مسئله ۱ –

قسمت الف) اولا قابل به ذکر است که به دلیل اینکه در شبکه شکل آ ، اتصال واحد مخفی-واحد مخفی نداریم، نیاز به این داریم که واحد های خروجی تمام اطلاعات مربوط به گذشته را ضبط(capture) کرده و از آن برای پیش بینی آینده استفاده کند. یک شباهتی که بین این دو معماری شبکه وجود دارد این است که هر دو در هر واحد زمانی، یک خروجی تولید میکنند.

ابتدا در مورد معماری شکل ب، صحبت میکنیم. میدانیم که هر تابعی که توسط یک ماشین تورینگ قابل محاسبه است، توسط اینچنین معماری ای با سایز محدود نیز قابل محاسبه است.لذا این نوع RNN نوعی پیاده سازی ماشین تورینگ است.اگر تابع فعالساز مورد استفاده در این معماری را tanh در نظر بگیریم و فرض کنیم خروجی شبکه، گسسته است(مثلا برای پیش بینی کلمه یا کاراکتر) لذا در خروجی برای پیش بینی مورد نظر از احتمال لگاریتمی نرمال نشده(احتمال به هر کلمه یا کاراکتر و پیدا کردن بالاترین احتمال) استفاده میکنیم.در مرحله post-processing هم از تابع softmax برای ایجاد بردار y که برداری نرمال شده روی احتمالات است، استفاده میکنیم. اگر بخواهیم روابط این معماری را نشان دهیم، میدانیم پس از مقداردهی اولیه به h(0) ، در فاز انتشار رو به جلو روابط زیر را داریم:

$$a^t = b + Wh^{t-1} + Ux^t$$

$$h^t = \tanh(a^t)$$

$$o^t = c + Vh^t$$

$$\hat{y}^t = softmax(o^t)$$

گمشدگی یا *loss* با داشتن یک دنباله x و جفت دنباله y ، برای $y(t)$ با ورودی $x(1)$ تا $x(t)$ نیز برابر است با:

$$-\sum_t \log p_{model}(y^t|\{x^1, \dots, x^t\})$$

توجه شود بخاطر اینکه در این معماری، فاز انتشار رو به جلو به صورت *sequential* است، یعنی هر خروجی هر مرحله زمانی تنها وقتی خروجی مرحله قبل ایجاد شده است، قابل دستیابی است، لذا زمان اجرای آن با موازی سازی بهبودی پیدا نمیکند و پیچیدگی زمانی آن $o(T)$ است که T همان تعداد *time step* ها است.چون هیدن استیت هر مرحله در فاز انتشار رو به جلو باید نگهداری شود تا بعدا در فاز انتشار رو به عقب مورد استفاده قرار گیرد، لذا پیچیدگی حافظه ی این مدل نیز $o(T)$ است.الگوریتم انتشار رو به عقب مورد استفاده در این معماری، *back-propagation through time* است. این معماری بسیار با توان پردازشی و قدرت بالا اما پر هزینه برای آموزش است.

حال در مورد معماری شکل ب صحبت میکنیم. این معماری به دلیل فقدان ارتباطات بین واحد های مخفی، از توان پردازشی و قدرت پایین تری برخوردار است و به همین دلیل نمیتواند یک ماشین تورینگ ***universal*** را پیاده سازی کند.

مزیت حذف کردن ارتباطات واحد مخفی-واحد مخفی این است که برای هر تابع هزینه مبتنی بر مقایسه پیش بینی مرحله t و هدف آموزشی مرحله t ، تمام مراحل زمانی ***decouple*** میشوند لذا فاز آموزش میتواند موازی سازی شود و لذا پیچیدگی زمانی این مدل را پایین بیاورد(نیازی به محاسبه خروجی مرحله قبل برای خروجی مرحله بعد نیست یعنی مدل ***sequential*** نیست).

قسمت ب) این چنین مدلی میتواند توسط ***teacher forcing*** آموزش داده شود.این متد یک پروسه ای است که از مفهوم ***MLE*** استفاده میکند که در مرحله آموزش، مدل خروجی ***ground truth*** در مرحله t را به عنوان ورودی در مرحله $t+1$ دریافت میکند.از این متد برای جبران فقدان روابط واحد مخفی-واحد مخفی برای فاز انتشار رو به عقب استفاده میشود. اما یک نقطه ضعف ***teacher forcing*** زمانی است که شبکه بخواهد در آینده در ***closed loop mode*** استفاده شود یعنی وقتی که باید خروجی شبکه(یا نمونه هایی از توزیع خروجی) به عنوان ورودی استفاده شود.در این گونه مواقع ورودی گفته شده ای که در فاز آموزش داریم، کاملا میتواند با ورودی فاز تست متفاوت باشد. یکی از راه حل ها برای این مشکل این است که همزمان از ورودی نوع ***teacher-forced*** و ***free-running*** استفاده کنیم.

در مرحله تست، در این نوع معماری، ما دنباله هدف را نداریم که به عنوان ورودی از آن استفاده کنیم.در اینجا مدل کاملا باید متکی به خود عمل کند و اگر یک خروجی بد تولید کند، اثرات این خروجی بد در کل دنباله منتشر میشود.به این مشکل ***exposure bias*** میگویند.

قسمت ج) راه حل برطرف کردن این مشکل این است که بتوانیم به نحوی وابستگی مدل به ***ground truth*** را از بین ببریم. یکی از این راه ها ***scheduled sampling*** است که مدل به صورت رندوم انتخاب میکند که از کجا نمونه برداری انجام دهد.این یعنی تصمیم گیری راجع به اینکه آیا از ***ground truth*** برای ورودی مرحله بعدی استفاده کنیم یا اینکه خروجی مرحله قبلی را به عنوان ورودی مرحله بعد، استفاده کنیم.این روش به طور قابل توجهی ***reliance*** مدل را کاهش میدهد.

قسمت د) همانطور که میدانیم در شبکه های بازگشتی معمولی، در فاز انتشار رو به عقب یک سری گرادیان در هم ضرب میشوند و از آن برای آپدیت پارامتر های شبکه استفاده میشود.حال در صورتی که عمق شبکه زیاد باشد، این ضرب شدن گرادیان ها در یکدیگر میتواند به سمت صفر میل کند که به این رویداد ناپدید شدن گرادیان گفته میشود(وقتی گرادیان ها

کوچک تر از ۱ باشند، با ضرب شدن متوالی و زیاد طبیعتا به سمت صفر میل میکند). از طرف دیگر این ضرب شدن های متوالی میتواند باعث میل به اعداد بسیار بزرگ شود(وقتی گرادیان ها اعداد بزرگ تر از ۱ باشند).در صورت روی دادن هریک از این مشکلات، آموزش شبکه با مشکل مواجه میشود به طوری که یا آموزش بسیار کند و یا حتی متوقف میشود و یا آموزش اشتباه و بد صورت میگیرد.

در شبکه های بازگشتی، هرچه وابستگی های بلند مدت بیشتر باشد، یعنی شبکه مایل به آموزش روی دنباله های بلند تری باشد، لذا مجبور است عمق شبکه را بیشتر کند(در رویکرد بد) و این مشکل پدید می آید که باعث فراموشی این وابستگی های بلند مدت یا آموزش بد روی این وابستگی ها میشود.

قسمت ه) یکی از راه های جلوگیری از انفجار گرادیان در شبکه های بازگشتی، روش ***norm clipping*** است که در این روش به صورت خلاصه وقتی حاصل ضرب گرادیان ها از یک آستانه ای بیشتر شد، به عددی کمتر ***rescale*** میشود.شبه کد این الگوریتم در تصویر زیر آمده است:

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

یکی از راه های جلوگیری از ناپدید شدن گرادیان نیز روش ***soft constraint*** است که در اصل یک رویکرد برای مسائل بهینه سازی است. در مسائل بهینه سازی سعی در مینیموم کردن یک تابع هدف داریم.حال برای کنترل مقادیر پارامتر ها میتوانیم از یک محدودیت اضافی در بهینه سازی استفاده کنیم که این محدودیت به شکل یک مقدار پنالتی به تابع هدف اضافه میشود. حال با در نظر گرفتن این مقدار پنالتی سعی در کاهش تابع هدف میکنیم(همان مفهوم ***regularization***)

این مقدار پنالتی باعث میشود که گرادیان ها از یک مقداری کوچک تر نشوند و لذا ناپدید شدن گرادیان رخ ندهد.

مسئله ۲ –

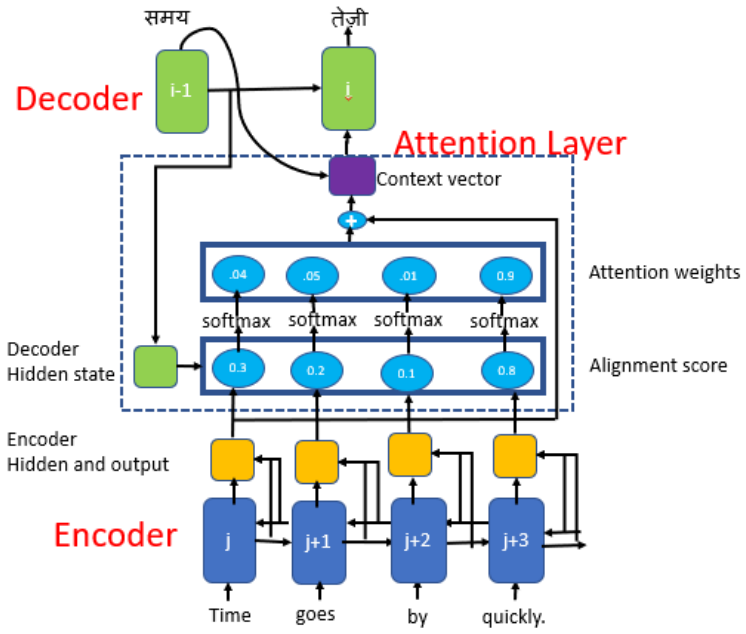
قسمت الف) ابتدا از لحاظ قدرت مدل کردن این سه رابطه را بررسی میکنیم.روش اول یا همان روش ***dot*** چون یک ضرب نقطه ای معمولی بین $h_e^{(t)}, h_d^{(t)}$ است، لذا میبینیم که پارامتری برای یادگیری وجود ندارد لذا قدرت مدل کمی دارد.در روش دوم هم $h_e^{(t)}, h_d^{(t)}$ به صورت مستقیم در هم ضرب نمیشوند لذا اثربخشی المان های آن ها نادیده گرفته

میشود و قدرت مدل بیشتر از روش اول دارد اما باز هم زیاد نیست. اما در روش سوم چون این دو مستقیما باهم *concat* میشوند لذا در تعامل اند و در نهایت *scale* میشوند لذا قدرت مدل آن از همه بیشتر است.

از لحاظ پیچیدگی محاسباتی واضح است که روش اول از روش دوم محاسبات کمتری دارد(بخاطر عدم وجود ماتریس وزن) و روش سوم هم از روش دوم بخاطر *tanh* و ماتریس *V* ، محاسبات بیشتری دارد.

برای مقایسه این سه روش در عبور گرادیان، توجه شود که روش اول و دوم ، یک ضرب ساده هستند و گرادیان آن ها بخاطر خطی بودنشان، باعث انفجار یا ناپدیدی نمیشود اما روش سوم ضرب ها از *tanh* رد میشوند که میدانیم این تابع غیرخطی میتواند باعث مشکل در عبور گرادیان شود.

قسمت ب) ابتدا به توضیح در مورد مکانیزم توجه *bahdnau* میپردازیم.در سال ۲۰۱۵ آقای *bahdnau* یک مکانیزم توجه ارائه داد که ترازبندی(*align*) و ترجمه را به صورت مشترک و همزمان یاد میگرفت.اسم دیگر این نوع مکانیزم توجه *additive/concat* است چون از ترکیب خطی حالت(*state*) های ورودی و خروجی استفاده میکند.در این روش ارائه شده تمام حالت های مخفی انکودر و دیکودر(در فاز انتشار رو به جلو و عقب) برای تولید *contex vector* استفاده میشود(برعکس وقتی که از مکانیزم توجه استفاده نمیکردیم و فقط آخرین حالت مخفی انکودر استفاده میشد در *seq2seq*).در این روش، مکانیزم توجه، دنباله های ورودی و خروجی را توسط یک امتیاز ترازبندی(*alignment score*) که در فاز *feed forward* ایجاد میشود، ترازبندی میکند.این رویکرد باعث میشود که به اطلاعات مرتبط و مهم در منبع ورودی بیشتر توجه شود.این مدل بر اساس یک بردار *context* که در اصل با توجه به مکان ورودی و پیش بینی های قبلی ایجاد شده است، کلمه ی هدف را پیش بینی میکند.



Bahdanau et al. attention mechanism

لایه توجه(*attention layer*) شامل لایه ترازبندی(*alignment layer*) و وزن های توجه(*attention weights*) و همان بردار *context* است. امتیاز ترازبندی به ما نشان میدهد که چقدر ورودی در موقعیت *j* با خروجی در موقعیت *i* تطابق دارد.این امتیاز دهی بر اساس حالت مخفی قبلی دیکودر *s(i-1)* درست قبل از پیش بینی کلمه هدف و حالت مخفی *h(j)* در دنباله ورودی است:

$$e_{ij} = a(s_{i-1}, h_j) = score(h_d^{(t)}, h_e^{(s)})$$

به جای اینکه انکودر کل اطلاعات ورودی را به یک بردار با سایز ثابت انکود کند، دیکودر تصمیم میگیرد که کدام قسمت از جمله ورودی باید بیشتر مورد توجه واقع شود.بردار ترازبندی که سایز برابر با جمله ورودی دارد در هر مرحله دیکودر، محاسبه میشود.

برای محاسبه وزن های توجه، از تابع *softmax* بر روی امتیاز های توجه استفاده میکنیم:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{Tx} \exp(e_{ik})}$$

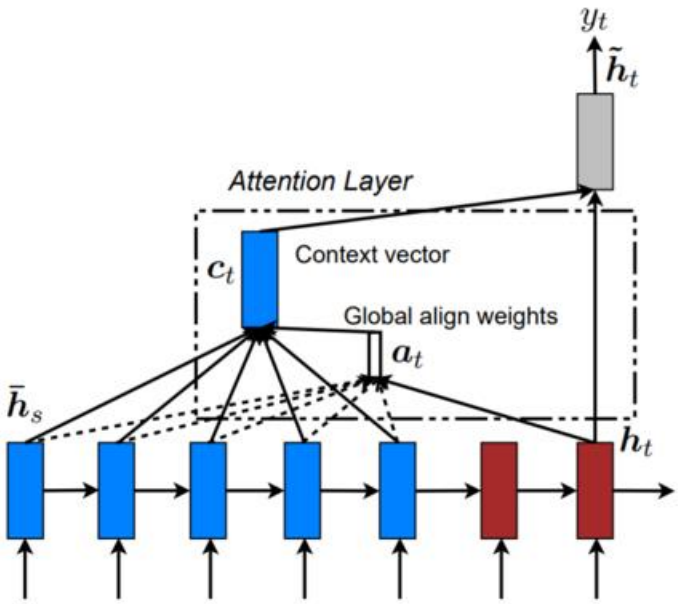
بردار *context* هم برای محاسبه خروجی نهایی دیکودر استفاده میشه.بردار *context* به نام *c(i)* مجموع وزن های توجه ضربدر حالت های مخفی است یعنی:

$$c_i = \sum_{j=1}^{Tx} a_{ij} h_j$$

در نهایت برای پیش بینی کلمه هدف،

$$S_i = f(S_{i-1}, c_i, y_{i-1})$$

حال توضیحی در مورد مکانیزم توجه *luong* ارائه میدهیم.اسم دیگر این مکانیزم *multiplicative* است.در این روش با ضرب ماتریسی، با سرعت بالاتر و حافظه بهینه تر امتیاز های توجه را محاسبه میکند. در مکانیزم *luong* دو نوع مکانیزم توجه را مبنی بر اینکه به کجای جمله ورودی توجه بشود، تعیین شد: ۱- توجه عمومی(*general attention*) و ۲- توجه محلی(*local attention*) که صورت سوال مرتبط با توجه عمومی است.در مکانیزم توجه عمومی سه نوع امتیازدهی مختلف داریم که در قسمت الف آن ها را بررسی کردیم. در شکل زیر به معماری *luong* میپردازیم:



تفاوت های *bahdau* و *luong* :

۱- در مکانیزم *bahdanau* از اتصال(*concatenate*) حالت های مخفی رو به جلو و عقب در یک انکودر دو طرفه و حالت مخفی هدف قبلی در یک دیکودر یک طرفه استفاده کردیم. اما روش *luong* از حالت های مخفی در بالای لایه *LSTM* در انکودر و دیکودر استفاده میکند.

۲- روش *luong* از حالت فعلی دیکودر برای محاسبه بردار ترازبندی استفاده میکند در صورتی که *bahdanau* از خروجی مرحله زمانی قبلی استفاده میکرد.

۳- روش *bahdanau* فقط از امتیازدهی به روش *concat* استفاده کرد اما *luong* از *dot* و *general* و *concat* استفاده کرد.

۴- روش *luong* سریع تر و کم حافظه تر است.

۵- معماری این دو روش متفاوت است.

لذا روش *luong* در مدل سازی از روش *bahdanau* بهتر است زیرا امتیازدهی های متنوعی را ارائه کرده، از لحاظ محاسباتی سبک تر است و حافظه کمتری نیز استفاده میکند.لذا میتوانیم وابستگی های بلندمدت را بهتر پوشش دهیم.

قسمت ج) همان طور که از بخش های قبلی یاد داریم، ***weight regularization*** زمانی استفاده میشد که میخواستیم مدل توجه کمتری به ورودی های کم اهمیت بکند و خطای ***generalization*** را از این طریق کاهش دهیم.حال در این کاربرد خاص، در ***encoder-decoder*** ها وقتی ورودی سمت ***encoder*** بلند باشد، اگر از روش ***global attention*** بخواهیم استفاده کنیم، چون کل ورودی باید پردازش شود، اینکار هزینه بر و شاید حتی غیر ممکن شود.حال برای حل این مشکل روش ***local attention*** پیشنهاد شد که فقط روی موقعیت های خاصی از ورودی تمرکز میکند(به ازای هر کلمه هدف). برای اینکه نخواهیم از ***local attention*** استفاده کنیم، میتوانیم مانند رویکرد ***regularization*** در وزن های ترازبندی(***alignment weights***) بیاییم از یک عبارت پنالتی استفاده کنیم و بتوانیم با اینکار ارزش قسمت هایی از ورودی را کم کرده و باعث شویم مدل کمتر به آن ها توجه کند. یعنی به تابع هزینه یک ترم پنالتی اضافه میکنیم تا با اینکار وزن های کم که کلمات ورودی کم اهمیت را نشان میدهند، به سمت صفر میل کنند و عملا توجهی به آن ها نشود.(مانند رویکرد ***L1 regularization*** که برای ***feature selection*** استفاده میشد).

قسمت د)بر اساس اینکه مکانیزم توجه آیا به کل ورودی دسترسی دارد یا به یک ***batch*** از ورودی،توجه به دو نوع نرم و سخت تقسیم بندی میشود.مکانیزم نرم یعنی وزن های ترازبندی روی تمام ورودی یاد گرفته میشود.یعنی دسترسی اش به کل ورودی است.مزیت این روش: مدل ***smooth*** و ***differentiable*** است. معایب: وقتی ورودی بزرگ باشد، هزینه بر است. مکانیزم توجه سخت یعنی در هر زمان فقط روی یک ***batch*** از ورودی یادگیری انجام شود یعنی دسترسی به کل ورودی وجود ندارد. مزایا: محاسبات کمتر و سریع تر معایب: مدل ***non-differentiable*** است و برای آموزش به تکنیک های کاهش واریانس یا یادگیری تقویتی نیاز است.

مسئله ۳-

قسمت الف) توجه شود که ***auto-encoder*** شکل دیگری از همان ورودی را در خروجی پیش بینی و تولید میکند برای مثال عمل فشرده سازی تصویر را در ابعاد کوچک تر انجام میدهد. اما ***skip-gram*** کلمه ای را ورودی گرفته و کلمات ***context*** آن را تولید میکند اینطور که احتمالی بین صفر و یک به هر کلمه دیکشنری میدهد و کلماتی که احتمال ***max*** را دارند به عنوان کلمات ***context*** آن معرفی میشوند.حال اگر بخواهیم بوسیله ***skip-gram*** خود آن کلمه را مانند ***auto-encoder*** تولید کنیم، مشکلی که هست این است که یک کلمه نمیتواند در ***context*** خودش باشد طبق ساختار ***skip-gram*** و ربطی به نحوه آموزش آن ندارد. لذا این عمل با شکست مواجه میشود.از طرف دیگر خروجی ***skip-gram*** یک بردار حاوی احتمالات است و عملا خروجی نمیتواند یک کلمه باشد مگر اینکه یک لایه اضافه کرده و کلمه متناظر با احتمال ***maximum*** را به عنوان خروجی تولید کند که خب این ساختار دیگر ***skip-gram*** نیست بلکه یک ساختار جدید است(با فرض اینکه مشکلی که گفته شد وجود نداشته باشد.

قسمت ب)ما در *skip-gram* یک لایه ورودی داریم که یک بردار *one-hot* به طول سایز دیکشنری کلمات ورودی گرفته و به وسیله ماتریس اول که بین لایه ورودی و لایه مخفی است، *embedding* کلمه ورودی را ایجاد میکند.این ماتریس هر ستون اش *pre-activation* واحد های مخفی در لایه مخفی است.توجه شود که ما نیازی به هیچ *activation* ای روی واحد های مخفی در لایه مخفی نداریم و باید مقادیر مستقیما به لایه آخر شبکه بروند.اما توجه شود که *embedding* ورودی کافی نیست و معنی ای ندارند و باید خروجی شبکه یک بردار *one-hot* دیگر که متفاوت از بردار *one-hot* ورودی است باشد که کلمات *context* ورودی را مشخص کند.ماتریس دوم بین لایه مخفی و لایه خروجی به این دلیل کاربرد دارد که *embedding* را از لایه مخفی گرفته و بردار *one-hot* جدید را برایمان تولید کند.در لایه خروجی از *softmax* استفاده میکنیم تا کلمات مرتبط *context* ورودی تولید شوند(از بردار *one-hot* در لایه خروجی). ما به لایه آخر و ماتریس دوم نیاز داریم زیرا نمیتوانیم *softmax* را مستقیما بعد از لایه مخفی بگذاریم چون نیاز داریم ابتدا یک *embedding* از ورودی ایجاد شود و سپس از آن *embedding* استفاده کنیم تا بردار *one-hot* جدید ایجاد کنیم.لذا ماتریس دومی و لایه خروجی ای نیز نیاز است.

قسمت ج) اگر یک کلمه را در word2vec به عنوان کلمه center بهش نگاه کنیم، با همان کلمه وقتی به عنوان context بهش نگاه میکنیم متفاوت است.توجه شود که context یک کلمه نمیتواند همان کلمه باشد.یعنی ماتریس مرتبط با کلمه center هه یک ماتریس با بردار متناظر با کلمه context هه آن کلمه متفاوت است.اگر این دو ماتریس اشتراک داشته باشند، چون یک کلمه در context خودش نباید بیاید، مدل باید P(word|word) را کمینه کند اما بخاطر اشتراک در ماتریسی که گفته شد، توانایی اینکار از بین میرود چون ضرب داخلی این دو ماتریس صفر نیست دیگر.

مسئله ۴ – گزارش این سوال در نوتبوک مربوطه آورده شده است.

مسئله ۵ – گزارش این سوال در نوتبوک مربوطه آورده شده است.

مسئله ۶ – گزارش این سوال در نوتبوک مربوطه آورده شده است.