وسط جلسه 17 از دقیقه 55:

Bag of word: در این شیوه مجموعه کلمات متن به صورت برداری خواهد بود یعنی به هر کلمه یک بردار نسبت بدهیم و feature بشود جمع این بردار ها و کارایی ندارد چون ترتیب دنبال متن را حفظ نمیکند.

One-hot: مشکلش این است که ابعاد آن بالا هست به ازای هر کلمه یک بردار به اندازه دیکشنری خودمان داریم که در هر بردار فقط یکی 1 هست (کلمه متناظر با آن) و بقیه 0 هستند. sparse هستند یعنی یک مقدار زیادی صفر دارند.

یک روش دیگر این است که بیایم به ازای هر کلمه یک featureهایی در نظر بگیریم و برداری که برای آن کلمه در نظر میگیریم میتواند مفید باشد و بردار ها مفهوم میگیرند و اختلاف مرد با زن مانند اختلاف پادشاه با ملکه هست. به این روش word embedding گفته میشود.

Word embedding: dense هستند یعنی ابعاد کوچولو هست ولی information دارند و قابل یادگیری هستند این feature هایی بردار ها در این روش و relation بین کلمات میتواند یاد گرفته شود مثلا پرتغال دید احتمالا بعدش آب پرتغال هست نه فیل. دقت کن این بردار ها قابل یادگیری هستند و در کنار کار اصلی که مثلا classification روی متن ها هست صورت میگیرد در کنار آن word embedding صورت بگیرد برای بدست آمدن بردار ها. یک روش دیگر استفاده از مدل های قبلی هست که قبلا آموزش داده شده اند. یعنی بردار نسبت به کلمات اتلاق میشود.

برای بدست آوردن این بردار ها مثلا میتوانیم یک پنجره در نظر بگیریم که هر خانه یک کلمه است و رابطه بین هر خانه را بدست میاوریم یک پنجره لغزان است که دارد حرکت میکند و 2 به 2 ارتباط بین کلمات را بدست میاوریم. خروجی شبکه به اندازه بردار ورودی هست و از softmax classifier استفاده شده است به این معنی که خروجی یکی همیشه 1 خواهد بود. نقش لایه وسطی چی خواهد بود؟ بر طبق پنجره هر سری یکی از لایه خروجی دارد فعال میشود پس ما یک متن بزرگ بدهیم انگار این لایه وسطی دارد یادگیری انجام میدهد و در نهایت یک feature vector dense به ما میدهد. یعنی اکثر بردار ما غیر صفر است و حاوی اطلاعات است. این لایه مخفی عین look up است یعنی ما 10 هزار تا کلمه داریم به ازای هر کلمه 300 تا feature داریم یا نورون که هر کدام یک بردار میشود و بعد اون روش one hot را که یک بردار هست در این ماتریس ضرب میکنیم و خروجی متناظر با one hot میاد و dense آن را میدهد و اطلاعات راجب اون کلمه هست و ارتباط بین کلمات را در خودش نگه میدارد به جای اینکه sparse باشد و کلی مقدار 0 داشته باشیم.

مشکل اینجا هست که اگر بر اساس کلمات فقط embedding کنیم بعد میانگین مثلا بگیریم بین feature ها و بعد classify کنیم شاید زیاد دقیق نباشد که میانگین بگیریم از feature vector. چون هیچ relation لحاظ نکردیم و صرفا داریم میانگین میگیریم.

Sigmoid میزنیم معمولا binary classification است. یک تعداد لغات پر کاربرد داریم یک تعداد بردار داریم و یک عددی هم داریم برای اینکه مثلا بدانیم چند تا کلمه اول را در نظر بگیریم.

تعداد پارامتر embedding؟ تعداد کلمات پر کاربرد ضربدر مقداری که برای هر بردار در نظر گرفتیم مثلا گفتیم بردار هایی 8 تایی داشته باشیم یعنی embedding ما چند تایی باشد.

مدل Stanford بیش بردازش شده است مشکل اینجا هست که ما یک سری بردار داریم برای هر کلمه و به عنوان ورودی به شبکه میدهیم و ارتباطی بین آن نیست. یکی از روش ها این است که برای هر کلمه یک con 1بعدی است برای اینکه relation زمانی یا ارتباط آنها را بدست بیاوریم حالا مثلا در اینجا روی یک بعد و فقط روی زمان حرکت بکند به جای 2 بعدی که عمق ثابت بود، یعنی فقط روی یک محور حرکت میکند.

VGG مقدار feature را جستجو کن.

عمق conv 1 بعدی به اندازه هم عمق بردار است مثلا 512. دقت کن وقتی فیلتر میزنی عدد اسکالر میشود وقتی روی یک بعد میزنی در نتیجه خروجی میشود 40 در 1 یا یک بردار 40 تایی. اینجا relation 3 تایی زدیم با کانولوشن یک بعدی بخاطر اون فیلتر و پیچیدگی زیاد ندارد و با این فیلتر در واقع ارتباط اون 3 تا را بدست آوردیم یعنی relation زمانی بدست آوردیم و دقت کن بعد از این کانولوشن یک relation دیگه میتونی بزنی با کانولوشن یک بعدی میتوانیم الگو ها و ارتباطات بیشتری را بدست بیاوریم. دقت کن فرقی نمیکند 40 تا کلمه باشد یا 80 تا میخواهیم یک feature هم سایز بدست بیاوریم و مسئله این است.

دقت کن feature vector یکتا باید باشد و global باشد و دیگر سایز متن فرقی نمیکند 40 تا کلمه باشد یا 80 تا.

شبکه های عصبی feedforward: در این شبکه ها قابلیت پردازش دنباله ها با طول های متفاوت نداشتیم و ترتیبی در پردازش نداشتیم و هر ورودی به صورت مستقل بدون در نظر گرفتن سایر ورودی ها محاسبه میشد و حالا نیاز داریم relation بین اینها داشته باشیم یعنی خروجی این لحظه در ورودی لحظه بعدی تاثیر گذار خواهد بود. H حالت سیستم هست که با خروجی یکی هست. و یک feedbackیی داریم.

شبکه های RNN:

حالا این feedback در این شبکه ها بدرد میخورد. ماتریس U وزن های اون خروجی هستند در لحظه قبل. یعنی خروجی اولی به همراه ورودی لحظه بعدی خروجی بعدی بدست و این خروجی به همراه ورودی لحظه بعدی خروجی بعدی بدست میاد و این روش برای پردازش سری های زمانی و دنباله ها بسیار مناسب هستند و relation ها را میتوانیم در نظر بگیریم. انتخاب خروجی هم 2 گزینه داریم: 1. میتوانیم دنباله همه خروجی را در نظر بگیریم یا صرفا فقط آخرین خروجی را در نظر بگیریم.

One to one: یک ورودی داریم یک feedbackیی هم میگیریم به عنوان state و خروجی محاسبه میشود.

One to many: یک ورودی بهش میدهیم و خروجی ها در لحظه های بعدی وارد میشوند به عنوان ورودی در لحظه بعدی تاثیر گذار هستند یعنی یک ورودی داریم و یک دنباله بدست میاریم.

Many to one: یک سری ورودی میدهیم و فقط یک خروجی میگیریم.

Many to many: یک جمله به یک زبانی بگیرد و ترجمه کند به زبان دیگر حالا میتواند طول ورودی با خروجی برابر باشد یا برابر نباشد اگر برابر نبود یک ورودی به عنوان encoder میسازیم feature بدست میاد بعد به decoder میدهیم.

Many to one مثل همین review رستوران. One to many مثل تولید کننده شعر یا image captioning.