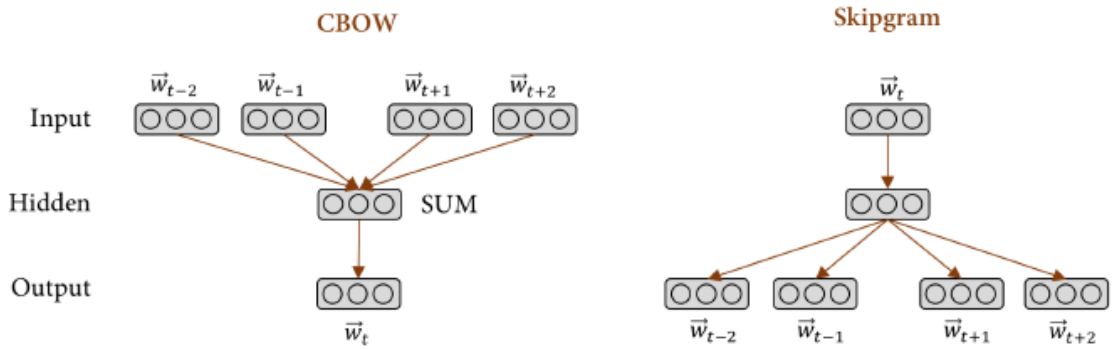
-1

در پردازش زبان‌های طبیعی word embedding به کار گرفته می‌شود تا بازنمایی مناسبی برای کلمات به دست آوریم به طوری که وابستگی و روابط کلمات به خوبی نشان داده شود. برای اینکه بازنمایی مناسب را به دست بیاوریم همیشه داده برچسب خورده نداریم که supervised training انجام دهیم. در چنین مواقعی سعی می‌کنیم خودمان یک task تعریف کنیم تا مدل از روی آن embedding یا بازنمایی مناسب را به دست آورد. CBOW و skip-gram دو روش word embedding برای انجام tokenization هستند. در این روش‌ها هدف یادگیری وزن‌های لایه‌های hidden هست که در واقع همان بردار کلمات هست که می‌خواهیم یادشان بگیریم.

CBOW آموزش دیده است تا یک کلمه را از یک پنجره ثابت کلمات زمینه پیش‌بینی کند ، در حالی که Skip-gram برعکس عمل می‌کند و سعی می‌کند چندین کلمه زمینه مجاور را از یک کلمه ورودی واحد پیش بینی کند. در شکل زیر معماری این دو روش را می‌بینید.



Skip-gram: با داده‌های آموزشی کم به خوبی کار می‌کند ، حتی برای کلمات یا عبارات کمیاب بازنمایی مناسبی پیدا می‌کند.

CBOW: آموزش آن چندین برابر سریعتر از Skip-gram است، دقت برای کلمات متداولتر کمی بهتر است.

CBOW روابط نحوی بهتری بین کلمات یاد می‌گیرد در حالی که Skip-gram در گرفتن روابط معنایی بهتر است. این به این معنی است که برای کلمه "cat" CBOW به عنوان نزدیکترین بردار از نظر ریخت شناسی کلمات مشابه مانند جمع ، یعنی " cats" را بازیابی می‌کند در حالی که Skip-gram کلمات از نظر ریخت‌شناسی متفاوت (اما از نظر معنایی مربوط است) مانند "dog" که به "cat" نزدیک است.

From word to sense embeddings: A survey on vector representations of meaning

NLP 101: Word2Vec — Skip-gram and CBOW

2-

الف)

$L(A, P, N) = \max(d(A, P) - d(A, N) + \alpha , 0) =$

$\max(0.4- 0.5 + 0.2 , 0) = 0.1$

ب)

$d(A, P) = \|e_A - e_P\|^2$

$d(A, N) = \|e_A - e_N\|^2$

در مثال فوق $d(A, P) - d(A, N) + \alpha > 0$ بنابراین برای تابع ضرر داریم:

$L(A, P, N) = d(A, P) - d(A, N) + \alpha = \|e_A - e_P\|^2 - \|e_A - e_N\|^2 + \alpha$

$g_A = 2(e_A - e_P) - 2(e_A - e_N) = 2(e_N - e_P)$

$g_P = 2(e_A - e_P)\ (-1) = 2(e_P - e_A)$

$g_N = - 2(e_A - e_N)\ (-1) = 2(e_A - e_N)$

پ)

$L(A, P, N) = \max(d(A, P) - d(A, N) + \alpha , 0) =$

$\max(0.4- 1.5 + 0.2 , 0) = 0$

چون $d(A, P) - d(A, N) + \alpha < 0$ بنابراین برای تابع ضرر داریم:

$L(A, P, N) = 0$

بنابراین مشتق آن نسبت به تک تک عضوهای هر embedding برابر صفر است یعنی:

g_A = [0 ... 0] <---- برداری 256 بعدی که تمام عضوهای آن برابر 0 هستند

g_P = [0 ... 0] <---- برداری 256 بعدی که تمام عضوهای آن برابر 0 هستند

g_N = [0 ... 0] <---- برداری 256 بعدی که تمام عضوهای آن برابر 0 هستند

-3

مجموعه داده :

مجموعه دادهی MS COCO (Microsoft Common Objects in Context) یک مجموعه داده تشخیص اشیاء (object detection) ، تقسیم بندی (segmentation) ، تشخیص نقطهی کلیدی (key-point detection) و کپشن (captioning) با مقیاس بزرگ است. مجموعه داده شامل 328K تصویر است.

تقسیم‌بندی مجموعه داده: اولین نسخه از مجموعه دادههای MS COCO در سال 2014 منتشر شد. این مجموعه شامل 164K تصویر تقسیم شده به مجموعه‌های آموزش (83K) ، اعتبار (41K) و تست (41K)است. در سال 2015 مجموعه تست دیگری از تصاویر 81K شامل همه تصاویر تست قبلی و 40K تصاویر جدید منتشر شد. براساس بازخورد جامعه ، در سال 2017 تقسیم آموزش / اعتبار از 41K / 83K به 5K / 118K تغییر یافت. تقسیم جدید از همان تصاویر و انوتیشن‌ها (متن) استفاده می کند. مجموعه تست 2017 زیرمجموعه‌ای از شامل 41K تصاویر از مجموعه تست 2015 است. علاوه بر این ، نسخه 2017 شامل مجموعه داده جدیدی است که حاوی K123 تصویر بدون متن است.

1.5 میلیون نمونه شی

80 دسته شی

5 کپشن برای هر تصویر

What is COCO?

COCO (Microsoft Common Objects in Context)

پیش‌پردازش:

پیش‌پردازش شامل بخش است: پیش‌پردازش روی تصاویر و پیش‌پردازش روی کپشن‌ها

پیش‌پردازش روی تصاویر:

تصاویر به تنسورهای سه بعدی تبدیل شده و سپس سایز همه‌ی آن‌ها را به (229,229) تغییر داده و همه را scale کردیم تا مقدار پیکسل‌ها بین 1- و 1 قرار بگیرد.

پیش‌پردازش روی کپشن‌ها:

5000 کلمه‌ی پرتکرار را انتخاب کرده تا Tokenization را روی آنها انجام داده و بقیه کلمات out of vocabulary words در نظرگرفته شده و به تمام برچسب <unk> به معنای unknown می‌زنیم. در ضمن بسیاری از علائم نگارشی (punctuation) از جمله !"#$%&()*+,-./:;=?@[\]^_`{|}~ را از متون حذف کردیم.

استخراج ویژگی (استفاده از inception-v3):

برای استخراج ویژگی از تصاویر را به یک مدل InceptionV3 که وزن‌های آن روی مجموعه داده‌ی ImageNet آموزش داده شده‌اند و از آخرین لایه کانولوشنی خروجی می‌گیریم که این خروجی‌ها همان ویژگی‌های استخراج شده هستند.

Tokenization

بهرای کلماتی که به صورت توکن به توکن (مثلا کلمه به کلمه) در آمده‌اند اندیس‌هایی از دیکشنری لغات نسبت داده و دو نگاشت یکی از کلمه به اندیس و دیگری برعکس تولید کرده و دنباله حاصل برای هر جمله را به اندازه بزرگترین دنباله pad می‌کنیم (post padding).یعنی به اندازه تفاضل با بزرگترین دنباله در انتها صفر می‌گذاریم.

مدل و معماری شبکه

مدل پیشنهادی شامل یک مدل CNN Encoder برای تصاویر و یک مدل RNN Decoder برای کپشن‌ها است.

CNN Encoder: این مدل ویژگی‌های استخراج شده از Inception3 را به عنوان ورودی گرفته و فقط یک لایه fully connected و تابع فعالسازی Relu روی آن‌ها اعمال می‌کند. خروجی آن ویژگی‌هایی هستند که به RNN Decoder داده می‌شوند.

RNN Decoder: این مدل کپشن‌های tokenize شده را به همراه ویژگی‌های استخراج شده از CNN Encoder را به عنوان ورودی می‌گیرد و روی ویژگی‌های استخراج شده برای تصاویر حرکت می‌کند تا

بخش‌هایی از آن که برای پیشبینی کپشن متناسب است را بیابد. همچنین در RNN Decoder برای پیاده‌سازی attention روی تصویر استفاده شده است.

بررسی مکانیزم attention

مکانیزم attention در این task برای تشخیص و توجه بیشتر به قسمت‌هایی از تصویر که در ایجاد کپشن موثرتر هستند استفاده شده است. این مکانیزم attention (BahdanauAttention) باعث می‌شود تا برای ایجاد کپشن به بخشی از تصویر که الان decoder برای آن کپشن می‌زند و نواحی اطرافش توجه و اهیت بیشتری نسبت به نواحی دیگر داده شود. درباره این مکانیزم در مقاله‌ی زیر که مربوط به هموروک 7 و task ترجمه‌ی ماشینی عصبی بود توضیح داده شد.

بردار زمینه‌ی c_i به دنباله‌ای از annotationهای ($h_1, ..., h_{Tx}$) بستگی دارد که انکود تصاویر ورودی را به آن‌ها نگاشت می‌کند. هر انوتیشن h_i شامل اطلاعاتی در مورد کل دنباله‌ی ورودی است اما تمرکز بیشتری بر قسمت‌هایی که اطراف ویژگی i ام دنباله‌ی ویژگی‌های تصاویر هستند دارد. بردار زمینه به صورت جمع وزن‌دار این انوتیشن‌ها محاسبه می‌شود.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

وزن α_{ij} برای هر انوتیشن h_j به صورت زیر محاسبه می‌شود:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

که $e_{ij} = a(s_{i-1}, h_j)$

یک مدل تراز بندی است که score می‌دهد ورودی های اطراف موقعیت j و خروجی در موقعیت i چقدر خوب تطبیق می‌یابند. این score بر اساس حالت مخفی s_{i-1} در RNN و j امین انوتیشن h_j روی دنباله‌ی ورودی است.

ما مدل a را به عنوان یک شبکه‌ی عصبی feed-forward که همراه با دیگر مولفه‌های سیستم آموزش می‌یابد پارامتر می‌کنیم.

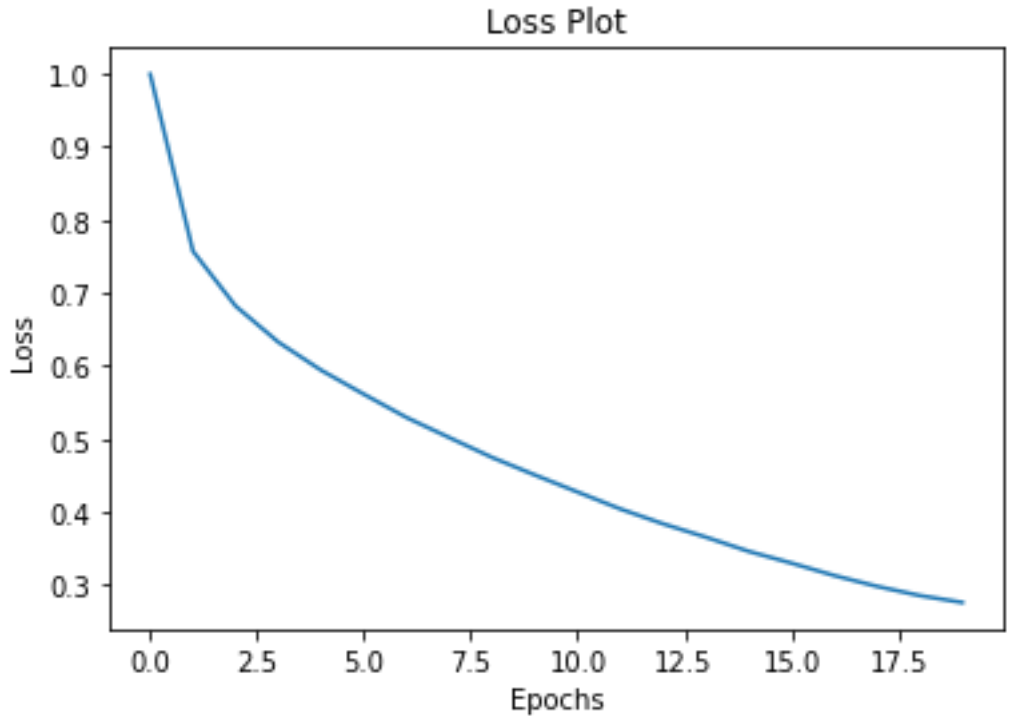
می‌توان عمل جمع وزن‌دار روی انوتیشن‌ها را به صورت متوسط انوتیشن‌ها تفسیر کرد. که این متوسط‌گیری روی ترازهای ممکن صورت می‌گیرد.

این یک مکانیزم توجه را در دیکودر پیاده‌سازی می‌کند. دیکودر تصمیم می‌گیرد به چه قسمت‌هایی از تصویر مبدا توجه کند.

[Neural machine translation by jointly learning to align and translate]

نتایج

خطا در این مدل در انتهای ایپاک اول مقدار 1 را داشته و در نهایت پس از 20 ایپاک به کمتر از 0.3 رسیده است. خطای مدل به نظر به اندازه کافی کم نیست و یا حداقل به ایپاکهای بیشتری احتیاج دارد.



```
Real Caption: <start> a large red bus on a city street <end>
Prediction Caption: a red double decker bus driving on a building <end>
```