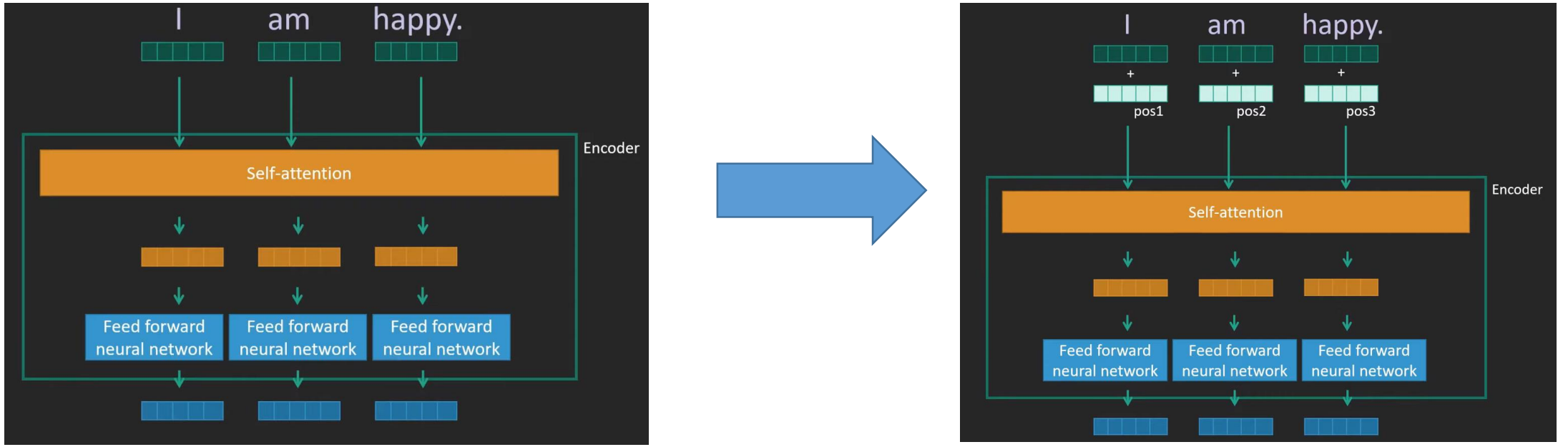
۲. یکی از کاربردهای رویکرد گفته شده در قسمت قبل، استفاده از Transformer ها است. این معماری چگونه مشکل گفته شده(بهم خوردن ترتیب کلمات) را برطرف می‌کند؟ (۴ نمره)

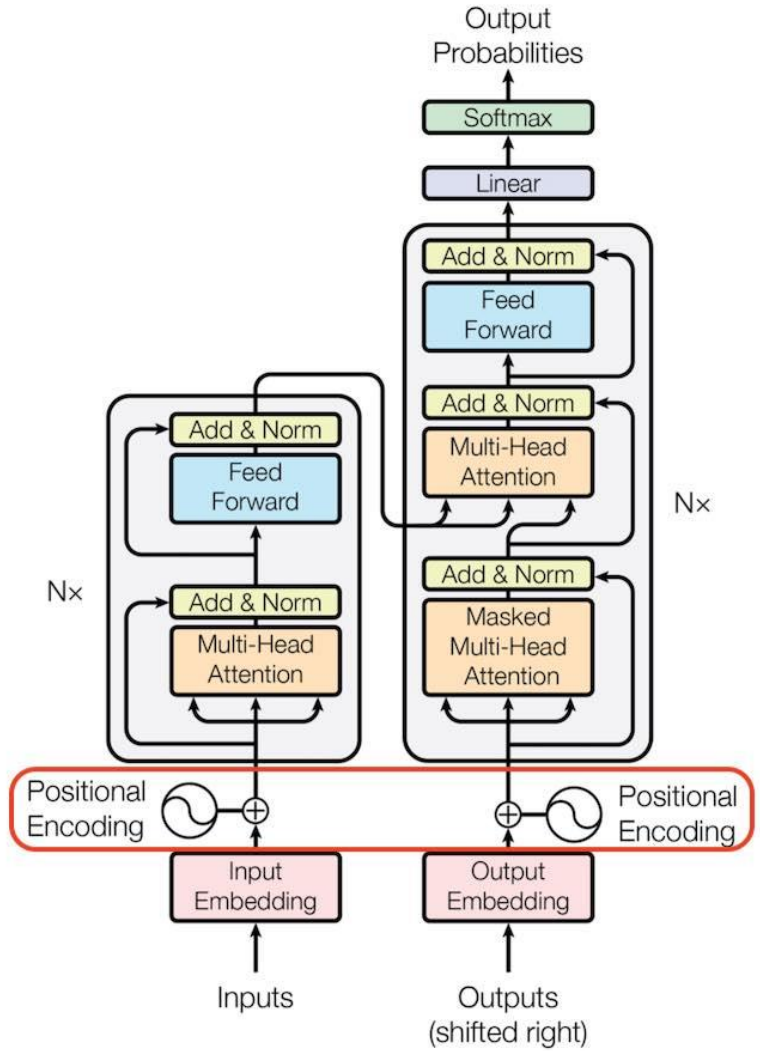
Transformer ها یکی از مدل های شبکه های عصبی عمیق هستند که برای کاربردهایی مثل ترجمه ی ماشینی مورد استفاده قرار می گیرند. که کاملا با مکانیزم self-attention سازگار هستند که این مکانیزم برای توجه به کلمات موجود در یک جمله و پیدا کردن روابط بین آنها مناسب است و این در حالی است که در attention ما توجه بین کلمات بین جملات مختلف و روابط بین آنهاست. بخش های مختلف ورودی ها را وزن دهی می کنند. در واقع برخلاف معماری های encoder-decoder است (که نسبتا زمانگیر هستند چون ترتیبی است) که کلمه به کلمه ورودی ها را بررسی می کرد و سپس خروجی را محاسبه می‌کرد. اما در transformer ها ما به صورت موازی و یکجا کل داده های ورودی را پردازش می کنیم. معماری ترنسفور بر این اساس است که از تعدادی لایه ی encoder و سپس تعدادی لایه ی decoder تشکیل شده است. هر لایه ی ترنسفور متشکل از یک لایه ی self-attention و یک شبکه ی عصبی feed forward است. ویژگی ترنسفور ها بر این اساس است که به جای شبکه های rnn معمولی که ورودی ها به ترتیب وارد لایه های بعدی می شود در ترنسفورمرها ورودی ها به صورت موازی پردازش می شوند و عملا ترتیب در آنها معنایی ندارد و این مسئله ی عدم رعایت ترتیب می تواند مشکل ساز شود زیرا ممکن است ما نیاز داشته باشیم که ترتیب حفظ شود و این مسئله ی عدم ترتیب دلخواه ما نباشد.





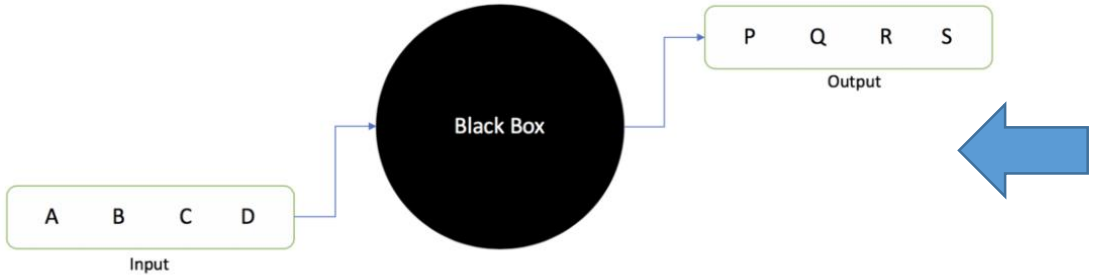
برای حل مشکل موجود که در واقع مسئله ی عدم رعایت ترتیب است transformer ها یک رویکرد رد نظر گرفته اند که در آن به ورودی ها یک بردار position نیز اختصاص داده می شود (همانطور که در شکل زیر می بینیم). بنابراین با استفاده از این راهکار ما می توانیم ترتیب کلمات را به نوعی حفظ کنیم و از مشکلات به وجود آمده جلوگیری کنیم. بنابراین در هنگام پیش بینی کلمات خروجی عملا ترتیب را نیز در نظر می گیریم. بنابریان عملیات موازی سازی پردازش کلمه ها در transformer ها نه تنها سبب نمی شود که ترتیب ورودی ها و روابط بین آنها دچار مشکل شود بلکه سرعت را به ورت چشم گیری افزایش می دهد.



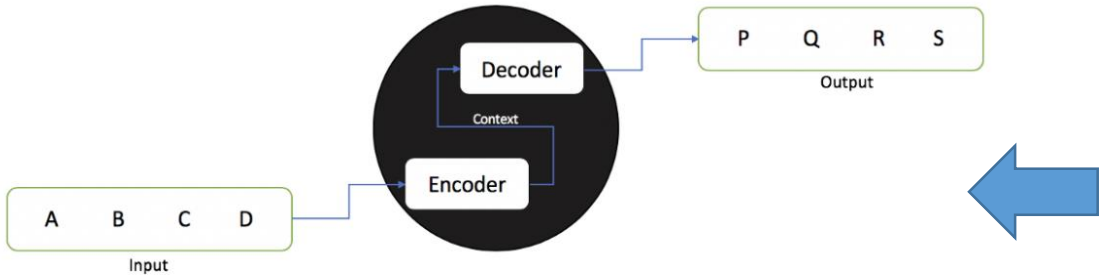

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

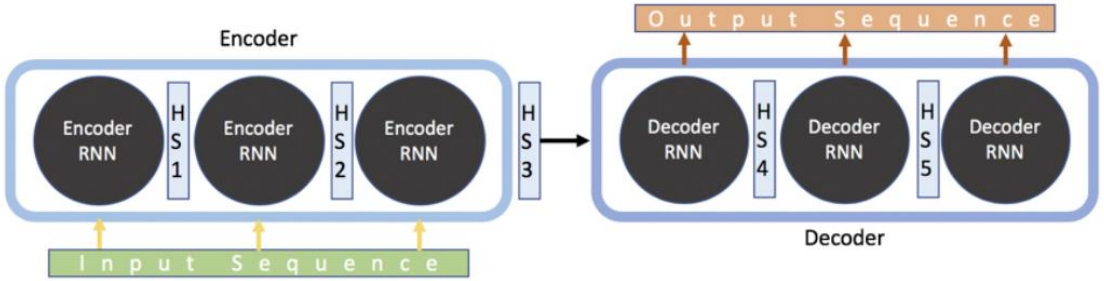
۳. محدودیت معماری Encoder-Decoder ای که از مکانیزم توجه استفاده نمی‌کند را در حوزه ترجمه ماشینی بیان کنید و سپس به صورت خلاصه توضیح دهید که چگونه مکانیزم توجه می‌تواند این مشکل را برطرف کند. (۶ نمره)

یکی از روش های ترجمه ی ماشینی استفاده از مدل های seq2seq هست. ایده‌ی این روش ها بر این اساس است که ما یک دنباله ای از کلمات را به عنوان ورودی دریافت کنیم و پس از استخراج ویژگی های کلمات، با استفاده از آنها کلمات دنباله ی خرجی را تولید کنیم. همانطور که در شکل مقابل می بنید.

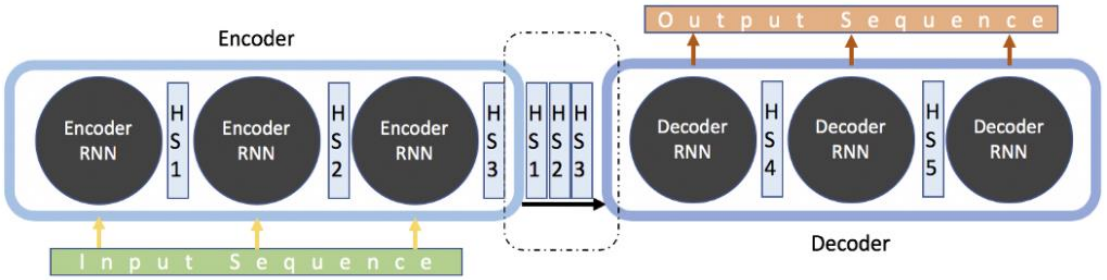


بنابریان بر اساس نکات گفته شده ایده ی معماری encoder-decoder شکل گرفت. این معماری همانطور که از اسم آن پیداست از دو بخش encoder و decoder تشکیل شده است. وظیفه ی بخش encoder بر این اساس است که دنباله ای از ورودی را گرفته و یک بردار Context ایجاد می کند. این بردار Context نمایانگر تمامی کلمات موجود در رشته های ورودی است. وظیفه ی بخش Decoder این است که با استفاده از اطلاعات استخراج شده و موجود در context دنباله ی خروجی را تولید کند.



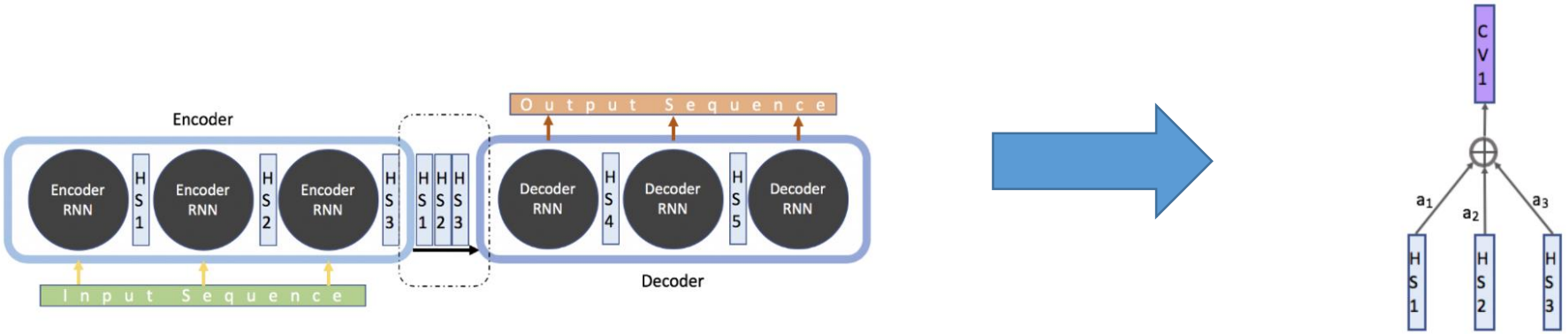


معماری encoder و decoder در واقع یک شبکه ی بازگشتی است (RNN) مثل LSTM و GRUو.... طول بردار context متغییر است و وابسته به طول دنباله ی ورودی است و معمولا توانی است ۲ است مثلا ۲۵۶ و ۵۱۲ و.... اما مشکلی در این روش وجود دارد این است که عملا ظرفیت context محدود است و طولانی نیست. بنابراین تا یه حدی می توانیم سایز آن را بزرگ در نظر بگیریم. اما مسئله ای که وجود دارد این است که زمانی که طول دنباله ی کلمات ورودی بزرگ می شود. مشکلی که پیش می آید این است که اطاعات مربوط به کلمات مجود در دنباله ی ورودی فراموش می شود. در واقع همانطور که گفته شد سایز Context عموما کوچکتر از سایز دنباله ی ورودی است بنابراین وقتی سایز دنباله بیش از اندازه بزرگ شود در این حالت بخشی از اطلاعات مهم موجود در دنباله ی ورودی به نوعی فراموش می شود و بخش هایی که به نظر خیلی مهم نیستند در Context باقی بماند و این مسئله می تواند سبب شود خروجی های ما به خوبی تولید نشود.

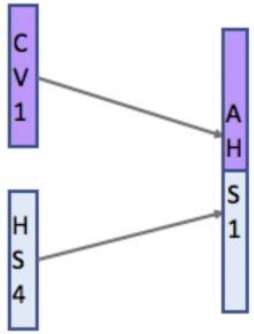


برای حل این مشکل روش Attention ارائه شد که این این روش بر این اساس است به مدل اجازه می دهد تا در هر مرحله از دنباله خروجی بر روی بخش های مختلف دنباله ورودی تمرکز کند و اطلاعات Context را از ابتدا تا انتها حفظ کند. (در واقع مشکل اصلی این بود که یک hidden state برای لایه ی آخر encoder کافی نبود.) در این روش به دنبال این هستیم که تا بخش هایی که مهم تر هستند را به نوعی حفظ کنیم و بخش های دیگر را به نوعی فرامش کنیم بنابریان با استفاده از این روش مشکل مطرح شده تا حدی خوبی بر طرف می شود.

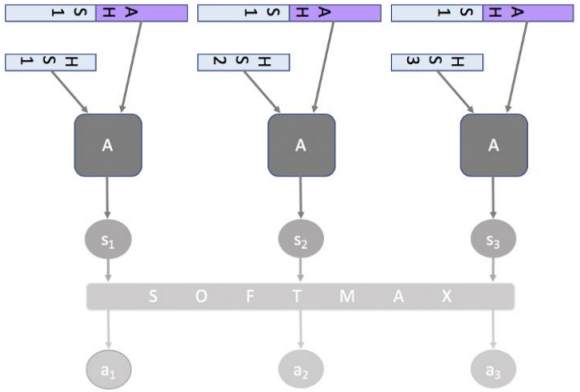
همچنین در این روش برای hidden state های هر مرحله یک وزن در نظر بگریم و این وزن ها اهمیت هر بخش را مشخص کند. بنابراین می توانیم بخش های مهم تر را در hidden state حفظ کنیم و با مسئله ی فراموشی که در به خاطر محدودیت های context وجود داشت مقابله کنیم. بنابراین در گام زمانی بردار Context یک جمع وزنی از لایه های hidden است همانطور که در شکل پایین می بنید:



همچنین در هر گام زمانی بردار Context را با hidden state آن مرحله concatenate می کنیم در این حالت یک بردار attention برای آن مرحله ایجاد می کنیم که برای پیش بینی آن مرحله می توانیم از آن استفاده کنیم و جایگزین hidden state می شود.



همچنین ما در هر مرحله score هایی ار محاسبه می کنیم که تعیین می کند که تا چه حد ورودی (که توسط hidden state مربوط به خودش بازنمایی شده با خروجی قبل که توسط لایه ی attention بانمایی شده تطابق دارد. و این کار برای همه ی ورودی های با توجه به خروجی قبل انجام می شود. بنابریان یک لایه ی سافت مکس بر روی همه این score ها قرار می گیرد و خروجی آن attention score برای هر ورودی است.



بنابراین حال ما می دانیم که کدام بخش از ورودی ها برای تولید بخش های مختلف خروجی مناسب است. از این رو، اکنون می دانیم که کدام بخش از ورودی برای پیش بینی هر یک از نمونه های توالی خروجی مهم تر است. بنابراین در مرحله آموزش، مدل یاد گرفته است که چگونه نمونه های مختلف را از دنباله خروجی به دنباله ورودی Align کند.