

بسم الله الرحمن الرحيم



دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

بررسی روش‌های تولید متن و ایجاد مدل مولد متون فارسی

گزارش پروژه کارشناسی

مهدی علی‌خاصی

۹۵۳۰۴۸۳

استاد پروژه

دکتر مهران صفایانی

۱۳۹۹

فهرست مطالب

صفحه	عنوان
چهار	فهرست مطالب
شش	فهرست تصاویر
هفت	فهرست جداول
۱	چکیده
۲	فصل اول : مقدمه
۴	فصل دوم : پیش زمینه
۴	۱-۲ مقدمه ای بر شبکه های عصبی عمیق
۷	۲-۲ word embedding
۹	۳-۲ شبکه های عصبی بازگشتی
۱۱	۴-۲ Long Short-Term Memory
۱۴	۵-۲ مدل خود رمزگذار متغیر
۱۷	۶-۲ شبکه های مولد خصمانه
۱۸	۷-۲ پیش پردازش زبان های طبیعی
۱۹	فصل سوم : معرفی داده های مسأله
۲۰	۱-۳ اشعار فردوسی
۲۱	فصل چهارم : حل مسئله
۲۱	۱-۴ مدل شبکه عصبی بازگشتی
۲۳	۱-۱-۴ توابع خطا و دقت
۲۴	۲-۱-۴ پیاده سازی
۲۵	۳-۱-۴ نمونه گیری از مدل
۲۶	۴-۱-۴ نمونه خروجی در حین یادگیری
۲۶	۵-۱-۴ نمونه خروجی پس از یادگیری
۲۸	۶-۱-۴ بازسازی متن
۲۹	۲-۴ مدل خود رمزگذار متغیر
۳۰	۱-۲-۴ تابع خطا

۳۱	پیاده‌سازی ۲-۲-۴
۳۲	نمونه‌گیری از مدل ۳-۲-۴
۳۳	بازسازی متن ۴-۲-۴
۳۶	مدل‌های تولیدکننده‌ی خصمانه ۳-۴
۳۸	پیاده‌سازی ۱-۳-۴
۴۰	نمونه‌گیری از مدل ۲-۳-۴
۴۲	فصل پنجم: نتیجه‌گیری
۴۵	فصل ششم: کارهای آینده
۴۷	مراجع

فهرست تصاویر

۶	۱-۲ نحوه محاسبه مقدار هر گره [۱۷]
۹	۲-۲ ساختار شبکه‌های عصبی بازگشتی [۱۶]
۱۱	۳-۲ ساختار شبکه‌های عصبی بازگشتی [۲]
۱۱	۴-۲ ساختار LSTM [۱۸]
۱۲	۵-۲ دروازه‌ی فراموشی [۱۸]
۱۲	۶-۲ دروازه‌ی ورودی [۱۸]
۱۳	۷-۲ Cell State [۱۸]
۱۳	۸-۲ دروازه‌ی خروجی [۱۸]
۱۴	۹-۲ LSTM دوطرفه [۱۸]
۱۵	۱۰-۲ LSTM پشته شده [۱۸]
۱۶	۱۱-۲ ساختار مدل‌های خود رمزگذار [۱۵]
۱۸	۱۲-۲ ساختار مدل‌های مولد خصمانه [۱۲]
۲۳	۱-۴ ساختار مدل مطرح شده
۲۵	۲-۴ تفاوت یادگیری بازور و یادگیری معمولی [۲۱]
۲۵	۳-۴ ضابطه تابع ELU [۱]
۲۷	۵-۴ ساختار مدل شبکه‌ی عصبی بازگشتی مورد استفاده
۲۹	۶-۴ ساختار مدل‌های خود رمزگذار متغیر [۵]
۳۳	۷-۴ ساختار مدل خود رمزگذار متغیر مورد استفاده
۳۸	۸-۴ استفاده از خود رمزگذار در مدل‌های مولد خصمانه [۷]
۳۹	۹-۴ ساختار مدل خود رمزگذار
۴۰	۱۰-۴ ساختار مدل تولیدکننده
۴۱	۱۱-۴ ساختار مدل تمیزدهنده

فهرست جداول

۱-۲	انواع توابع فعال‌ساز [۲]	۷
۲-۲	کاربردهای شبکه‌های عصبی بازگشتی [۲]	۱۰
۱-۴	خروجی مدل شبکه عصبی بازگشتی در حین یادگیری	۲۸
۲-۴	خروجی مدل شبکه عصبی بازگشتی پس از یادگیری	۲۸
۳-۴	متن‌های ورودی پس از حذف کلمات از بیت	۲۹
۴-۴	خروجی مدل پس از بازسازی متن	۲۹
۵-۴	ورودی و خروجی مدل در دوره‌های مختلف	۳۲
۶-۴	خروجی مدل خودرمزگذار متغیر پس از یادگیری	۳۴
۷-۴	متن‌های ورودی پس از حذف کلمات از بیت	۳۵
۸-۴	خروجی مدل پس از بازسازی متن	۳۵
۹-۴	متن‌های ورودی پس از حذف کلمات از بیت	۳۵
۱۰-۴	خروجی مدل پس از بازسازی متن	۳۶
۱۱-۴	ورودی و خروجی مدل در دوره‌های مختلف	۳۹
۱۲-۴	خروجی‌های ایجاد شده توسط مدل تولیدکننده	۴۱

چکیده

یکی از موارد بدون پاسخی که در سال‌های اخیر توجه بسیاری به خود جذب کرده و از طریق روش‌های مختلف و متنوع با استفاده از هوش مصنوعی (اعم از مدل‌های بازگشتی بر حسب زمان، مدل‌های خودرمنگارمتغیر^۱، مدل‌های تولیدکننده‌ی خصمانه^۲، مدل‌های خودرمننگار^۳، زنجیره‌ی مارکوف^۴ و ...) سعی بر پاسخ‌دهی به آن شده است، ایجاد مدل‌هایی است که قادر به تولید متن‌هایی شبیه به متن‌های واقعی در قالب‌های متنوع مانند دیالوگ‌های روزمره، شعر، متون ادبی و ... باشد. درواقع مدل‌های ارائه شده در سال‌های اخیر همگی همواره به دلایل مختلف اعم از عدم وجود معیار مناسب برای سنجش دقت، پیچیدگی ذاتی زبان و ... قادر به رسیدن به اهداف واقعی خود نبوده‌اند. در پروژه‌ی پیش‌رو سعی بر آن شده است که با تکیه بر تکنیک‌های مختلف پردازش زبان‌های طبیعی^۵ و هوش مصنوعی، در گام اول اقدام به بررسی مدل‌های پیشین کرده و عملکرد آن‌ها بر روی داده‌های مختلف و به طور ویژه فارسی بررسی گردد و سپس اقدام به ایجاد مدلی مناسب برای تولید متون فارسی گردد. در پایان شایان ذکر است که سیستم فوق می‌تواند در زمینه‌های متفاوتی اعم از تولید شعر، ایجاد سیستم چت بات هوشمند و ... مفید واقع گردد.

واژه‌های کلیدی: پردازش زبان‌های طبیعی، تولید متن، هوش مصنوعی، یادگیری عمیق

¹Variational Auto-Encoder

²Generative Adversarial Network

³Auto-Encoder

⁴Markov chain

⁵Natural language processing

فصل اول

مقدمه

تولید زبان طبیعی^۱ در واقع یکی از زیرمجموعه‌های پردازش زبان‌های طبیعی^۲ است. در این شاخه از پردازش زبان‌های طبیعی، هدف آن است که با استفاده از اطلاعات و دانشی که از زبان‌های طبیعی و ساختار آن موجود است و با ترکیب آن با هوش مصنوعی، مدل و نرم‌افزاری تولید شود که قادر به ایجاد متن‌های مرتبط با زبان مدنظر باشد. نکته‌ی مهم در رابطه با متن‌های تولید شده آن است که این متون باید حداقل‌های لازم برای برقراری ارتباط^۳ را ارضاء کنند و تا حد ممکن شبیه متن‌های واقعی باشند. چنین سیستمی می‌تواند در زمینه‌های مختلفی استفاده شود. برای مثال شرکت‌ها می‌توانند از آن برای تولید گزارش‌های خودکار استفاده کنند و یا از چنین سیستمی برای تولید محتوا برای وب سایت‌ها و وب‌اپلیکیشن‌ها نیز می‌تواند استفاده شود.

همچنین این سیستم می‌تواند کارایی منحصر به فردی در اپلیکیشن‌هایی که ارتباط مستقیم^۴ با کاربر دارند مانند ربات‌های چت^۵ داشته باشد و یا حتی در سیستم‌های مذکور به جای استفاده‌ی مستقیم، توسط یک سیستم تبدیل متن به صوت استفاده گردد.

¹natural-language generation

²natural language processing

³communication

⁴interactive applications

⁵chatbot

مدل‌های تولید متن را میتوان با پردازش‌های ذهنی انسان‌ها هنگامی که سعی در تبدیل یک ایده به یک متن دارند مقایسه کرد. از این رو مدل‌های تولید زبان طبیعی نیز با دریافت ورودی خاص (که درواقع معادل ایده اولیه در انسان‌ها است) اقدام به تولید متن می‌کنند.

البته لازم به ذکر است که در عمل، عوامل زیادی وجود داشته که فرآیند تولید زبان طبیعی را با چالش‌های خاص روبرو می‌کند. از این موارد میتوان ساختار بسیار پیچیده‌ی زبان‌های طبیعی، وجود مواردی مانند ابهام، ایهام، ضرب‌المثل و ... در زبان طبیعی نام برد.

همچنین فرآیند تولید زبان طبیعی را میتوان عملیاتی در مقابل درک زبان طبیعی^۱ دانست که در آن سعی بر آن است که از متنی خاص، ایده‌ها و هدف اصلی استخراج شود ولی در تولید زبان‌های طبیعی سعی شده‌است که بر اساس یک ایده‌ی اولیه، متن تولید شود.

مدل‌های تولید زبان را میتوان به مدل‌های قطعی^۲ و غیرقطعی^۳ نیز تقسیم‌بندی کرد. در این تقسیم‌بندی در مدل‌های غیرقطعی برخلاف مدل‌های قطعی، سعی شده‌است که خروجی علاوه بر حفظ ظاهر و ساختار خود، مقداری تصادفی بودن و عدم قطعیت داشته باشد به طوریکه با دریافت ورودی یکسان، همواره خروجی‌های یکسانی تولید نکند. برای رسیدن به این هدف، در اغلب موارد از توزیع‌های تصادفی^۴ و یا از نویز^۵ به عنوان عامل ایجاد عدم قطعیت استفاده می‌شود.

نهایتاً یکی از مهم‌ترین قسمت‌های مدل ایجاد شده، ارزیابی مدل است که در رابطه با روش‌های مختلف آن در فصل ۵ به تفصیل پرداخته خواهد شد.

^۱natural-language understanding

^۲deterministic

^۳non-deterministic

^۴random distributions

^۵noise

فصل دوم

پیش زمینه

در این فصل به بیان پایه‌های پردازش زبان طبیعی و همچنین شبکه‌های عصبی عمیق^۱ که در این پروژه استفاده شده است، می‌پردازیم.

۱-۲ مقدمه‌ای بر شبکه‌های عصبی عمیق

یادگیری عمیق^۲ زیرشاخه‌ای از علم یادگیری ماشین^۳ است که در آن با تکیه بر شبکه‌های عصبی^۴، به کامپیوتر این قابلیت را می‌دهد که بدون استفاده از هیچ برنامه‌نویسی آشکاری بتواند یاد بگیرد (بدون مشخص کردن کامل تمامی شرایط در غالب جملات شرطی، کامپیوتر قادر باشد در شرایط مختلف بر اساس مدلی که قبلاً از داده‌ها یادگرفته است، اقدام به تصمیم‌گیری کند)

در مدل‌های یادگیری عمیق، یادگیری می‌تواند به صورت نظارت‌شده^۵ یا بدون نظارت^۶ باشد. در مدل‌های نظارت‌شده، کامپیوتر سعی می‌کند که یک مدل ریاضی بر اساس داده‌های ورودی و خروجی ایجاد کند و در این

¹deep neural networks

²deep learning

³machine learning

⁴neural networks

⁵supervised

⁶unsupervised

راستا، خروجی‌ها از قبل تفکیک شده و برچسب زده شده^۱ هستند. در این فرایند که به فرایند یادگیری^۲ شناخته می‌شود، کامپیوتر سعی می‌کند با استفاده از داده‌های ورودی که به آن داده شده است، مدلی ریاضی ایجاد کند تا با استفاده از مدل به دست آمده و ورودی‌ها، به خروجی‌های مدنظر که آن‌ها نیز به کامپیوتر داده شده است برسد. در حین این فرایند تابعی نیز به عنوان تابع هدف و یا تابع خطا^۳ در نظر گرفته شده است و دقت عملکرد مدل به دست آمده، بر اساس تابع هدف یا تابع خطا محاسبه می‌شود (و کامپیوتر سعی بر آن می‌کند که به صورت مکرر، اقدام به بهینه‌سازی تابع هدف خود یا کاهش مقدار تابع خطا کند). بدین ترتیب امید است که در آینده هنگامی که کامپیوتر با داده‌های جدیدی روبرو می‌شود، با توجه به کمینه بودن تابع خطای مدل، کامپیوتر بتواند با استفاده از مدلی که قبلاً یادگرفته است اقدام به ایجاد خروجی‌های مناسب کند.

در این فرایند به داده‌هایی که مدل بر اساس آن‌ها یاد می‌گیرد داده‌های آموزش^۴ و به داده‌هایی که مدل بر اساس آن آزمایش می‌شود داده‌های آزمایش^۵ می‌گویند. الگوریتم‌های نظارت‌شده نیز به خودی خود می‌توانند به دو دسته‌ی اصلی طبقه‌بندی^۶ و رگرسیون^۷ تقسیم‌بندی شوند. الگوریتم‌های طبقه‌بندی هنگامی استفاده می‌شود که خروجی‌ها مقدارهای مشخص دسته‌بندی شده دارند و به صورت گسسته هستند. برای مثال الگوریتمی را فرض کنید که با مشاهده‌ی یک تصویر، اقدام به قرار دادن تصویر در سه دسته‌بندی مجزای سگ، گربه، درخت کند. در این الگوریتم ورودی به شکل یک تصویر بوده که نهایتاً بر اساس مدل به دست آمده، خروجی به شکل یکی از سه دسته‌ی فوق خواهد بود. البته لازم به ذکر است که در اکثر موارد، مدل به جای خروجی دادن تنها یک دسته، سعی بر خروجی دادن یک مدل احتمالاتی می‌کند بدین شکل که احتمال قرار گرفتن تصویر در هر کدام یک از دسته‌بندی‌های مشخص شده را خروجی می‌دهد. درواقع به چنین مدل‌هایی میتوان به شکل یک تابع احتمالاتی شرطی نگاه کرد که به ازای هر ورودی، یک تابع احتمالاتی خاص را خروجی می‌دهند. از دیگر مسائل دسته‌بندی میتوان به مسأله‌ی تشخیص ایمیل‌های هرز^۸ اشاره کرد که در آن ایمیل‌ها بر اساس محتوا به دو دسته‌ی هرزنامه و هرز نبودن دسته‌بندی می‌شوند. در این پروژه به دلیل ساختار کلمات و هدف پروژه که تولید یک رشته متن است، از مدل‌های نظارت‌شده و دسته‌بندی استفاده می‌شود که ساختار مدل را در ادامه خواهیم دید.

در مدل‌های بدون نظارت، کامپیوتر سعی می‌کند با پیدا کردن تشابه‌های موجود بین داده‌ها، داده‌ها را در

¹labeled

²training

³loss function

⁴training set

⁵test set

⁶classification

⁷regression

⁸spam

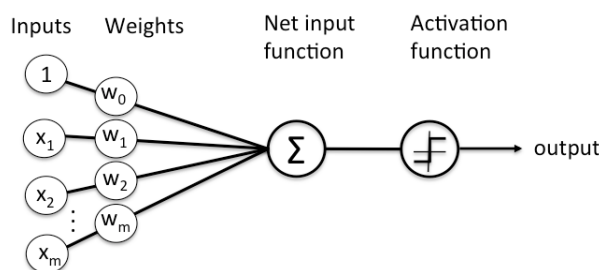
گروه‌های مختلفی دسته‌بندی کرده به طوری که داده‌های هم گروه دارای بیشترین تشابه به یکدیگر باشند و مدل تلاش می‌کند که داده‌های مشابه به یکدیگر را تشخیص دهد.

یادگیری عمیق خود دارای ساختارهای متفاوتی مانند deep neural networks و deep belief networks و recurrent neural networks و convolutional neural network هستند که هر کدام از ساختارهای بالا، در اپلیکیشن‌ها و کاربردهای خاص خود به کار می‌روند. همچنین شبکه‌های عصبی از ساختار عصب‌های مغزی الهام گرفته‌اند و در یادگیری عمیق سعی بر آن شده است که با تکیه بر شبکه‌های عصبی و با استفاده از چند شبکه‌ی عصبی به صورت پشته شده و لایه لایه، مدل‌های پیچیده‌تری ایجاد کرد که قادر به تشخیص پیچیدگی‌های پنهان داده‌های ورودی باشد.

نهایتاً هر شبکه‌ی عصبی عمیق را میتوان به شکل مجموعه‌ای از لایه‌ها در نظر گرفت که هر لایه، بر روی خروجی لایه‌ی قبل خود عملیات‌های مختلف ریاضی انجام می‌دهد و بدین وسیله سعی در ارائه و ایجاد یک مدل ریاضی می‌کند و یک خروجی به شکل زیر تولید می‌کند:

$$f(x) = f[a_{(L+1)}(h_{(L)}(a_{(L)}(\dots(h_{(2)}(a_{(2)}(h_{(1)}(a_{(1)}(x)))))) \dots))]$$

در ساختار شبکه‌های عصبی و شبکه‌های عصبی عمیق هر لایه شامل تعدادی گره^۱ است که عملیات‌های ریاضی در این گره‌ها اتفاق می‌افتد و با توجه به وزن‌های به دست آمده در طول عملیات یادگیری، مقادیر گره‌های هر لایه با توجه به لایه‌ی قبل به دست می‌آید.



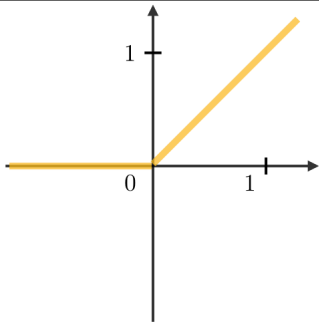
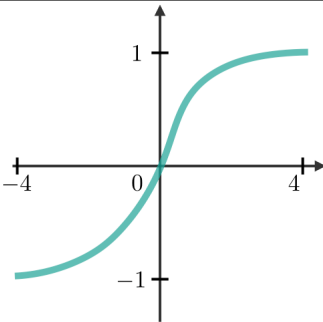
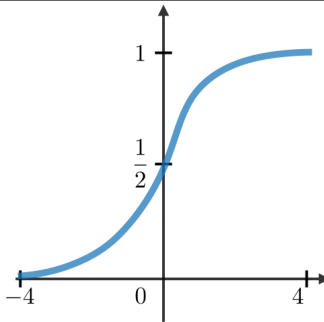
شکل ۲-۱: نحوه محاسبه مقدار هر گره [۱۷]

همچنین در هر لایه با مفهومی به نام تابع فعال‌ساز^۲ روبرو هستیم که درواقع بر روی هر گره یک تابع ریاضی مانند tanh یا relu در طول فرایند یادگیری اعمال می‌شود که این عملیات باعث افزودن پیچیدگی بیشتر به مدل و غیر خطی کردن خروجی‌های گره‌ها می‌شود.

هر کدام از توابع فعال‌ساز مطرح شده را می‌توانید در جدول ۲-۱ ببینید

¹node

²activation function

RELU	Tanh	Sigmoid
$g(z) = \max(0, z)$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \frac{1}{1 + e^{-z}}$
		

جدول ۲-۱: انواع توابع فعال‌ساز [۲]

WORD EMBEDDING ۲-۲

یکی از مشکلات اصلی که در پردازش زبان‌های طبیعی وجود دارد، کار با کلمات است. همانطور که قبلاً به آن اشاره شد، در یادگیری ماشین و همچنین در یادگیری عمیق، هدف آن است که با به دست آوردن یک مدل ریاضی، یک ورودی خاص را به یک خروجی مشخص انطباق دهیم (در مدل‌های نظارت‌شده). حال در پردازش زبان‌های طبیعی، مشکل اصلی این موضوع است که عناصر تشکیل‌دهنده ورودی، کلمات یا حروف هستند در حالی که مدل ریاضی انتظار دارد که ورودی‌ها در غالب اعداد ریاضی باشد تا با استفاده از آن‌ها و با توجه به تابع هزینه و تابع خطا، مدل را یادگیری کند. بنابراین یکی از مراحل اصلی کار، تبدیل فضای ورودی از کلمه‌ها به اعداد است که در این فرایند خود نکات زیادی وجود دارد. شاید یکی از ساده‌ترین روش‌ها این باشد که به جای هر کلمه‌ی مطرح شده در داده‌های ورودی، یک عدد نظیر کنیم. برای مثال به جای کلمه‌ی «خانه» عدد ۱ و به جای کلمه‌ی «پردازش» عدد ۲ را قرار دهیم و به همین روال ادامه دهیم. مشکل اصلی که در این سیستم وجود دارد این است که علی‌رغم بیان ورودی‌ها به شکل عدد و ریاضی (که مناسب مدل‌های ریاضی و یادگیری هستند) در این بیان، عملاً کلمه‌ی «پردازش» از کلمه‌ی «خانه» بزرگ‌تر است و حال آنکه چنین رابطه‌ای بین دو کلمه‌ی مطرح شده وجود ندارد و این موضوع هنگامی که این ورودی‌ها را به مدل می‌دهیم، بیشتر نمایانگر می‌شود زیرا مدل ریاضی به دست آمده چیزی جز توابع ریاضی نیستند و در هنگام یادگیری، آن‌ها برای کلمه‌ی «پردازش» وزن بیشتری نسبت به کلمه‌ی «خانه» قائل خواهند شد. برای فایق آمدن بر این مسئله، یکی از روش‌های مطرح شده استفاده از تکنیکی به اسم رمزگذاری یک رمزگذار^۱ است. در این رمزگذاری، با فرض آن‌که تعداد کل کلمات موجود برابر X باشد، برای هر کلمه به طور معادل یک آرایه با ابعاد X در نظر

^۱one hot encoder

گرفته می‌شود که تنها یکی از خانه‌های آن ۱ و بقیه ۰ است. به طور مثال برای کلمه‌ی «خانه» در این حالت یک آرایه‌ی به طول X در نظر گرفته می‌شود که خانه‌ی اول آن ۱ است و بقیه‌ی خانه‌ها ۰ است و یا برای کلمه‌ی «پردازش» یک آرایه‌ی به طول X نظر گرفته می‌شود که خانه‌ی دوم آن ۱ است و مابقی خانه‌ها ۰ است و بدین ترتیب میتوان بدون آنکه رابطه‌ی خاصی را بین کلمات مختلف متصور شد، کلمات مختلف را به شکل‌های متفاوتی نمایش داد و به مدل به عنوان ورودی داد. البته این مدل خود دو ایراد اساسی دارد. ایراد اول آن است که برای مثال در زبان انگلیسی بیش از ۲۰۰ هزار کلمه‌ی یکتا وجود دارد و برای زبان فارسی نیز به همین شکل است. بنابراین نمایش کلمات به شکل آرایه‌های ۲۰۰ هزار تایی علاوه بر مصرف مقدار زیادی از حافظه، عملیات یادگیری را به شدت سخت و آهسته می‌کند (مخصوصاً با توجه به ویژگی تنک بودن^۱ آرایه‌ها). ایراد دوم این است که کلماتی مانند «پادشاه» و «ملکه» را در نظر بگیرید. این دو کلمه با وجود متفاوت بودن، از نظر معنایی مفهوم‌های مشابهی دارند و این قاعده برای کلماتی مانند «زن» و «مرد» نیز صادق است و این تشابه‌های معنایی خود قسمت اعظمی از اطلاعات پنهان داخل داده‌های ورودی است که میتوان گفت بدون در نظر گرفتن آن‌ها عملیات تولید متن به مشکل برخورد خورد. در این حالت است که از تکنیکی به اسم word embedding برای فایق آمدن بر هر دو مشکل مطرح شده استفاده می‌شود.

درواقع word embedding اقدام به ارائه‌ی یک آرایه یا یک وکتور با یک اندازه‌ی مشخص برای هر کلمه می‌کند که مقادیر این وکتور به صورت پیوسته بوده است و سعی بر آن شده است که در آن‌ها کلمات مشابه، دارای وکتورهای نزدیک به یکدیگر باشند.

یکی از مزیت‌های استفاده از وکتورهایی با ابعاد کم در این روش، نیاز به پردازش کم نسبت به روش‌های مشابه است. [۱۰]

درواقع همانطور که در زبان‌های طبیعی ما برای کلماتی مانند «زن» و «مرد» رابطه‌ی خاصی قائل می‌شویم در این روش نیز سعی شده است که برای این کلمات از طریق ارائه‌ی یک آرایه‌ای با ابعاد مشخص، رابطه‌های مد نظر اعمال شود. [۳]

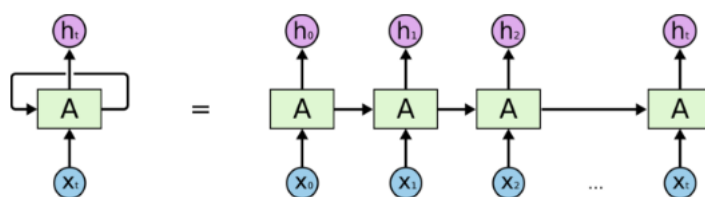
برای مثال میتوان انتظار داشت که در وکتورهای ارائه شده برای کلمات فوق، رابطه‌ی زیر به صورت تقریبی برقرار باشد.

$$\text{«زن»} - \text{«مرد»} = \text{«ملکه»} - \text{«پادشاه»}$$

¹sparse

۲-۳ شبکه‌های عصبی بازگشتی

شبکه‌های عصبی بازگشتی^۱ را میتوان مدلی از شبکه‌های عصبی در نظر گرفت که دارای حافظه داخلی می‌باشند. در این شبکه‌ها، برخلاف دیگر مدل‌های شبکه‌ی عصبی، هر نمونه‌ی ورودی نه به شکل یک آرایه از ویژگی‌ها، بلکه به شکل یک دنباله‌ای از آرایه‌هایی شامل ویژگی‌ها است. شبکه‌های عصبی بازگشتی در ذات خود، به صورت بازگشتی تعبیه شده‌اند به شکلی که با دریافت یک ورودی، خروجی آن‌ها نه تنها به ورودی فعلی، بلکه به ورودی‌های قبلی نیز وابسته است و این وابستگی از طریق اعمال یک حافظه‌ی داخلی به دست می‌آید. درواقع این شبکه‌ها پس از رسیدن به یک خروجی، خروجی مد نظر را به شکل یک فیدبک دوباره به داخل شبکه برمی‌گردانند تا در کنار ورودی مرحله‌ی بعد، برای تعیین خروجی بعدی استفاده شود. به دلیل وجود این ویژگی بازگشتی، شبکه‌های عصبی بازگشتی برای ایجاد مدل‌هایی که ذات دنباله‌ای دارند مانند تشخیص گفتار^۲ یا مدل‌های مرتبط با متن بسیار مناسب هستند.



An unrolled recurrent neural network.

شکل ۲-۲: ساختار شبکه‌های عصبی بازگشتی [۱۶]

البته لازم به ذکر است که شبکه‌های عصبی بازگشتی خود مشکلات عدیدی اعم از vanishing and exploding gradient و یا سخت بودن و طولانی بودن عملیات یادگیری دارند و بایستی به این نکته نیز توجه کرد که به دلیل ماهیت دنباله‌ای بودن یادگیری، استفاده از پردازش‌های موازی نیز در این حالت دچار مشکل می‌شود که این مسأله نیز خود فرایند یادگیری مدل را طولانی‌تر می‌کند.

$$h_t = f(h_{t-1}, x) \quad (۱-۲)$$

مشکل دیگری که این شبکه‌ها دارند نیز این موضوع است که حافظه‌ی موجود قادر به نگهداری داده‌ها و فیدبک‌های خیلی قدیمی نمی‌باشد و با توجه به ساختار شبکه، پس از مدتی داده‌های قدیمی اصطلاحاً محو می‌شوند.

بسته به استفاده‌ی شبکه‌های عصبی بازگشتی، این شبکه‌ها می‌توانند کاربردهای زیاد و متنوعی داشته باشند

^۱recurrent neural networks

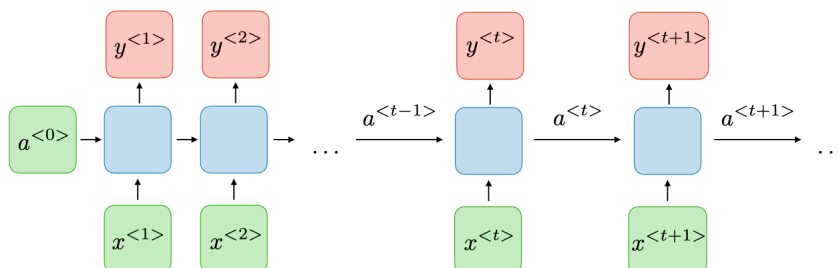
^۲speech recognition

که در جدول ۲-۲ خلاصه شده است

نوع	شماتیک	مثال
یک به یک $T_x = T_y = 1$		شبکه‌ی عصبی عادی
یک به چند $T_x = 1, T_y > 1$		تولید موسیقی
چند به یک $T_x > 1, T_y = 1$		دسته‌بندی معنایی
چند به چند $T_x = T_y$		تشخیص نقش کلمه ^۱
چند به چند $T_x \neq T_y$		ترجمه‌ی ماشینی ^۲

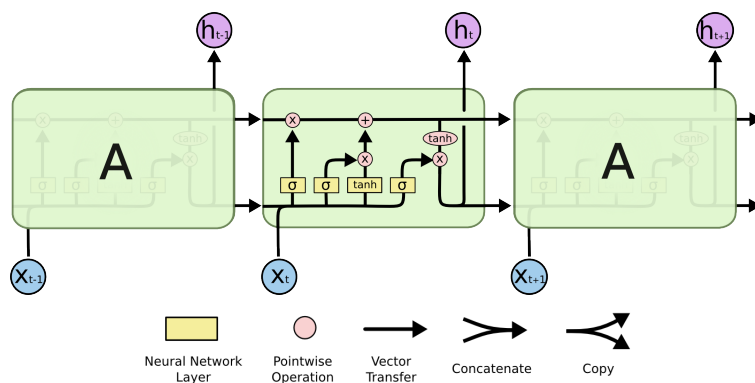
جدول ۲-۲: کاربردهای شبکه‌های عصبی بازگشتی [۲]

ساختار LSTM یک ساختار خاص از مدل‌های شبکه‌های عصبی بازگشتی است. همانطور که قبلاً گفته شد شبکه‌های عصبی بازگشتی دارای ساختاری شبیه به شکل ۲-۳ هستند.



شکل ۲-۳: ساختار شبکه‌های عصبی بازگشتی [۲]

هدف اصلی LSTM ها فایده آمدن بر موضوع vanishing gradient problem است که در این مسئله، مدل پس از گذشت زمان داده‌های قدیمی در سری زمانی را فراموش کرده و وزن کمی برای داده‌های قدیمی قائل می‌شود و به همین دلیل در طول یادگیری با مشکل مواجه می‌شود. LSTM با ارائه ساختار خود تلاش بر این دارد که داده‌های یادگیری شده در فیدبک‌های قبلی را در طول زمان بیشتری حفظ کند و برای رسیدن به این هدف از دروازه‌ها^۱ استفاده می‌کند. در این ساختار سه دروازه وجود دارد. دروازه ورودی که به عنوان یک سلول^۲ برای دریافت داده عمل کرده و با تعیین وزن مشخص شده، میزان اهمیت داده‌های ورودی فعلی در سری زمانی را مشخص می‌کند. دروازه خروجی که به عنوان یک سلولی عمل کرده که خروجی را مشخص می‌کند و دروازه فراموشی که درواقع یک سلولی است که میزان اهمیت و وزن داده‌های قبلی در سری زمانی را مشخص می‌کند. ساختار این عناصر همانند شکل ۲-۴ است.

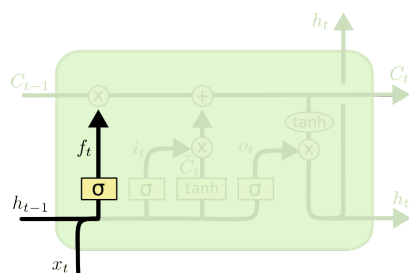


شکل ۲-۴: ساختار LSTM [۱۸]

از آنجایی که LSTM ها یکی از اصلی‌ترین ساختارهای استفاده شده در این پروژه می‌باشند در ادامه به طرز

^۱gates
^۲cell

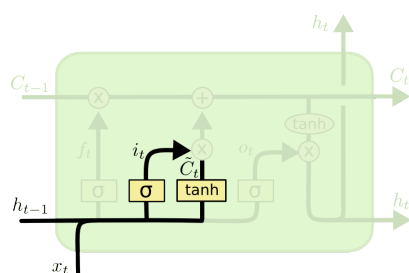
کار LSTM به طور دقیق تر می پردازیم.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

شکل ۲-۵: دروازه‌ی فراموشی [۱۸]

هر LSTM دارای یک حافظه‌ی داخلی به اسم C یا cell state است که در تصویر مشخص شده است. در مرحله‌ی اول LSTM تصمیم می‌گیرد که چه مقدار از cell state مرحله‌ی قبل را نگهداری کند و چه مقدار از داده‌های cell state را دور انداخته و استفاده نکند. برای رسیدن به این هدف C_{t-1} در یک ضرب به صورت ضرب نقطه‌ای ضرب می‌شود که میتوان این ضرب را به صورت یک وزن برای مشخص کردن هدف مطرح شده در نظر گرفت. میزان این ضرب توسط یک لایه به اسم دروازه‌ی فراموشی تعیین می‌شود که این دروازه خود از یک تابع sigmoid تشکیل شده است. دروازه‌ی فراموشی همانطور که در شکل بالا مشخص شده است، ابتدا مقدار h_{t-1} (مقدار h در سری زمانی قبلی) و x_t (مقدار ورودی جدید سری زمانی) را در کنار یکدیگر قرار داده و با ضرب کردن در یک وزن (که این وزن در طول فرایند یادگیری، یاد گرفته می‌شود) و اضافه کردن مقدار بایاس^۱ (که این مقدار نیز در طول فرایند یادگیری، یاد گرفته می‌شود) مقدار جدیدی به دست می‌آورد که با گذر دادن این مقدار از تابع sigmoid به یک عدد بین ۰ و ۱ میرسد. ۱ به معنی این است که مقدار C_{t-1} را کاملاً نگهداری کرده و ۰ به معنی دور ریختن کامل مقدار C_{t-1} است.



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

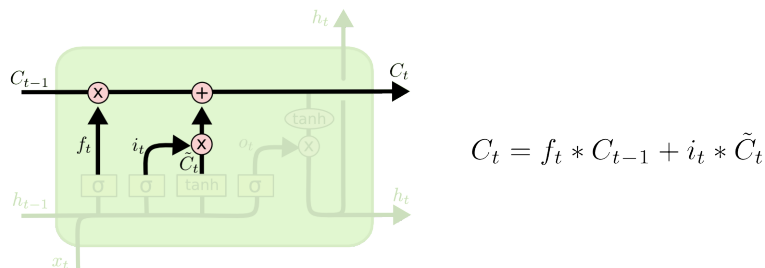
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

شکل ۲-۶: دروازه‌ی ورودی [۱۸]

در مرحله‌ی بعد همانطور که طبق شکل فوق مشخص شده است، LSTM تصمیم می‌گیرد که از مقدار جدید دریافتی از ورودی، چه مقدار را در cell state خود نگهداری کند. در این مرحله از یک دروازه به اسم دروازه‌ی ورودی استفاده می‌کنیم که روند کار مانند قبل است و مشخص می‌کند چه مقدار از ورودی قرار است در cell

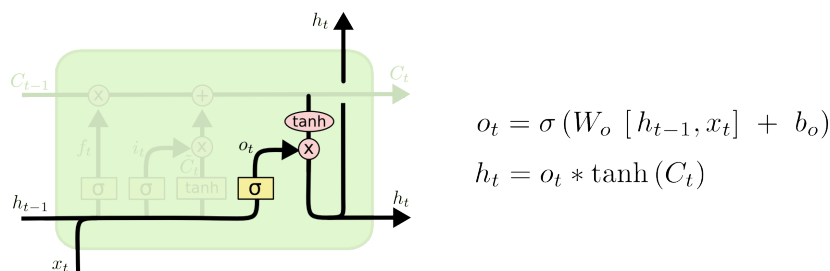
^۱bias

state ذخیره شود. برای این منظور ابتدا دوباره از ترکیب h_{t-1} و x_t و گذر دادن آن از یک تابع sigmoid ضریب نگه‌داری مقدار ورودی را به دست می‌آوریم که این مقدار را i_t می‌نامیم. از طرف دیگر h_{t-1} و x_t رو پس از ترکیب از تابع \tanh گذر می‌دهیم تا مقدار \tilde{C}_t را به دست آوریم. از حاصل ضرب \tilde{C}_t در مقدار خروجی دروازه‌ی ورودی تهیه می‌شود که مشخص می‌کند چه مقدار از ورودی فعلی قرار است در cell state ذخیره شود.



شکل ۷-۲: Cell State [۱۸]

نهایتاً مقدار به دست آمده توسط دروازه‌ی ورودی با مقدار به دست آمده در دروازه‌ی فراموشی جمع شده و مقدار فعلی Cell state را ایجاد می‌کند. در آخرین مرحله باید مشخص کنیم که این LSTM در سری زمانی فعلی، چه خروجی خواهد داشت که این مقدار توسط دروازه‌ی خروجی مشخص می‌شود. در حال حاضر این مقدار در cell state ذخیره شده است ولی به جای خروجی دادن خود cell state سعی می‌کنیم یک نسخه‌ی ویرایش شده از آن را خروجی دهیم.



شکل ۸-۲: دروازه‌ی خروجی [۱۸]

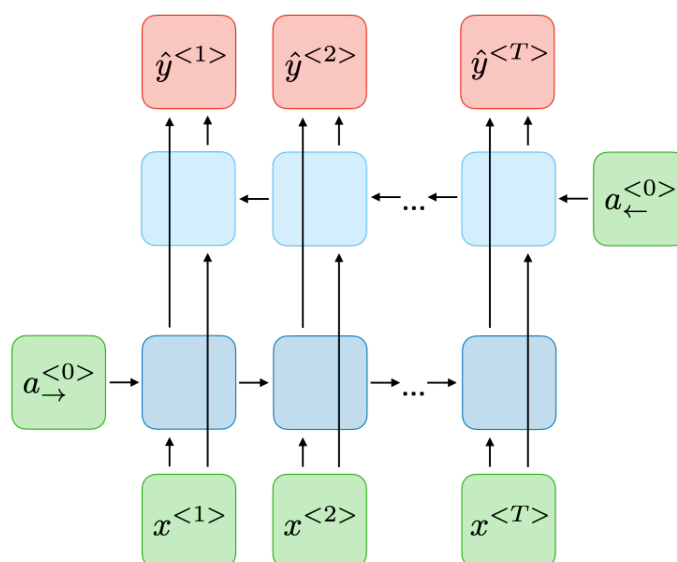
بدین منظور دوباره مانند قبل، ترکیب h_{t-1} و x_t را از یک تابع sigmoid می‌گذرانیم تا مشخص کنیم که چه مقدار از cell state را قرار است خروجی دهیم. سپس مقدار cell state را از یک تابع \tanh گذر داده (تا مقدار آن بین -1 و 1 شود) و سپس با استفاده از مقدار به دست آمده در تابع sigmoid و ضرب آن در خروجی تابع \tanh ، مقدار دروازه‌ی خروجی را به دست می‌آوریم و آن را به عنوان خروجی نهایی، خروجی می‌دهیم.

البته لازم به ذکر است که مورد مطرح شده یکی از ساده‌ترین ساختارهای LSTM بود و LSTM دارای ساختارهای متفاوت و متنوع زیادی است که در برخی از کاربردها و عملکردها، ممکن است نتایج متفاوتی را

به همراه داشته باشند.

همچنین از ساختارها و گره‌های LSTM در عمل به شکل‌های مختلفی استفاده می‌شود. برای مثال برای آنکه داده‌های بیشتری در این ساختارها داشته باشیم، به طور معمول میتوان از شبکه‌های دوطرفه و شبکه‌های عمیق و یا پشته شده استفاده کرد.

در حالت دو طرفه، درواقع به ازای هر قدم در زمان به جای یک خروجی دو خروجی خواهیم داشت بدین شکل که دو شبکه‌ی بازگشتی عمیق تعبیه شده که در یکی بر اساس زمان و دنباله‌ی ورودی رو به جلو حرکت کرده و در دیگری در جهت مخالف زمان شبکه را یادگیری می‌کنیم و نهایتاً از هر کدام یک خروجی گرفته و از ترکیب این دو خروجی استفاده می‌کنیم (با استفاده از یک تابعی مانند argmax یا یک فرایند ریاضی سعی می‌کنیم از ترکیبی از هر دو خروجی استفاده کنیم)



شکل ۲-۹: LSTM دوطرفه [۱۸]

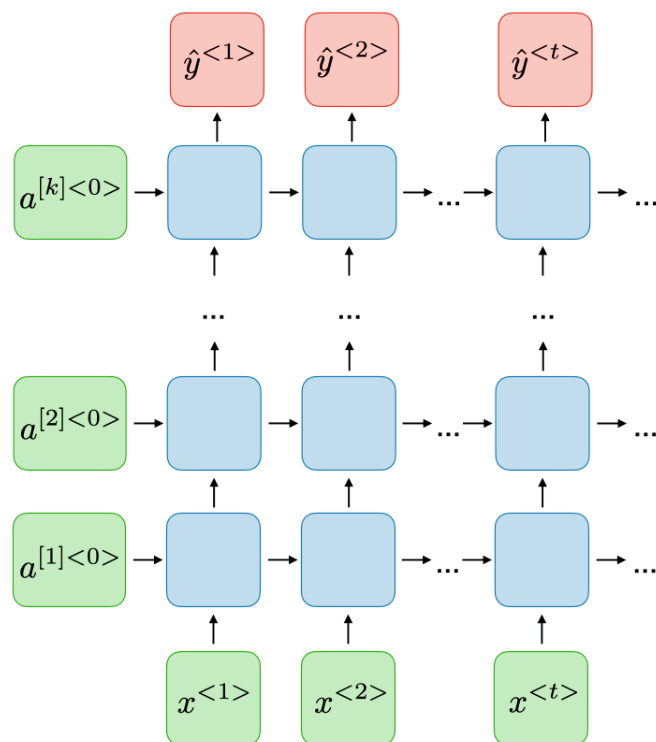
در شبکه‌های پشته شده، خروجی لایه‌ی اول که یک شبکه‌ی بازگشتی است به لایه‌ی بعدی که خود آن نیز دوباره یک شبکه‌ی بازگشتی است داده می‌شود و سعی می‌شود که با استفاده از لایه‌های بیشتر، پیچیدگی‌های بیشتری از داده دریافت شود.

۲-۵ مدل خود رمزگذار متغیر

یکی دیگر از روش‌هایی که در سال‌های اخیر توجه زیادی در مدل‌های تولیدکننده به خود جلب کرده‌است، استفاده از ساختارهای خود رمزگذار^۱ و خود رمزگذارهای متغیر^۲ است. در مدل‌های خود رمزنگار، تلاش بر

^۱auto encoder

^۲variational auto encoder



شکل ۲-۱۰: LSTM پشته شده [۱۸]

آن است که دو شبکه‌ی عمیق مجزا در کنار هم توسعه داده شده و اقدام به یادگیری کنند. در این مدل‌ها ورودی و خروجی اعمال شده به مدل یکسان است بنابراین مدل سعی می‌کند که ساختار خود داده‌ی ورودی را یادگیری کند و به این مدل‌ها به شکل مدل‌های خود یادگیر^۱ نیز میتوان نگاه کرد. در این مدل، شبکه‌ی عمیق اول رمزگذار^۲ و شبکه عمیق دوم رمزگشا^۳ است. در ابتدا شبکه رمزگذار سعی می‌کند که با دریافت یک ورودی خاص، این ورودی را به یک فضای پیوسته با ابعاد مشخص نگاشت^۴ دهد که این فضای پیوسته به عنوان فضای نهفته شناخته می‌شود. شبکه‌ی عمیق دوم در جهت عکس شبکه‌ی قبلی کار کرده و تلاش می‌کند که با انطباق فضای نهفته به داده‌ی اصلی، داده‌ی اصلی را بازسازی کند. بنابراین یکی از ایده‌های اصلی در این پروژه برای تولید متن و ایجاد مدل مولد متن می‌تواند این باشد که با استفاده از خود رمزگذار، ابتدا شبکه‌های رمزگذار و رمزگشا و فضای نهفته یادگیری شود و داده‌های یادگیری به یک فضای نهفته ی پیوسته با یک توزیع احتمالی خاص نگاشت پیدا کند. سپس اگر از توزیع احتمالی به دست آمده فضای نهفته بتوان نمونه‌گیری^۵ به صورت تصادفی انجام داد، با دادن نمونه‌ی به دست آمده به شبکه‌ی رمزگشا، میتوان انتظار داشت که عملیات انطباق صورت گرفته و متن‌های جدیدی تولید شود که شبیه به متن‌های اصلی است. درواقع میتوان به فضای نهفته به

^۱self learning model

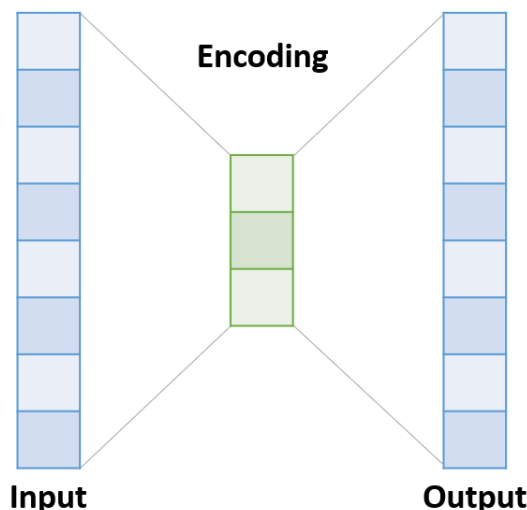
^۲encoder

^۳decoder

^۴map

^۵sampling

عنوان یک فضای احتمالاتی نگاه کرد که همانند روش‌هایی مانند PCA سعی در کاهش ابعاد و پیچیدگی مسأله داشته [۱۹] و برای رسیدن به این هدف شبکه‌ی رمزگذار بر روی داده‌ها عملیات یادگیری را انجام می‌دهد. در سال‌های اخیر شبکه‌های خود رمزنگار کاربردهای فراوانی در کار با تصویر داشته‌اند که در این پروژه با ترکیب این مدل‌ها با شبکه‌های بازگشتی، سعی در استفاده از آن‌ها در حیطه‌ی پردازش زبان‌های طبیعی داریم.



شکل ۲-۱۱: ساختار مدل‌های خود رمزگذار [۱۵]

مشکل اصلی که در کار با شبکه‌های خود رمزگذار پیش می‌آید عملیات نمونه‌گیری است زیرا با وجود نگاشت داده‌ها به یک فضای نهفته، این فضای نهفته خود نیز دارای توزیع احتمالی پیچیده‌ای خواهد بود که نمونه‌گیری از آن به طور مستقیم کار ساده‌ای نیست. بنابراین برای رفع این مشکل، از مدل‌های خود رمزگذار متغیر استفاده می‌شود که علاوه بر عملکرد به صورت یک خود رمزگذار، سعی بر آن می‌کند که توزیع احتمالاتی فضای نهفته، دارای توزیعی نزدیک به فضای نرمال^۱ یا دارای توزیع گاوسی^۲ باشد و برای رسیدن به این هدف نیز از توابع خطایی همچون KL Divergence استفاده می‌کند. نهایتاً اگر به چنین هدفی دست یابیم، با استفاده از روش‌های نمونه‌گیری میتوان عملیات نمونه‌گیری از فضای نهفته را انجام داد. بنابراین برای رسیدن به این هدف، به جای آنکه قسمت رمزگذار مدل در خود رمزگذارهای متغیر، یک ورودی را به یک نقطه در فضای نهفته انطباق دهد، بلکه نقطه را به یک توزیع احتمالی نگاشت می‌کند. درواقع یادگیری مدل به شکل زیر خواهد شد:

- ابتدا ورودی به یک توزیع احتمالی در فضای نهفته رمزگذاری می‌شود
- در مرحله‌ی بعدی، یک نقطه از فضای نهفته نمونه برداری می‌شود
- نقطه‌ی نمونه برداری شده به عنوان ورودی به رمزگشا داده‌شده و به ورودی ابتدایی بازگردانده می‌شود و در

^۱normal space

^۲gaussian distribution

این حین سعی می‌شود که خطای بازگردانی کمینه گردد

- در نهایت خطای بازسازی^۱ در شبکه بصورت عقب‌گرد منتشر می‌شود^۲ و سعی می‌شود که مدل با استفاده از آن یاد بگیرد

۲-۶ شبکه‌های مولد خصمانه

یکی دیگر از مدل‌هایی که در سال‌های اخیر توجه بسیاری را به خود در زمینه‌ی ایجاد مدل‌های مولد جلب کرده‌است، شبکه‌های مولد خصمانه^۳ است. [۱۱] ساختار این شبکه که الهام گرفته از مدل‌های نظریه‌ی بازی^۴ است، متشکل از دو مدل مجزا است که به صورت رقابتی با یکدیگر عمل کرده و در این حین سعی می‌کنند که در کنار غلبه بر یکدیگر، هر دو بهبود یافته و پیشرفت کنند. این دو مدل، یکی مدل مولد^۵ و دیگری مدل تمیزدهنده^۶ است. در رقابت بین این دو مدل، مدل مولد همواره سعی می‌کند با دریافت یک ورودی (که اغلب به شکل یک نویز به عنوان عملگر تصادفی است) خروجی‌های جدید تولید کند. از طرف دیگر مدل تمیزدهنده در هر بار یک ورودی دریافت می‌کند که این ورودی، یا یک داده‌ی اصلی است یا یک داده‌ی تقلبی است که توسط مدل مولد تولید شده است. مدل مولد بایستی که در فرایند یادگیری، نهایتاً بتواند که بین داده‌های اصلی و داده‌های تقلبی تفاوت را تشخیص دهد و از طرف دیگر در رقابت با این مدل، مدل تولیدکننده باید داده‌هایی را تولید کند که مدل تمیز دهنده را به اشتباه بیندازد. نهایتاً هنگامی که مدل تمیزدهنده به عنوان ورودی با یکی از داده‌های تقلبی روبرو می‌شود، اگر بتواند که تقلبی بودن آن را درست تشخیص بدهد (یا ندهد)، خروجی مدل تمیزدهنده به عنوان یک فیدبک به مدل تولیدکننده باز خواهد گشت و مدل تولیدکننده سعی می‌کند با استفاده از رفتار مدل تمیزدهنده در برابر داده‌های جعلی، خود را بهبود ببخشد. [۱۱]

درواقع این تقابل دو مدل را میتوان به شکل یک min-max در قالب رابطه‌ی زیر مدل کرد

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2-2)$$

در رابطه‌ی فوق، تابع D بیانگر احتمال آن است که داده‌ی ورودی x از توزیع داده‌های اصلی آمده باشد (و نه از توزیع داده‌های تقلبی). حال در رابطه‌ی بالا، مدل تمیزدهنده در تلاش است که اگر یک داده از توزیع داده‌های اصلی آمده باشد مقدار تابع $D(x)$ را به سمت یک بیشینه کند تا مقدار $\log(D(x))$ افزایش یابد و اگر داده از توزیع داده‌های تقلبی به وجود آمده باشد مقدار $1 - D(G(z))$ را کمینه کند که $G(z)$ خروجی مدل تولیدکننده

¹reconstruction error

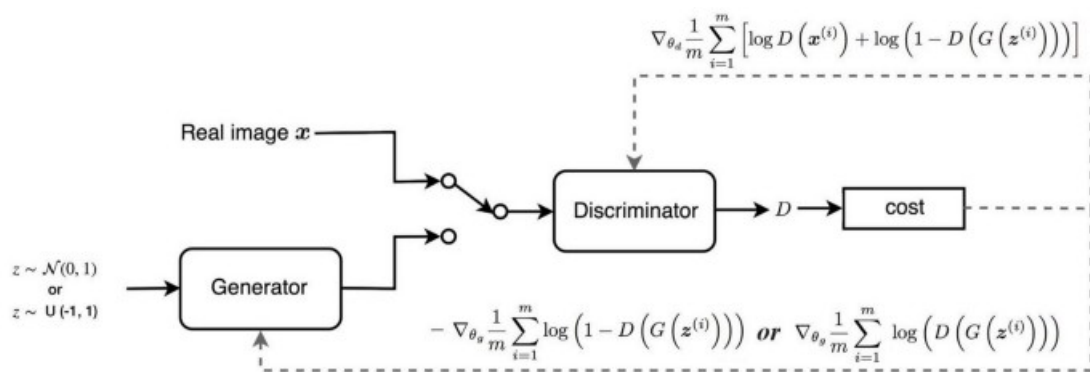
²back propagated

³generative adversarial networks

⁴game theory

⁵generator

⁶discriminator



شکل ۲-۱۲: ساختار مدل‌های مولد خصمانه [۱۲]

بر روی نویز ایجاد شده از توزیع p_z است. از طرف دیگر نیز مدل تولیدکننده در جهت عکس سعی بر آن دارد که تابع V را که بیانگر بیشینه سود مدل تمیزدهنده است را کمینه کند. [۱۱]

تصویر ۲-۱۲ نهایتاً شبکه‌های تولیدکننده‌ی خصمانه را دقیق‌تر تشریح می‌کند

۲-۷ پیش‌پردازش زبان‌های طبیعی

یکی از قسمت‌های مهم در پردازش زبان طبیعی، پیش‌پردازش داده‌های ورودی است. این عملیات از اهمیت ویژه‌ای برخوردار است زیرا داده‌های ورودی در ابتدای کار ممکن است تمیز^۱ نباشند و قبل از اعمال داده‌ها به مدل به عنوان ورودی، نیاز به اصلاح داشته باشند. از مهم‌ترین عملیات‌های پیش‌پردازش متن میتوان به Tokenize کردن و افزودن Padding اشاره کرد. از آنجایی که داده‌های ورودی اغلب به شکل یک جمله^۲ هستند ولی مدل نیاز به یک ورودی به شکل یک سری زمانی دارد، با اعمال عملیات Tokenization اقدام به ایجاد ورودی مناسب می‌کنیم. در این مرحله یک جمله به یک دنباله‌ی زمانی از کلمات تبدیل می‌شود. در این حین ممکن است کلماتی که از اهمیت کمتری در مدل برخوردار هستند (مانند علائم نگارشی) برای افزایش کارایی مدل حذف گردند. نکته‌ی دیگری نیز که مطرح است Padding است. بسیاری از مدل‌های طراحی شده توسط هوش مصنوعی و مدل‌های سری زمانی، نیاز به ورودی به طول ثابت دارند که باید این مورد از قبل مشخص شده باشد تا بتوانند مدل مطرح را ایجاد کنند. از طرف دیگر پس از Tokenize کردن به دلیل اینکه جملات مختلف دارای تعداد کلمات متفاوت هستند، طول آرایه‌های داده‌های مختلف ورودی، متفاوت خواهد بود. در این مرحله از Padding استفاده شده و برای یکسان کردن طول تمام داده‌ها، به ورودی‌هایی که اندازه‌ی آن‌ها کوچکتر است Token ی خاص به عنوان Padding اضافه می‌شود.

^۱clean

^۲sentence

فصل سوم

معرفی داده‌های مسأله

در این پروژه قصد داریم که در نهایت اقدام به تولید متن‌های فارسی کنیم. بنابراین نیاز به داده‌های فارسی داریم. از بین داده‌های مختلف فارسی موجود، با توجه به هدف‌های مختلف (تولید متن‌های محاوره‌ای یا تولید متن‌های فاخر یا تولید متن‌های داستان‌گونه) میتوان داده‌ی مدنظر را انتخاب کرد. برای این پروژه تصمیم بر آن شد که در نهایت، مدل‌های ایجاد شده اقدام به تولید شعر و متن‌های فاخر کنند و بنابراین مجموعه‌ی داده‌ی استفاده شده، مجموعه‌ی اشعار فارسی است.

در سال‌های اخیر پروژه‌ی گنجور [۸] با هدف گردآوری و تنظیم اشعار و متون سخنرایان فارسی تحت وب اقدام به فعالیت کرده است که با توجه به هدف پروژه، داده‌های موجود در این وب سایت مناسب به نظر می‌رسد. همچنین با محوریت پروژه‌ی گنجور، داده‌ها توسط امین قادری در یک مخزن گیت‌هاب [۹] تهیه و تنظیم شده‌است و اقدام به نرمال سازی آن‌ها شده‌است که در این پروژه برای یادگیری مدل‌ها از اشعار نرمال شده‌ی فردوسی استفاده شده‌است که در مرحله‌ی نرمال سازی، حروف عربی با معادل فارسی آن‌ها جایگزین شده‌است (مانند تفاوت ی در فارسی و عربی).

در مجموعه داده‌ی مورد استفاده، هر مصرع از شعر در یک خط نوشته شده‌است و همچنین بین هر دو کلمه یک فاصله وجود دارد. در این مجموعه‌ی داده در مجموع ۹۹۲۱۷ مصرع وجود دارد. همچنین تمام ابیات مطرح شده در قالب مثنوی هستند و تمامی این ابیات از شاهنامه‌ی فردوسی جمع‌آوری شده‌است. در این پروژه در مرحله‌ی اول مصرع‌های موجود دو به دو با یکدیگر زوج شده و نهایتاً تشکیل ۴۹۶۰۸ بیت دادند. همچنین برای تشخیص مکان اتمام یک مصرع و شروع مصرع بعدی، در مرحله‌ی پیش پردازش یک token بین هر دو مصرع قرار داده شده و در صورت نیاز مدل، ممکن است token‌هایی در شروع و پایان یک بیت نیز قرار داده شود. همچنین تمامی ابیات مورد استفاده نهایتاً دارای ۱۷۷۶۱ کلمه‌ی یکتا هستند. طولانی‌ترین بیت دارای ۲۰ کلمه است که بیت «نه جا هست ما را نه بوم و نه بر - نه سیم و سرای و نه گاو و نه خر» است. همچنین در مرحله‌ی پیش پردازش، اقدام به همسان کردن طول ابیات کرده که این کار را با استفاده از تکنیک padding انجام داده و نهایتاً به ته هر بیت، تعدادی token با ارزش ۰ اضافه می‌کنیم.

فصل چهارم

حل مسئله

در این پروژه برای فایق آمدن بر مسئله‌ی مطرح شده، از روش‌های زیر استفاده شد و مدل‌های مختلفی که در ادامه به آنها خواهیم پرداخت بررسی گردیدند. همچنین در این فصل علاوه بر پرداختن به جزئیات هر مدل، به جزئیات پیاده‌سازی‌های انجام شده نیز پرداخته خواهد شد. لازم به ذکر است که در تمام پیاده‌سازی‌های انجام شده از کتابخانه‌ی Tensorflow استفاده شده است. Tensorflow یک بستر و مجموعه‌ای از کتابخانه‌ها و ابزارهای رایگان و متن‌باز است که به افراد اجازه‌ی پیاده‌سازی موثرتر و بهینه‌تر شبکه‌های عصبی و سیستم‌های پیچیده یادگیری ماشین و... را می‌دهد.

۴-۱ مدل شبکه عصبی بازگشتی

از ساختارهای شبکه‌های بازگشتی همانطور که قبلاً اشاره شد، در شکل‌های متفاوتی میتوان استفاده کرد. یکی از کاربردهای شبکه‌های بازگشتی چند به چند، مدل‌های تولیدکننده است که در این حالت شبکه‌های بازگشتی با یادگیری روند^۱ یک دنباله‌ی ورودی، سعی بر تقلید و ایجاد دنباله‌ای مشابه می‌کنند. یکی از ساده‌ترین راه‌های ایجاد یک مدل زبانی^۲ استفاده‌ی مستقیم از شبکه‌های عصبی بازگشتی است. در این روش که در اکثر مواقع

^۱pattern

^۲language model

برای بهره‌برداری بیشتر از LSTM ها استفاده می‌شود، سعی بر آن است که در نهایت خروجی به دست آمده یک مدل احتمالاتی زبان بر اساس شبکه‌های عصبی^۱ باشد. [۴]

در این روش که بسیار شبیه به استفاده از مدل‌های sequence-to-sequence است، داده‌های یادگیری به شکل یک دنباله از w_1, w_2, \dots, w_t است که هر w یک کلمه‌ی عضو مجموعه‌ی کلمات V است. همچنین V با وجود داشتن اندازه‌ی بزرگ، یک مجموعه‌ی شمارا و دارای تعداد مشخص عضو است. هدف نهایی آن است که یک مدل مناسب به شکل $P(w_t | w_1^{t-1}) = F(w_t, w_{t-1}, w_{t-2}, \dots, w_{t-n})$ به دست بیاید که در عمل با مشاهده‌ی $t-1$ کلمه‌ی اول، با ضریب اطمینان مناسب و تقریب مناسبی^۲ احتمال رخ دادن کلمه‌ی t ام را از مجموعه‌ی V به دست بیاورد. همچنین باید شرط زیر نیز برای هر انتخاب w_1^{t-1} برقرار باشد:

$$\sum_{i=1}^{|V|} P(w_i | w_1^{t-1}) = 1 \quad (1-4)$$

برای رسیدن به این هدف، ساختارهای متفاوتی می‌توان ارائه داد [۴] که در ادامه در این پروژه ساختار مستقیم استفاده شده است. در ساختار مستقیم استفاده شده، پس از پیش‌پردازش اولیه داده‌ها که در فصل قبل به آن اشاره شد، به اول هر بیت از داده‌های ورودی^۳ یک توکن به عنوان «شروع بیت» اضافه شده و به انتهای داده‌های خروجی^۴ نیز یک توکن «پایان بیت» اضافه شده است و به مدل ایجاد شده داده شده است. همچنین مدل مورد استفاده شده، یک مدل پشته شده از LSTM ها است که در لایه‌ی اول خود و قبل از رسیدن داده‌ها به LSTM از یک لایه‌ی word embedding استفاده شده است و نهایتاً پس از عبور داده‌ها از گره‌های LSTM با استفاده از لایه‌های کاملاً متصل شبکه‌ی عصبی^۵ به ازای هر واحد زمانی^۶ در داده‌ی ورودی، $|V|$ خروجی متفاوت داده می‌شود که خروجی i ام متناظر با $P(w_i | w_1^{t-1})$ است. برای برقرار بودن شرط ۴-۱ نیز از یک لایه‌ی softmax در آخرین لایه‌ی شبکه‌ی عصبی استفاده شده است. درواقع در هنگام خروج V مقدار مختلف از یک لایه‌ی softmax تغییر زیر به طور همزمان بر روی مقادیر تمام گره‌ها رخ می‌دهد.

$$node_i = \frac{e^{node_i}}{\sum_i e^{node_i}} \quad (2-4)$$

¹neural probabilistic language model

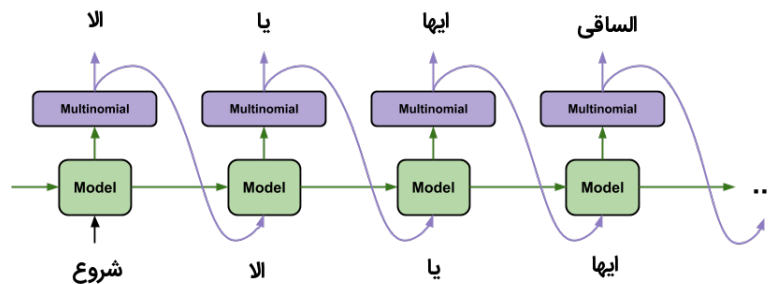
²likelihood

³Input Set

⁴output set

⁵fully connected layers

⁶time step



شکل ۴-۱: ساختار مدل مطرح شده

۴-۱-۱ توابع خطا و دقت

یکی از مهم‌ترین موارد هر مدل که علاوه بر ایجاد معیار سنجشی برای مشخص کردن میزان صحت و درست عمل کردن مدل، برای یادگیری خود مدل نیز بسیار حیاتی است تعیین یک تابع خطای^۱ مناسب است که در حین عملیاتی مانند انتشار روبه عقب^۲ با استفاده از آن و عملیات‌های ریاضی مانند مشتق‌گیری، مدل اقدام به یادگیری کند. در مدل مورد بحث در این پروژه از توابع خطا و صحت^۳ زیر استفاده شد.

اگر در یک واحد زمانی در لایه‌ی آخر مدل، V خروجی متفاوت داشته باشیم که این V خروجی نیز از تابع softmax گذر کرده باشند، و کلمه‌ی مورد انتظار ما w_x باشد، خطای محاسبه شده برابر خواهد بود با:

$$-\log(q(w_x)) \quad (۴-۳)$$

که $q(x)$ مقدار گره شماره‌ی x در آخرین لایه را مشخص می‌کند. بنابراین با توجه به اینکه ما در حال محاسبه‌ی خطای یک سری زمانی هستیم، خطای نهایی برابر می‌شود با:

$$\sum_{maxlength} -\log(q(x)) \quad (۴-۴)$$

بنابراین خطای نهایی در یک دسته^۴ از ورودی‌ها به شکل میانگین خطای تمام عناصر آن دسته است. همچنین دقت مدل نیز بدین شکل محاسبه می‌شود که ابتدا برای هر نمونه^۵ خروجی لایه‌ی آخر ($q(x)$) بررسی شده و argmax آن را به دست می‌آوریم. اگر مقدار $\text{argmax}(q(x))$ برابر با X (خروجی مورد انتظار ما) بود دقت این نمونه ۱ و در غیر این صورت دقت آن ۰ خواهد بود. نهایتاً در یک دسته از ورودی‌ها دقت نهایی برابر با میانگین دقت تمام عناصر خواهد بود.

^۱loss function

^۲back propagation

^۳accuracy

^۴batch

^۵sample

در مدل شبکه‌ی عصبی بازگشتی پیاده‌سازی شده، به عنوان اولین لایه‌ی پنهان^۱ از یک لایه‌ی word embedding با اندازه‌ی ۲۵۶ استفاده شد. همچنین پس از آن سه لایه‌ی LSTM به صورت پشته شده^۲ استفاده شد که هر کدام از آن‌ها دارای ۵۱۲ عدد گره بودند. پس از آن نیز دو لایه‌ی تماماً متصل^۳ استفاده شد که هر کدام از آن‌ها نیز دارای ۵۱۲ عدد گره بودند. نهایتاً در آخرین لایه برای تولید خروجی از یک لایه‌ی تماماً متصل استفاده شد که تعداد گره‌های آن برابر با تعداد کلمات موجود در داده‌های یادگیری بود.

به دلیل نیاز به پردازش‌های سنگین، برای یادگیری تنها از یک چهارم داده‌های یادگیری موجود استفاده شد که نهایتاً دارای ۲۴۸۰۴ مصرع یا معادلاً ۱۲۴۰۲ بیت بود و در ابیات طولانی‌ترین بیت دارای طول ۲۱ کلمه بود. همچنین برای بررسی درستی عمل یادگیری، هنگام یادگیری ۰.۲ از داده‌ها به عنوان داده‌ی بررسی^۴ مورد استفاده قرار گرفت.

برای یادگیری مدل از تکنیکی مرسوم به یادگیری بازور^۵ استفاده شد. در این تکنیک، هنگام ایجاد و یادگیری دنباله‌ی مد نظر در شبکه‌ی بازگشتی، در مرحله‌ی $t+1$ به عنوان ورودی به جای استفاده از خروجی مرحله‌ی t ام، از داده‌ی یادگیری مرحله‌ی $t+1$ استفاده می‌شود و در هنگام پیشبینی^۶ کردن طبق روال عادی برای ورودی $t+1$ از خروجی t استفاده می‌شود. از مزیت‌های یادگیری بازور میتوان به بهینه‌تر بودن و سریع‌تر بودن فرایند یادگیری مدل اشاره کرد.

همچنین در لایه‌های کاملاً متصل، از تابع ELU به عنوان تابع فعال ساز استفاده شده است.

$$R(z) = \begin{cases} z & z > 0 \\ \alpha \cdot (e^z - 1) & z \leq 0 \end{cases} \quad (۵-۴)$$

از مزایای ELU میتوان به توانایی تولید اعداد منفی و یادگیری بهتر در فرایند اشاره کرد. همچنین برخلاف RELU دارای مشکل مردن^۷ [۱۳] نیز نمی‌باشد.

از معایب استفاده از ELU نیز میتوان به سخت‌تر بودن یادگیری آن و متعاقباً افزایش زمان یادگیری اشاره کرد. همچنین برای جلوگیری از مشکلاتی مانند بیش‌برازش^۸ از لایه‌های حذف تصادفی^۹ استفاده شد و مقدار ۰.۲ برای تمام لایه‌های حذف تصادفی در نظر گرفته شد. نهایتاً در حین یادگیری تغییرات توابع خطا و دقت به

^۱hidden layer

^۲stacked

^۳fully connected

^۴validation set

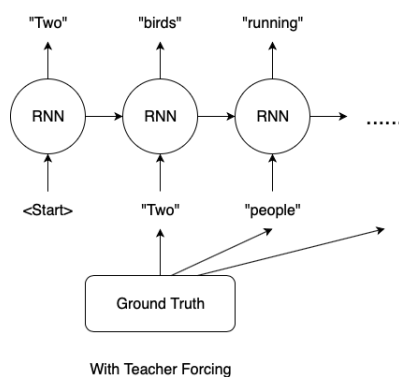
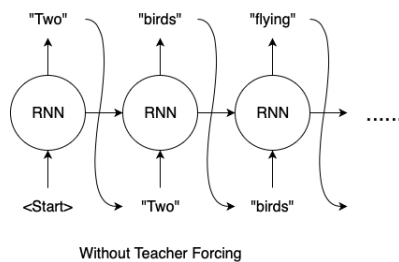
^۵teacher forcing

^۶predict

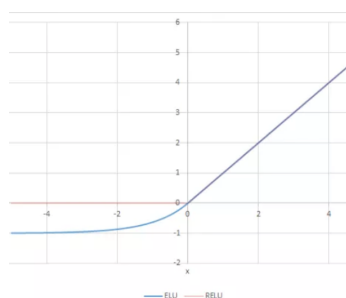
^۷dying problem

^۸over fitting

^۹dropout



شکل ۴-۲: تفاوت یادگیری بازور و یادگیری معمولی [۲۱]



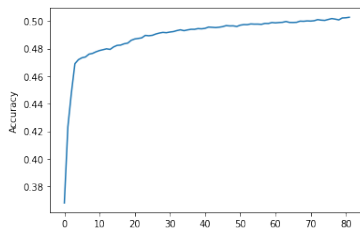
شکل ۴-۳: ضابطه تابع ELU [۱]

شکل ۴-۴ و ۴-۴ ب بوده اند.

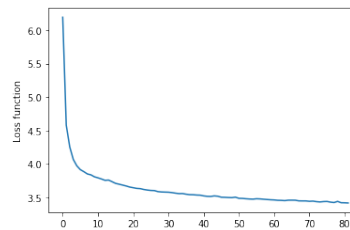
۴-۱-۳ نمونه گیری از مدل

پس از به دست آمدن مدل، روش های مختلفی را میتوان برای تولید متن های جدید با توجه به مدل در پی گرفت:

- تولید مستقیم متن : در این حالت ابتدا توکن «شروع بیت» به مدل داده می شود و همانگونه که توضیح داده شد مدل با محاسبه P احتمال رخ دادن کلمه ی بعدی را به ما خروجی می دهد. در این حالت ما کلمه ای که بیشترین احتمال دارد (argmax) را انتخاب کرده و به عنوان w_1 قرار می دهیم. حال دوباره w_0 و w_1 را به مدل داده و P را محاسبه می کنیم و به همان روال قبلی، کلمه ی بعدی را انتخاب می کنیم و این روند را تا جایی ادامه می دهیم که یا به توکن «پایان بیت» برسیم یا به حداکثر طول مجاز خود برسیم.



(ب) تغییرات تابع صحت



(آ) تغییرات تابع خطا

البته لازم به ذکر است که مشکل این حالت آن است که مدل در شرایط فوق کاملاً به صورت قطعی^۱ رفتار می‌کند و همواره متن‌های مشابهی ایجاد می‌کند در حالی که هدف ما مقداری عملکرد تصادفی را نیز می‌طلبید.

- تولید احتمالی متن : در این روش بر خلاف روش قبل، پس از به دست آمدن P برای انتخاب کلمه‌ی مناسب بعدی، به جای استفاده از تابعی مانند argmax ، با توجه به مدل احتمالی به دست آمده کلمه‌ی بعدی را انتخاب می‌کنیم. بنابراین میتوان اطمینان داشت که علاوه بر بالا بودن احتمال انتخاب کلمه‌ی مناسب، مقداری عملکرد تصادفی نیز وجود خواهد داشت.

- تولید متن به صورت ترکیبی : در این حالت میتوان با انتخاب قوانین^۲ مناسب، از ترکیبی از هر دو روش استفاده کرد. برای مثال میتوان احتمال کلماتی مانند «و» را با توجه به رخ دادن زیاد کم کرد و یا در چند کلمه‌ی ابتدایی بیت تصادفی عملکرد و در کلمات نهایی به صورت قطعی رفتار کرد.

۴-۱-۴ نمونه خروجی در حین یادگیری

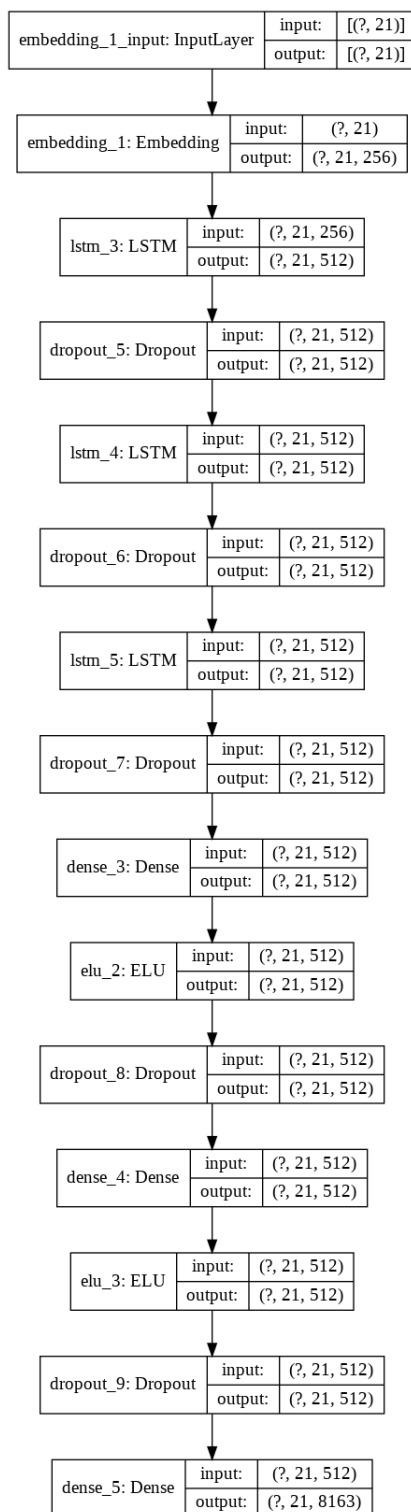
در حین عملیات یادگیری، مدل سعی در دریافت پیچیدگی‌های ساختاری دنباله‌ی ورودی داشته و نهایتاً سعی بر تقلید آن می‌کند. جدول ۴-۱ بیانگر دنباله‌های تقلیدی ایجاد شده توسط مدل در مرحله‌های مختلف یادگیری است.

۴-۱-۵ نمونه خروجی پس از یادگیری

پس از اتمام یادگیری، مدل می‌تواند به طور کامل متن‌های مصنوعی تولید کند. برای مثال متن‌های ۴-۲ تماماً توسط مدل ایجاد شده است.

^۱deterministic

^۲policy



شکل ۴-۵: ساختار مدل شبکه‌ی عصبی بازگشتی مورد استفاده

مرحله‌ی یادگیری	خروجی مدل
۱۰	بسازید از ما کارآزموده وگوی ز خود بازگشتش چهرتست مشت
۲۰	آن آزمایش به رای - - جوشن شاه خود و من
۴۰	همی برآید کنیزک - چندست مردم سوگواری به
۷۰	باشد بساید به سستی - نامورآنگه هر کهان پیش

جدول ۴-۱: خروجی مدل شبکه عصبی بازگشتی در حین یادگیری

خروجی مدل
سخت در نیستان - خروش چو نزیید اسپ پر سور
گریزی نزدیک بر زرد - به نیز تو راستی تخم برنشت
نامه و از کای بی - ببرد بزرگی را خسته را

جدول ۴-۲: خروجی مدل شبکه عصبی بازگشتی پس از یادگیری

۴-۱-۶ بازسازی متن

یکی از کاربردهای مدل‌های تولید متن، استفاده از آن‌ها به عنوان یک مدل بازسازی متن^۱ است. در این حالت، یک متن نیمه کامل که تعدادی از کلمات آن حذف شده است به مدل داده می‌شود و مدل بایستی سعی کند که بهترین کلمه‌ی ممکن را برای جاهای خالی پیدا کرده و جایگذاری کند. در مدل ارائه شده، یک روش برای بازسازی متن به شکل بازسازی کلمه به کلمه خواهد بود. بدین منظور، ابتدا جاهای خالی با توکن‌های ۰ پر شده و نهایتاً متن به دست آمده پس از انجام عملیاتی مانند padding و تبدیل به شکل یک ورودی مناسب، به عنوان ورودی به مدل داده می‌شود. سپس از خروجی به دست آمده، کلماتی که بایستی در جای خالی قرار بگیرند استخراج شده و هر بار تنها یک کلمه جایگذاری می‌شود و پس از جایگذاری یک کلمه، متن جدید دوباره به مدل داده می‌شود و این عملیات به قدری تکرار می‌گردد که تمامی کلمات خالی پر شود. جدول ۴-۳ و ۴-۴ از انجام و نمونه برداری بازسازی متن با مدل مطرح شده به دست آمده است.

در دو جدول مطرح شده، برای انجام عملیات بازسازی متن، ابتدا به صورت تصادفی یک شعر از بین مجموعه‌ی آموزشی انتخاب شده و سپس به صورت تصادفی بین دو تا سه کلمه از آن حذف گردیده است.

¹text imputing

متن اصلی	پس از حذف کلمه‌ها
خورشگر ببردی به ایوان شاه - همی ساختی راه درمان شاه	خورشگر ببردی به ؟ شاه - همی ساختی راه ؟ شاه
چنین گفت ضحاک را ارنواز - که شاه‌ها چه بودت نگویی به راز	چنین گفت ؟ را ؟ - که شاه‌ها چه بودت نگویی به راز
همی داشتم چون یکی تازه سیب - که از باد نامد به من بر نهیب	همی داشتم ؟ یکی تازه سیب - که از باد نامد ؟ من بر نهیب

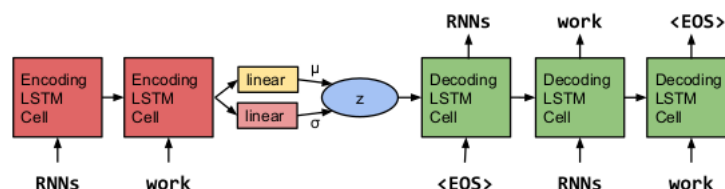
جدول ۴-۳: متن‌های ورودی پس از حذف کلمات از بیت

خروجی مدل
خورشگر ببردی به نمایش شاه - همی ساختی راه نزد شاه
چنین گفت کوس را چمید - که شاه‌ها چه بودت نگویی به راز
همی داشتم خزان یکی تازه سیب - که از باد نامد تبیره من بر نهیب

جدول ۴-۴: خروجی مدل پس از بازسازی متن

۲-۴ مدل خودرمزگذار متغیر

در مقایسه‌ی این مدل با مدل‌های استاندارد بازگشتی مدل زبانی^۱ که اقدام به تولید متن به صورت کلمه به کلمه می‌کنند، مدل‌های خودرمزگذار متغیر قادر به یادگیری پیچیدگی‌های جمله‌ها و در این پروژه پیچیدگی یک بیت هستند. در واقع در مدل‌های استاندارد بازگشتی به دلیل تولید کلمه به کلمه‌ی یک بیت، مدل از داشتن دید کلی در رابطه با بیت باز می‌ماند در حالی که در خودرمزگذارهای متغیر به دلیل انطباق کل جمله به یک فضای نهفته‌ی پیوسته، مدل قادر خواهد بود که دید جامعی از ویژگی‌هایی همانند سبک، موضوع یا وزن بیت‌ها پیدا کند. [۵]



شکل ۴-۶: ساختار مدل‌های خود رمزگذار متغیر [۵]

¹standard recurrent neural network language model

۴-۲-۱ تابع خطا

با توجه به توضیحات داده شده در رابطه با مدل، مدل دارای دو خطای مجزا است که سعی در کمینه کردن مجموع این خطاها دارد. خطای اول مربوط به خطای فضای نهفته است. این خطا که توسط رابطه ی Kulback-Leibler divergence به دست می آیند در واقع بیانگر میزان نزدیک بودن توزیع فضای نهفته نسبت به توزیع گاوسی است. در کل رابطه ی KL سعی بر آن دارد که با توجه به نظریه های تئوری اطلاعات و فضاهای احتمالاتی، فاصله ی دو توزیع از یکدیگر را به صورت شرطی به دست آورد. در واقع در این روش بر خلاف روش های مربوط به پیدا کردن فاصله ی آماری دو توزیع ^۱، اقدام به محاسبه ی واگرایی ^۲ یک توزیع احتمالاتی نسبت به یک توزیع دیگر می کنیم که این مقدار بیانگر آن است که یک توزیع چقدر از یک توزیع دیگر متفاوت است. نهایتاً مقدار KL-Divergence از طریق رابطه ی زیر به دست می آید :

$$D_{kl}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} = - \sum_{x \in X} P(x) \log \frac{Q(x)}{P(x)} \quad (۴-۶)$$

ایده ی پشت رابطه ی فوق به این شکل است که اگر احتمال رخدادی در P بسیار زیاد باشد ولی احتمال رخ دادن همان رخداد در Q کم باشد، یک تفاوت زیادی بین دو توزیع وجود دارد. در حالت متقابل اگر رخدادی در P دارای احتمال رخ دادن کمی باشد ولی همان رخداد در Q احتمال بالایی داشته باشد، بازهم تفاوت بین دو توزیع زیاد خواهد بود ولی نه به اندازه ی حالت قبلی. [۶]

از رابطه ی فوق در حالت توزیع های پیوسته نیز میتوان استفاده کرد که در آن حالت به جای سیگما از انتگرال استفاده خواهد شد.

$$D_{kl}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad (۴-۷)$$

خطای دیگری که در مدل با آن مواجه هستیم خطای بازسازی نمونه است که بیانگر آن است که با دادن یک ورودی به مدل خود رمزگذار، خروجی ما تا چه اندازه نزدیک به ورودی است. در مدل مطرح شده، به ازای هر قدم زمانی ^۳ در ورودی، یک توزیع احتمالی با اندازه ی |V| در خروجی تولید خواهد شد که هر کدام از آنها بیانگر احتمال رخ دادن یک کلمه است (همانند مدل هایی که قبلاً مطرح شد در لایه ی آخر از یک softmax استفاده شده است) و بنابر این خطای بازسازی توسط رابطه ی زیر به دست خواهد آمد :

$$\sum_{maxlength} -\log(q(x)) \quad (۴-۸)$$

که در آن x کلمه ای است که در به عنوان i مین کلمه ی بیت در داده های ورودی رخ داده است.

¹statistical distance

²divergence

³time step

در مدل خودرمزگذار متغیر پیاده‌سازی شده، به عنوان اولین لایه‌ی پنهان از یک لایه‌ی word embedding با اندازه‌ی ۲۵۶ استفاده شد. همچنین پس از آن دو لایه‌ی LSTM به صورت پشت‌پشته شده استفاده شد که هر کدام از آن‌ها دارای ۵۱۲ عدد گره بودند. برای جلوگیری از بیش‌برازش لایه‌های LSTM به کار رفته، از خطای regularization با مقدار 0.001 استفاده شده است. پس از آن نیز یک لایه‌ی تماماً متصل استفاده شد که دارای ۵۱۲ عدد گره است. نهایتاً دو لایه‌ی مجزای کاملاً متصل یکی به عنوان لایه‌ی میانگین فضای نهفته و یکی به عنوان لایه‌ی واریانس فضای نهفته از لایه‌ی تمام متصل قبلی گرفته شده است که از ترکیب این دو لایه، نهایتاً فضای نهفته با ابعاد ۵۱۲ به دست آمد.

به دلیل نیاز به پردازش‌های سنگین، برای یادگیری تنها از یک چهارم داده‌های یادگیری موجود استفاده شد که نهایتاً دارای ۲۴۸۰۴ مصرع یا معادلاً ۱۲۴۰۲ بیت بود و در ابیات طولانی‌ترین بیت دارای طول ۲۱ کلمه بود. همچنین همانند قبل در لایه‌های کاملاً متصل، از تابع ELU به عنوان تابع فعال ساز استفاده شده است. برای بررسی صحت یادگیری مدل در حین یادگیری نیز مقدار 0.2 از داده‌های یادگیری به عنوان داده‌ی بررسی صحت^۱ استفاده شده است.

سپس از فضای نهفته دوباره با استفاده از یک لایه‌ی LSTM یک سری زمانی جدیدی ساخته شده است که نهایتاً به یک لایه‌ی تماماً متصل با اندازه‌ی ۱۷ متصل گردیده است که فضای احتمالی رخ‌دادن کلمات در بیت را مشخص می‌کند. در حین یادگیری از تکنیک KL-Annealing برای محاسبه‌ی خطا استفاده شده است. درواقع با توجه به این موضوع که خطای خودرمزگذار از دو خطای مجزای خطای بازسازی و خطای KL تشکیل شده است، هنگام یادگیری مشاهده شد که میزان خطای KL بسیار بیشتر از خطای بازسازی بوده است. بنابراین قسمت عمده‌ای از خطای مدل توسط خطای KL ایجاد شده و این خطا بر خطای بازسازی غلبه کرده و مدل در طول یادگیری تمرکز خود را بیشتر بر روی کم کردن خطای KL متمرکز کرده و سعی می‌کند با کاهش آن بهینه رفتار کند. در نتیجه مدل خودرمزگذار متغیر به عنوان یک خودرمزگذار عملکرد خوبی از خود نشان نخواهد داد و در بازسازی جملات ورودی به مشکل برخورد خواهد خورد.

در روش KL-Annealing مدل برای خطای KL یک ضریب در نظر می‌گیرد که این ضریب در ابتدای فرآیند یادگیری برابر صفر است و مدل در ابتدای یادگیری سعی در کاهش مقدار خطای بازسازی می‌کند. سپس پس از گذراندن تعدادی دوره‌ی^۲ یادگیری، ضریب مربوط به خطای KL به مرور افزایش می‌یابد تا نهایتاً به ۱ برسد.

^۱validation set

^۲epoch

در پیاده‌سازی انجام شده، مدل در ۲۰ دوره‌ی ابتدایی دارای ضریب KL برابر صفر بود و سپس در هر دوره، ضریب با قدم‌های 0.01 افزایش یافت و مشاهده شد که این تکنیک در همگرا شدن مدل به نتیجه‌ی مطلوب تأثیر بسزایی داشت. همچنین میتوان میزان دقت مدل در بازسازی ابیات ورودی داده شده به مدل در دوره‌های مختلف یادگیری را در جدول ۴-۵ مشاهده کرد.

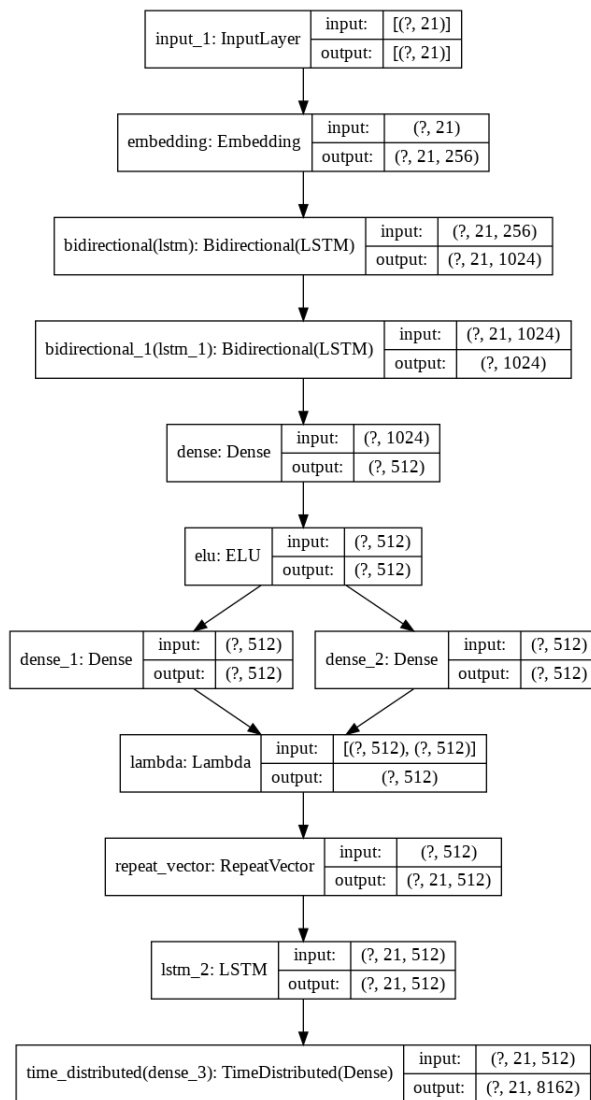
دوره	ورودی	خروجی
۱	در دژ بکنند ایرانیان - بغارت بیستند یکسر میان	به - - - - -
۱۳	نخست آفرین کرد بر دادگر - کزو دید نیروی و فر و هنر	گفت به و شاه - که و و و و و آب
۴۸	مرا بارگه زان تو برترست - هزاران هزارم فزون لشکرست	که بازماند و آزادگان - بدین شهریاری همم میوش
۸۰	یکی تاج پرگوهر شاهوار - دو تا یاره و طوق با گوشوار	زابلستان طبقها فروریختند - ابا جوشن و و و و گوشوار
۱۴۰	ور ایدونک آید ز اختر پسر - ببندش بیازو نشان پدر	ایدونک کیم من اختر پسر - ببندش بیازو نشان بز
۲۰۰	ابا خویشتن بر یکی پر من - خجسته بود سایه فر من	هوش بدادی یکی به من - خجسته بود سایه فر من
۴۰۰	همه لشکر از جای برخاستند - درفش فریدون بیاراستند	لشکر بر ز جای برخاستند - درفش فریدون بیاراستند
۴۸۳	بدرید کتفش بدنجان چو شیر - برو خیره شد پهلوان دلیر	کتفش بدنجان به شیر - برو خیره کار پهلوان دلیر

جدول ۴-۵: ورودی و خروجی مدل در دوره‌های مختلف

۴-۲-۳ نمونه‌گیری از مدل

همانطور که قبلاً مطرح شد، پس از اتمام عملیات یادگیری مدل، قسمت رمزگشای مدل خودرمزگذار متغیر به صورتی یادگرفته است که با گرفتن یک ورودی از فضای نهفته، خروجی‌های مدنظر ما را ایجاد کند. با توجه به اینکه در این پروژه خروجی مدنظر ما بیت‌های شعر است، مدل با دریافت یک نمونه از فضای نهفته (که در طول یادگیری سعی شده است دارای توزیع نزدیک به توزیع گاوسی باشد) یک دنباله‌ای از فضای احتمالاتی بازمی‌گرداند که i مین مقدار این دنباله دارای $|V|$ احتمال مختلف است که مقدار t ام آن بیانگر احتمال رخ دادن t مین کلمه در V به عنوان i مین کلمه در بیت است (و بدین منظور همانند قبل از تابع softmax استفاده شده است).

بنابراین برای ایجاد بیت‌های جدید، کافی است ابتدا به صورت تصادفی از توزیع گاوسی نمونه‌برداری کرده و به عنوان ورودی به قسمت رمزگشای مدل خودرمزگذار متغیر داده و سپس از خروجی ایجاد شده، برای انتخاب



شکل ۴-۷: ساختار مدل خودرمزگذار متغیر مورد استفاده

t مین کلمه‌ی بیت، کلمه‌ای که بیشترین احتمال را دارد انتخاب کنیم.

البته روش دیگر انتخاب t مین کلمه‌ی بیت می‌تواند آن باشد که با توجه به توزیع احتمالی ایجاد شده توسط مدل، کلمه‌ی t ام انتخاب گردد.

نهایتاً پس از یادگیری مدل، تعدادی از نمونه‌های ایجاد شده توسط مدل در جدول ۴-۶ آمده‌است. همانگونه که مشاهده می‌شود در این مدل خروجی‌های به دست آمده در اکثر موارد دارای قافیه است و وزن‌های به کار رفته تا حد مناسبی رعایت شده است.

۴-۲-۴ بازسازی متن

همانند مدل قبلی، از مدل فعلی نیز میتوان برای عملیات‌هایی مانند بازسازی متن استفاده کرد. در مدل‌های خودرمزگذار متغیر، با توجه به ساختار مدل و با توجه به این نکته که قسمت رمزنگار یک ورودی را به یک

خروجی مدل
جنب جنبان خواب بر و تار - یکی بند بر آمد و بند غار
چنین داستان زد خرد نسپرد - که بندق مدار ارچه خردست خوار
آن رخ را کشته جنگ جوی - یکی شاه بی اندر آن بروی
مرا و اندر هزار پایه نیست - بجز خاک تیره مرا جای نیست
مازندران یاد اکنون امید - نجست آن دلیران دیوان نبرد
و بیگنه باد سرم - بدان دریا جگر برنهند افسرم
گوید سپس بودی نماند دهم - ازو مازندران سرفرازی دهم

جدول ۴-۶: خروجی مدل خودرمرگذار متغیر پس از یادگیری

فضای نهفته‌ی نرمال شده انطباق می‌دهد، میتوان انتظار داشت که اگر یک بیت که تعدادی از کلمه‌های آن حذف شده است را به این مدل دهیم (و برای برهم نخوردن نظم بیت میتوان به جای کلمه‌های حذف شده یک کلمه با ارزش ۰ قرار داد) پس از انجام عملیات نگاشت، خروجی به دست آمده نقطه‌ای نزدیک به بیت اصلی در فضای نرمال باشد. بنابراین با دادن این فضای نهفته‌ی جدید به شبکه‌ی رمزگشا، میتوان انتظار داشت که مدل کلمه‌های حذف شده را با نزدیک‌ترین کلمه‌ی ممکن پر کند و به دلیل آنکه فضای نهفته نمایانگر ویژگی‌های کل بیت است، میتوان انتظار داشت که ویژگی‌هایی مانند وزن و قافیه تا حد مناسبی همچنان رعایت شود. در حالتی که تعداد کلمه‌های حذف شده در متن بیش از یک مورد باشد نیز، به صورت مکرر در هر تکرار یک کلمه‌ی حذف شده جایگزین می‌گردد. جدول ۴-۷ و جدول ۴-۸ از انجام و نمونه برداری بازسازی متن با مدل مطرح شده به دست آمده است. در دو جدول مطرح شده، برای انجام عملیات بازسازی متن، ابتدا به صورت تصادفی یک شعر از بین مجموعه‌ی آموزشی انتخاب شده و سپس به صورت تصادفی بین دو تا سه کلمه از آن حذف گردیده‌است.

همچنین به طور مشابه، از انجام عملیات فوق بر روی داده‌های آزمایش جدول‌های ۴-۹ و ۴-۱۰ به دست آمده‌است.

در انجام عملیات‌های بازسازی متن در مدل‌های استاندارد بازگشتی که قبلاً بررسی شد، به دلیل آنکه یک بیت به صورت کلمه به کلمه بررسی می‌شد و هر کلمه، وابستگی زیادی به کلمه‌های اطرافش داشت، در صورتی که دو کلمه که حذف شده باشند در جوار یکدیگر باشند و یا یکی از کلمه‌ها در نقطه‌ای مانند ابتدای بیت رخ

متن اصلی	پس از حذف کلمه‌ها
پرستنده ای کش به بر داشتی - زمین را به پی هیچ نگذاشتی	پرستنده ای ؟ به ؟ داشتی - ؟ را به پی هیچ نگذاشتی
چو آمد بنزدیک شاه آن سپاه - فریدون پذیره بیامد براه	چو آمد بنزدیک شاه آن ؟ - فریدون ؟ بیامد ؟
چو خسرو بران گونه آمد ز راه - چنین بازگشت از پذیره سپاه	چو خسرو بران گونه آمد ز ؟ - ؟ بازگشت از پذیره سپاه

جدول ۴-۷: متن‌های ورودی پس از حذف کلمات از بیت

خروجی مدل
پرستنده ای کش به بر داشتی - زمین را به پی هیچ نگذاشتی
چو آمد بنزدیک شاه آن دلیر - فریدون ناگاه بیامد شیر
چو خسرو بران گونه آمد ز راه - چنین بازگشت از پذیره سپاه

جدول ۴-۸: خروجی مدل پس از بازسازی متن

متن اصلی	پس از حذف کلمه‌ها
که خسرو ز توران به ایران رسید - نشست از بر تخت کو را سزید	که خسرو ز توران به ؟ رسید - نشست ؟ بر تخت کو را سزید
بدو گفت کاووس کین کار تست - که بیدار دل بادی و تن درست	بدو گفت کاووس کین کار تست - که بیدار دل ؟ ؟ تن درست
سپهدار سهراب نیزه بدست - یکی بارکش باره ای برنشست	سپهدار سهراب نیزه بدست - یکی ؟ ؟ ای برنشست
ز بهر بزرگان ایران زمین - برآرامش این رنج کردی گزین	ز ؟ ؟ ایران زمین - برآرامش این رنج کردی گزین

جدول ۴-۹: متن‌های ورودی پس از حذف کلمات از بیت

داده باشد، مدل برای حدس زدن کلمه‌ی مناسب برای پر کردن جای خالی با مشکل مواجه می‌شد. حال آنکه در مدل‌های خودرمنگار متغیر به دلیل استفاده از فضای نهفته و ویژگی‌های این فضا، این مسئله تاثیر بسیار کمتری در عملکرد مدل دارد.

همچنین در یکی از مثال‌های مطرح شده در جدول ۴-۸ مثالی وجود دارد که به شرح زیر است :

چو آمد بنزدیک شاه آن دلیر - فریدون ناگاه بیامد شیر

در اینجا کلمات قافیه سپاه و براه با کلمات دلیر و شیر جایگزین شده است که بنظر می‌رسد انتخاب مناسبی باشد زیرا علاوه بر رعایت قافیه و وزن، معنی شعر همچنان درست است. مورد دیگری که نیز لازم به توجه دارد این

خروجی مدل
که خسرو ز توران به خواب رسید - نشست موبد بر تخت کو را سزید
بدو گفت کاووس کین کار تست - که بیدار دل بادی و تن درست
سپهدار سهراب نیزه بدست - یکی نیرو باره ای برنشت
ز چندان بزرگان ایران زمین - برآرامش این رنج کردی گزین

جدول ۴-۱۰: خروجی مدل پس از بازسازی متن

است که با توجه به الگوریتم ارائه شده برای بازسازی متن، به جای هر کلمه‌ی حذف شده، یک کلمه بازسازی می‌گردد، حال آنکه در برخی از موارد برای برهم نخوردن وزن، ممکن است نیاز باشد که دو یا تعداد بیشتری کلمه جایگزین یک کلمه‌ی حذف شده گردد.

نهایتاً مدل مورد بحث در زمینه‌ی بازسازی کلمات دارای دقت 0.53 بود. برای محاسبه‌ی دقت کلمه‌ی انتخاب شده در این مدل، پس از حذف کلمه‌ی مد نظر از متن و دادن متن به مدل، از خروجی به‌دست آمده احتمال انتخاب کلمه‌ی صحیح را محاسبه می‌کنیم و نهایتاً بین تمام کلمات حذف شده و احتمال‌های آن‌ها میانگین گرفته می‌شود. البته لازم به ذکر است که با توجه به تعداد زیاد کلمه‌ها در دایره‌ی لغات مدل و همچنین این موضوع که مجموع احتمالات تمام کلمات دایره‌ی لغات بایستی برابر ۱ باشد، میتوان نتیجه گرفت که مدل همواره احتمال بسیار زیادی را برای کلمه‌ی درست و صحیح در نظر می‌گیرد.

۴-۳ مدل‌های تولیدکننده‌ی خصمانه

با توجه به ویژگی‌های مطرح شده در رابطه با شبکه‌های مولد خصمانه در فصل ۲ به نظر می‌رسد که استفاده از مدل‌های تولیدکننده‌ی خصمانه یکی از روش‌های مناسب برای تولید متن باشد. حال آنکه برای استفاده از شبکه‌ی فوق در این مسأله مشکلات زیادی همراه است. مهم‌ترین مشکل نیز مشتق ناپذیر بودن توابعی مانند argmax است. درواقع در قسمت تولیدکننده‌ی شبکه‌ی فوق، خروجی ایجاد شده همواره یک توزیع احتمالاتی از رخ دادن کلمه‌ها است و برای ایجاد نمونه‌ی متنی که بایستی به عنوان ورودی به مدل تمیزدهنده داده شود، بایستی که از توزیع احتمالاتی فوق با استفاده از عملیاتی مانند argmax یک کلمه به عنوان نمونه انتخاب شود و با انتخاب کلمه به کلمه نهایتاً یک جمله ساخته شود. حال پس از دادن این ورودی به مدل تمیزدهنده، استفاده از خروجی این مدل به عنوان فیدبک امکان‌پذیر نیست زیرا در لایه‌ی آخر مدل تولیدکننده از تابعی مانند argmax استفاده

شده است که حال با دریافت فیدبک در حالت عادی نمی تواند با انجام عملیاتی مانند مشتق گیری عملیات انتشار عقب گرد^۱ را انجام دهد.

مدل مولد خصمانه، با ارائه ی روشی جدید برای یادگیری و ایجاد یک مدل زبانی عصبی^۲ سعی بر غلبه کردن بر مشکلاتی که مدل های قبلی داشتند می کند. به طور مثال در مدل های خودرمنگار متغیر، علی رغم اینکه خروجی های به دست آمده تا حد خوبی دقت مناسب را دارا بودند، ولی به دلیل اینکه فضای نهفته ممکن است به توزیع یکنواخت و نرمالی که مدنظر است دست نیابد (به دلیل پیچیدگی ساختارها و جملات زبانی)، خروجی های به دست آمده همچنان تا سطح مطلوب و مورد نظر فاصله ی زیادی دارند. [۱۴] در این حالت به دلیل آنکه توزیع فضای نهفته ی به دست آمده دارای یکنواختی لازم نمی باشد، برخی از نقاط این فضا ممکن است خروجی های خوبی تولید کند در حالی که برخی از نقاط توزیع فضای نهفته به درستی نگاشت نشوند و این اتفاق باعث می شود که هنگام نمونه برداری از این توزیع، در برخی از حالات خروجی های به دست آمده پس از رمزگشایی، خروجی های معناداری نباشد. [۱۴]

همچنین مدل های مولد خصمانه، به دلیل ساختار خود سعی در بررسی و تولید یک نمونه در سطح یک جمله می کنند و بر خلاف مدل های استاندارد شبکه های عصبی بازگشتی که قبلا مطرح شد، خروجی به صورت کلمه به کلمه ایجاد نمی شود و این خود یک مزیت دیگر است زیرا در این حالت مدل می تواند پیچیدگی های در سطح جمله مانند وزن، قافیه و معنی کلی را یاد بگیرد که باعث پیوستگی بیشتر جملات ایجاد شده خواهد شد.

برای حل کردن مشکل گسسته بودن فضای کلمه ها در کار با متن برای مدل های مولد خصمانه راهکارهای متعددی می توان در پیش گرفت. برای مثال میتوان از روش های یادگیری تقویتی^۳ به همراه مدل های مولد خصمانه استفاده کرد. [۲۲]

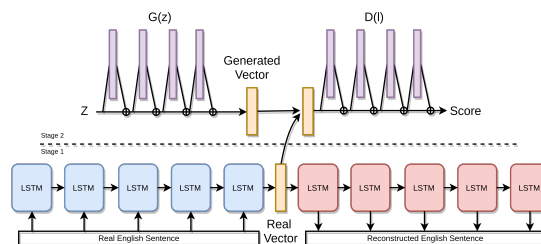
یکی دیگر از راه های فایق آمدن بر مشکل مطرح شده، استفاده از یک نگاشت است. بدین منظور ابتدا تمام داده های ورودی اصلی (در این مسأله بیت ها) به یک فضای پیوسته نگاشت شده و سپس شبکه ی تولیدکننده ی خصمانه تلاش می کند که با استفاده از داده های نگاشت شده، مدل های تولیدکننده و تمیزدهنده را یادگیری کند. در مرحله ی نهایی نیز پس از آنکه یادگیری مدل تولیدکننده پایان یافت، با استفاده از آن داده های جدید تولید شده و سپس داده ها دوباره از فضای پیوسته به فضای بیت ها و متن نگاشت وارون می شوند.

برای انجام نگاشت متن به فضای پیوسته، همانطور که در قسمت های قبل مطرح شد یکی از روش ها می تواند استفاده از مدل های خودرمنگار باشد. بدین منظور یک مدل خودرمنگار ابتدا بر روی داده ها یادگیری شده که

¹back propagation

²neural language model

³reinforcement learning



شکل ۴-۸: استفاده از خودرمزگذار در مدل‌های مولد خصمانه [۷]

داده‌ها را از فضای متن به فضایی پیوسته نگاشت کند و سپس با استفاده از آن داده‌های ورودی ابتدا نگاشت شده و سپس از داده‌های نگاشت شده برای یادگیری مدل تولیدکننده‌ی خصمانه استفاده می‌شود. [۷]

۴-۳-۱ پیاده‌سازی

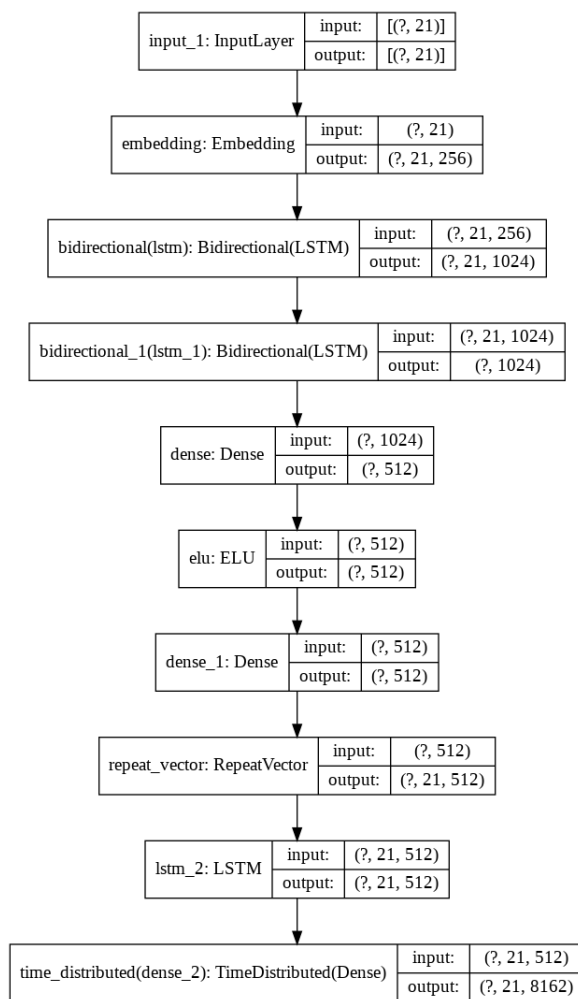
در قسمت مربوط به پیاده‌سازی، ابتدا یک مدل خودرمزگذار برای انجام نگاشت فضای بیت‌ها به یک فضای پیوسته استفاده شد. در مدل پیاده‌سازی شده، به عنوان اولین لایه‌ی پنهان از یک لایه‌ی word embedding با ابعاد ۲۵۶ استفاده شده است. سپس خروجی لایه‌ی مطرح شده به یک لایه‌ی LSTM دوطرفه وارد شده که این لایه نیز دارای ۵۱۲ گره می‌باشد. خروجی این لایه دوباره به شکل یک دنباله به یک لایه‌ی LSTM دیگر با ابعاد ۵۱۲ گره داده شده است. همچنین در هر دولایه‌ی عصبی بازگشتی مطرح شده، از تابع Relu به عنوان تابع فعال‌ساز استفاده شده است و برای جلوگیری از بیش‌برازش لایه‌های مطرح شده از مقادیر regularization نیز استفاده شده است.

سپس از آخرین لایه‌ی LSTM یک ورودی یک بعدی گرفته شده و به یک شبکه‌ی تماماً متصل با ابعاد ۵۱۲ گره داده شده و برای تابع فعال‌ساز نیز از تابع ELU در این لایه استفاده شده. در آخرین قدم نیز برای ایجاد فضای نهفته در خودرمزگذار، از یک لایه‌ی تماماً متصل دیگر با ابعاد ۵۱۲ گره استفاده شده است. نهایتاً مدل پیاده شده را در تصویر ۴-۹ می‌توانید مشاهده کنید.

همچنین توانایی بازسازی^۱ مدل خودرمزگذار در طول فرآیند یادگیری در دوره‌های مختلف در جدول ۴-۱۱ آمده است.

پس از آماده‌شدن مدل خودرمزگذار و انجام فرآیند نگاشت، شبکه‌ی مولد خصمانه بر روی فضای نگاشت شده اقدام به عملیات یادگیری کرد. مدل مورد استفاده شده در قسمت مدل تولیدکننده دارای ۵ لایه‌ی تمام متصل عصبی بوده که در تمام آن‌ها از ELU به عنوان تابع فعال‌ساز استفاده شده است. همچنین ابعاد آن‌ها به ترتیب ۱۲۸، ۲۵۶، ۲۵۶، ۲۵۶ و ۵۱۲ بوده است که در آخرین لایه نهایتاً خروجی یک داده‌ی جعلی شبیه به داده‌های نگاشت شده به فضای نهفته است.

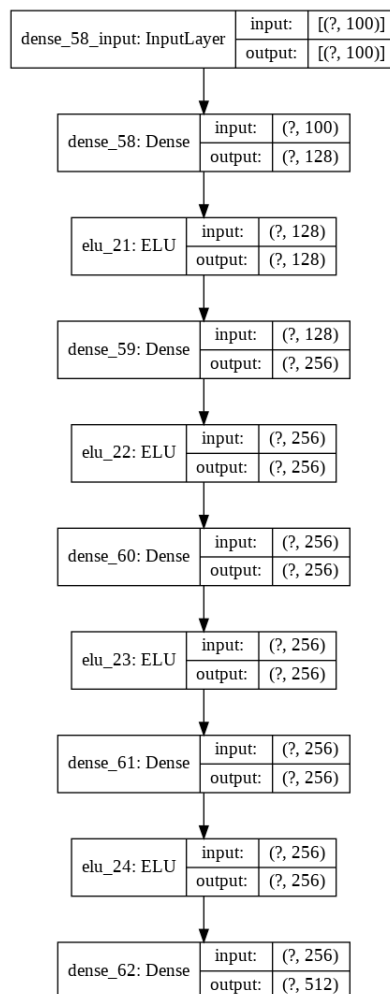
^۱reconstruct



شکل ۴-۹: ساختار مدل خودرگرگذار

دوره	ورودی	خروجی
۲	چو دژبان چنین گفتھا را شنید - همان مهر انگشتی را بدید	به
۳	چو آمد ز پهلوی برون پهلوان - همی نامزد کرد جای گوان	گفت به و و - - و و به انجمن
۶۰	سوی شهر ایران نهادند روی - فرنگیس و شاه و گو جنگ جوی	به به نزدیک شاه - پراز و و شاه روی
۹۰	بیاراست کشتی به چیزی که داشت - ز باد هوا بادبان برگذاشت	را بر به را روی - به گشت و قعر سمند
۳۰۰	چنان دید رودابه را در نهان - کجا نشنود پند کس در جهان	پس رودابه را در نهان - کجا نشنود پند کس در جهان
۴۶۰	بودند یک هفته زین گونه شاد - ز شاهان گیتی گرفتند یاد	بودند آمد هفته زین گونه شاد - ز شاهان گیتی گرفتند یاد

جدول ۴-۱۱: ورودی و خروجی مدل در دوره‌های مختلف



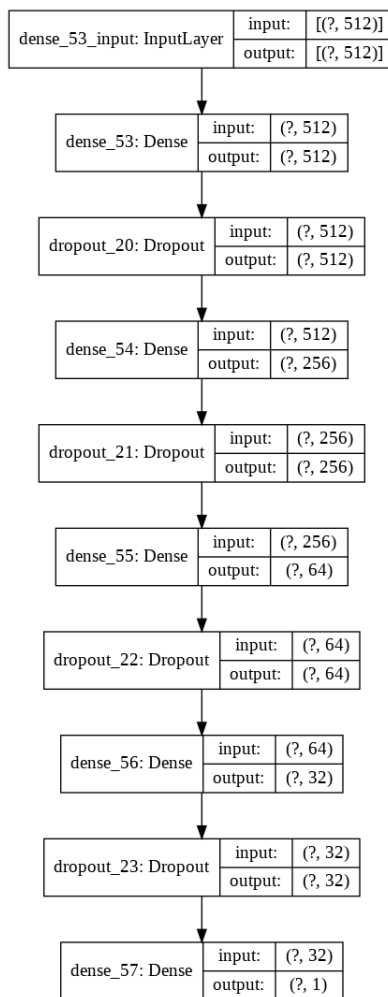
شکل ۴-۱۰: ساختار مدل تولیدکننده

برای مدل تمیزدهنده نیز به شکل مشابه از ۵ لایه‌ی تماماً متصل استفاده شده است که دارای ابعاد ۵۱۲، ۲۵۶، ۶۴، ۳۲، ۱ است که در ۴ لایه‌ی اول از تابع relu به عنوان فعال‌ساز استفاده شده است و در آخرین لایه برای تشخیص جعلی بودن یا نبودن داده‌ی ورودی از فعال‌ساز sigmoid استفاده شده است. نهایتاً شبکه‌ی استفاده شده همانند شکل ۴-۱۱ است.

همچنین برای همگرایی سریع‌تر مدل، از weight clipping در هنگام یادگیری استفاده شده است و در هر دوره‌ی یادگیری، مدل تمیزدهنده ۳ دوره‌ی یادگیری را طی کرده و سپس مدل مولد اقدام به یادگیری می‌کند.

۴-۳-۲ نمونه‌گیری از مدل

پس از اتمام فرایند یادگیری، با دادن نویز و ورودی‌های رندوم به مدل تولیدکننده، میتوان انتظار داشت که مدل خروجی‌هایی ایجاد کند که هنگام عبور دادن آن‌ها از قسمت رمزگشای مدل خودرمزگذار استفاده شده، نهایتاً به یک بیت تبدیل شود. تعدادی از خروجی‌های به دست آمده از مدل پیاده‌سازی شده بر روی داده‌های اشعار



شکل ۴-۱۱: ساختار مدل تمیزدهنده

فردوسی در جدول ۴-۱۲ آمده است.

یکی از به در زد پر نیم - دو دستش ترازو و از و سیم
همه شدن ایران افراسیاب - می و مشک به چرم شتاب
بنالید سودابه به او بلند - ز جنگ جنگ او شادمان
بتر آن مر پیشه او به کوه - جهان را نمانند بی کدخدای
گزاینده کاری بد آمد به پیش - کز اندیشه آن دلم گشت ریش

جدول ۴-۱۲: خروجی های ایجاد شده توسط مدل تولیدکننده

فصل پنجم

نتیجه‌گیری

پس از بررسی مدل‌های پیاده‌سازی شده، میتوان به این نتیجه رسید که استفاده‌ی مستقیم و ساده از LSTM ها و شبکه‌های عصبی بازگشتی، قادر به پاسخ‌گویی و ارضای نیازهای مطرح شده نمی‌باشد. درواقع در این مدل‌ها، شبکه سعی می‌کند که همواره با ایجاد یک مدل خودمختار^۱ با توجه به کلمه‌های دیده‌شده، کلمه‌ی بعدی و آینده را حدس بزند. در این حالت مدل از داشتن دید جامع و کلی بازمانده و پیچیدگی‌هایی همانند وزن و قافیه را از دست می‌دهد. همچنین با وجود آنکه کلمات در همسایگی یکدیگر دارای رابطه‌ی معنایی مناسب هستند، ولی عبارات به دست آمده در کل در سطح بیت فاقد معنای مدنظر است. این مشکل خود ناشی از چند دلیل می‌تواند باشد. اول آنکه مدل در حین یادگیری، وزن بیشتری برای کلمه‌های همجوار قائل شده و هنگام پیش‌بینی یک کلمه، کلمه‌های نزدیک‌تر به آن کلمه دارای وزن بیشتری خواهند بود. دوم آنکه به دلیل vanishing gradient مدل پس از مدتی، کلمات دورتر و قدیمی‌تر را فراموش می‌کند. نهایتاً آنکه تکرار کلماتی مانند و، یا و ... در متن‌ها از اکثر کلمات دیگر بیشتر است که ناخودآگاه مدل به سمت تکرار آن‌ها متمایل^۲ خواهد شد. برای رفع مشکل متمایل شدن مدل به سمت کلمات خاص، با استفاده از لایه‌های دورریز^۳ سعی بر آن شد که از تمایل

^۱Autoregressive

^۲bias

^۳dropout

مدل بر روی کلمات مذکور جلوگیری گردد. همچنین استفاده از تکنیک‌هایی مانند attention و استفاده از مدل‌های sequence-to-sequence می‌تواند به بهبود مدل در جهت داشتن دید جامع‌تری نسبت به کل جمله کمک کند.

از طرف دیگر در مدل پیاده‌سازی شده با استفاده از خودرمزگذارهای متغیر، به دلیل استفاده از لایه‌ی فضای نهفته و نرمال‌سازی این فضا و تشابه‌های موجود با روش‌های استنتاج بیزی^۱، مدل توانست که به دید جامع‌تری نسبت به کل بیت دست پیدا کرده و در موارد بسیاری، وزن و قافیه تا حد مناسبی رعایت شده‌بود. همچنین جملات به‌دست آمده از نظر ساختاری نیز تا حد مناسبی شبیه به مجموعه‌ی آموزشی بوده و مدل توانسته بود که پیچیدگی‌های داده‌های آموزشی را تا حد مناسبی تقلید کند. از این رو میتوان انتظار داشت که با انجام پردازش‌های بیشتر و تنظیم مناسب‌تر مدل، به نتایج بهتری دست‌یافت. نهایتاً شایان ذکر است که مدل به دست‌آمده توسط خودرمزگذار متغیر، علاوه بر ایجاد متن‌ها و ابیات مصنوعی، قادر به تولید یک مدل بازسازی متن با دقت بالا نیز بوده است.

یکی از مشکلاتی که مدل خودرمزگذار متغیر دچار آن شده‌است، توزیع غیرمتقارن نگاشت داده‌های ورودی به فضای نهفته است. در این حالت اکثر داده‌های ورودی به قسمت خاصی از فضای نهفته نگاشت شده و قسمت اعظمی از فضای نهفته، هیچ ورودی به آن نگاشت نمی‌شود که این غیرمتقارن بودن نگاشت داده‌ها، خود باعث آن می‌شود که در برخی از حالت‌ها هنگام نمونه‌گیری از فضای نهفته، خروجی به دست‌آمده فاقد ویژگی‌های موردنظر باشد. در راستای حذف این مشکل، از شبکه‌های تولیدکننده‌ی خصمانه در این پروژه استفاده شد و سعی شد که ابتدا برای حل مشکل گسسته بودن داده‌ها و کلمه‌ها و حل این موضوع که شبکه‌های مولد خصمانه در هنگام مواجه با داده‌های گسسته دچار مشکل می‌شوند، از یک خودرمزگذار برای نگاشت ابیات به یک فضای پیوسته استفاده‌شود و نهایتاً از شبکه‌ی مولد برای تولید داده‌های مصنوعی بر روی این فضای پیوسته استفاده شد. داده‌های تولید شده و خروجی به دست آمده همچنان سعی بر آن دارد که وزن و قافیه و ویژگی‌های یک بیت را به درستی رعایت کند ولی به نظر می‌رسد که خروجی‌های به دست آمده نسبت به مدل خودرمزگذار متغیر ضعیف‌تر باشد. یکی از راه‌های بهبود مدل می‌تواند استفاده از توزیع‌های دیگر شبکه‌های مولد خصمانه مانند wgan باشد. مشکل دیگری که در این پروژه با آن مواجه هستیم عدم وجود معیار مناسب برای سنجش کیفیت خروجی‌های به دست آمده است. با وجود آنکه در حین یادگیری و مقایسه‌ی بین دو مدل، سنجش کیفیت و برتر بودن یک خروجی از دیگری، برای انسان واضح می‌تواند باشد، ولی همچنان تعریف مناسب و دقیقی از معیار سنجش کیفیت برای خروجی‌های به دست آمده موجود نیست. در راستای بررسی سطح کیفیت خروجی‌های به دست

¹Bayesian inference

آمده، یک روش می‌تواند آن باشد که مدل دیگری بر روی داده‌های آموزشی و اشعار یادگیری شود که توانایی تشخیص بیت‌های جعلی از بیت‌های حقیقی را داشته باشد و با استفاده از این مدل تمیزدهنده، میزان دقت یک مدل دیگر برابر با درصد از خروجی‌های مصنوعی تولید شده‌ای باشد که بتواند مدل تمیزدهنده را فریب دهد. قابل ذکر است که در این حالت ما با یک مسأله‌ی one class classification روبرو هستیم که تنها با داشتن داده‌های یک دسته‌ی آموزشی، بایستی مدلی برای تشخیص داده‌های غیرنرمال^۱ و ناهنجاری^۲ ایجاد کنیم که توضیحات و ایجاد چنین مدلی خارج از موضوع این پروژه است.

^۱abnormal

^۲anomaly

فصل ششم

کارهای آینده

در پروژه‌ی انجام شده و مطرح شده، با بررسی مدل‌ها و روش‌های مختلف هوش مصنوعی و پردازش زبان‌های طبیعی، اقدام به بررسی تکنیک‌ها و مدل‌های تولید متن نمودیم. حال آنکه این پروژه تنها صرفاً شروع و مقدمه‌ای بر این زمینه بوده است و همچنان برای رسیدن به هدف اصلی که تولید مدلی ایده‌آل با ویژگی‌های مطرح شده است فاصله‌ی زیادی باقی مانده است. در ادامه برای بهبود خروجی‌های به دست آمده، میتوان از مدل‌ها و تکنیک‌های متنوع دیگری استفاده کرد که بررسی و انجام تک تک آن‌ها از حوصله‌ی این پروژه خارج بوده و تنها به آن‌ها به عنوان برنامه و نقشه‌راهی برای آینده‌ی این پروژه اشاره می‌کنیم.

یکی از راه‌های دیگر تولید متن‌های مصنوعی، می‌تواند استفاده از روش‌های یادگیری تقویت شده^۱ باشد. در این مدل‌ها با تکیه بر روش‌های برپایه‌ی قانون^۲ میتوان انتظار داشت مدلی تولید شود که همانند انسان با تکیه بر روش‌های یادگیری و خطا و دریافت پاداش و مجازات، پیچیدگی‌های زبان را درک کرده و اقدام به تولید متن‌های مصنوعی کند.

یکی دیگر از راه‌های افزایش کیفیت خروجی‌های به دست آمده، بهبود مدل‌های مطرح شده در این پروژه است. بدین منظور میتوان با استفاده از تکنیک‌های مختلف، پارامترهای مدل‌های مختلف را به منظور تولید

¹reinforcement learning

²policy based models

خروجی‌های بهتر تنظیم کرد. همچنین با توجه به این موضوع که در تمامی مدل‌های مطرح شده از ساختارهای LSTM به عنوان چهارچوب اصلی استفاده شده‌است، میتوان انتظار داشت که با استفاده از تکنیک‌ها و روش‌هایی مانند لایه‌های attention کیفیت خروجی‌ها افزایش یابد. در این حالت لایه‌ی attention کمک می‌کند که لایه‌ی LSTM در هر مرحله هنگام پیش‌بینی یک کلمه، دید جامع‌تری از کل متن و جمله داشته‌باشد و از مشکل دادن وزن‌های زیاد به کلمات همجوار در حین پیش‌بینی یک کلمه جلوگیری کرده و مدل قادر به توزیع مناسب‌تر وزن‌ها بر روی کلمات هنگام پیش‌بینی یک کلمه خواهد بود. همچنین با توجه به این موضوع که یکی از مشکلات اصلی در این پروژه توان پردازشی بوده است و یادگیری لایه‌های LSTM علاوه بر طولانی‌تر کردن زمان یادگیری نیاز به توان پردازشی بالایی نیز دارند، میتوان انتظار داشت که استفاده از لایه‌هایی همانند Transformers علاوه بر افزایش کیفیت خروجی (به دلیل استفاده از attention و self-attention و ...) سرعت یادگیری مدل‌ها را نیز افزایش داده و لذا قادر به یادگیری بیشتر مدل‌ها خواهیم بود. [۲۰]

References

- [1] Activation functions. Available at https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html.
- [2] AMIDI, A., AND AMIDI, S. Recurrent neural networks cheat-sheet. Available at <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>.
- [3] BENGIO, Y. A neural probabilistic language model. Available at <http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf>.
- [4] BENGIO, Y., DUCHARME, R., AND VINCENT, P. A neural probabilistic language model. Available at <https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf>.
- [5] BOWMAN, S., AND VILNIS, L. Generating sentences from a continuous space. Available at <https://arxiv.org/abs/1511.06349>.
- [6] BROWNLEE, J. How to calculate the kl divergence for machine learning. Available at <https://machinelearningmastery.com/divergence-between-probability-distributions/>.
- [7] DONAHUE, D., AND RUMSHISKY, A. Adversarial text generation without reinforcement learning. Available at <https://arxiv.org/abs/1810.06640>.
- [8] GANJOOR. Ganjoor. Available at <https://ganjoor.net/>.
- [9] GHADERI, A. Persian poems corpus. Available at https://github.com/amnghd/Persian_poems_corpus.
- [10] GOLDBERG, Y. Neural network methods in natural language processing. Available at <http://amzn.to/2wycQKA>.
- [11] GOODFELLOW, I. Generative adversarial nets. Available at <https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

- [12] HUI, J. Gan — why it is so hard to train generative adversarial networks. Available at https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b.
- [13] KARNIADAKIS, G. E. Dying relu and initialization: Theory and numerical examples. Available at <https://arxiv.org/abs/1903.06733>.
- [14] MAKHZANI, A., SHLENS, J., JAITLEY, N., GOODFELLOW, I., AND FREY, B. Adversarial autoencoders. Available at <https://arxiv.org/abs/1511.05644>.
- [15] MASRI, A. An intuitive explanation to autoencoders. Available at <https://towardsdatascience.com/autoencoders-in-keras-273389677c20>.
- [16] MITTAL, A. Understanding rnn and lstm. Available at <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>.
- [17] NICHOLSON, C. A beginner's guide to neural networks and deep learning. Available at <https://pathmind.com/wiki/neural-network>.
- [18] OLAH, C. Understanding lstm networks. Available at <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [19] ROCCA, J. Understanding variational autoencoders. Available at <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>.
- [20] VASWANI, A., AND ET AL. Attention is all you need. Available at <https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>.
- [21] WONG, W. What is teacher forcing? Available at <https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c>.
- [22] YU, L. Seqgan: Sequence generative adversarial nets with policy gradient. Available at <https://arxiv.org/abs/1609.05473>.