پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

بررسی سامانه‌های گفتگوگر موجود و پیاده‌سازی یک سامانه‌ی گفتگوگر مبتنی بر یادگیری ماشین

نگارنده

محمد رزاق‌نوری

استاد راهنما: دکتر هدیه ساجدی

پایاننامه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

تابستان ۱۳۹۶

چکیده

امروزه محققان زیادی در حال حل مسائل مختلفی که تا کنون قابل حل توسط ماشین‌ها نبودند با استفاده از یادگیری عمیق هستند. یکی از مسائلی که به نظر می‌رسد قابلیت حل آن با استفاده از شبکه‌های عصبی عمیق وجود دارد مسئله‌ی سامانه‌های گفتگوگر است. یک سامانه‌ی گفتگوگر سامانه‌ای است که قابلیت مکالمه با انسان از طریق زبان طبیعی را برخوردار می‌باشد و این به این معناست که این سامانه‌ها ورودی به زبان طبیعی گرفته و خروجی به زبان طبیعی تولید می‌کنند. در گذشته این سامانه‌ها با استفاده از روش‌های قوانین دست‌نویس، پردازش زبان طبیعی و یادگیری ماشین ساخته می‌شدند. اما شاید وقت آن رسیده تا راه جدیدی در پیاده‌سازی سیستم‌های گفتگوگر باز شود.

در این نوشتار به بررسی سامانه‌های گفتگوگر موجود خواهیم پرداخت و تمام اطلاعات مرتبط موجود از آن‌ها را جمع‌آوری خواهیم کرد و در انتها به پیاده سازی یک سامانه‌ی گفتگوگر با استفاده از شبکه‌های عصبی عمیق خواهیم پرداخت و نقاط قوت و ضعف آن را در برابر سامانه‌های گذشته بررسی خواهیم کرد.

پیشگفتار

سامانه‌های گفتگوگر به سامانه‌هایی اطلاق می‌شود که توانایی درک و انتقال ورودی و خروجی را به زبان طبیعی دارا هستند. هدف اصلی از ساختن این نوع سامانه ایجاد دستیاری هوشمند برای انسان طلقی می‌شود. از اهداف دیگر می‌توان به آموزش هوشمند افراد توسط ماشین، کمک به بیماران دارای اختلالات روانی، سرویس به مشتری، کسب اطلاعات و موارد بسیار دیگر اشاره کرد.

آلن تورینگ[1] در سال ۱۹۵۰ آزمایشی که بعدها به نام آزمایش تورینگ مشهور شد را مطرح می‌کند که به سوال آیا ماشین‌ها می‌توانند فکر کنند جواب می‌دهد. در این آزمایش هدف ماشین فریب دادن انسان نسبت به ماهیت اصلی خود است. به بیان ساده‌تر ماشین باید بتواند انسان را فریب داده و به او ثابت کند که در حال سخن گفتن با انسان دیگری است.

بعد از مطرح شدن آزمایش تورینگ بسیاری از محققان به دنبال ساخت چنین سیستمی فعالیت‌های گسترده‌ای انجام دادند. اما لازم بود تا جایزه‌ای سالانه نیز برای ایجاد انگیزه در محققان موجود باشد. از این رو هیو لوبنر[2] در سال ۱۹۹۰ جایزه‌ای سالانه برای ماشینی که از بقیه ماشین‌ها انسان‌تر به نظر رسد تعیین کرد. در این نوشتار ماشین‌های برنده‌ی جایزه لوبنر از سال ۲۰۰۰ تا کنون مورد بررسی قرار می‌گیرند.

در ادامه خواهیم دید که اکثر سامانه‌های گفتگوگر برنده‌ی جایزه لوبنر از روش‌هاش مبتنی بر قانون برای تولید پاسخ استفاده کرده و سعی در بیان زبان طبیعی با استفاده از قوانین دست‌نویس کرده‌اند و شاید به همین دلیل است که تا کنون هیچ سامانه‌ی گفتگوگری موفق به قبولی در آزمایش تورینگ نشده‌است. اما چند سالی است که یادگیری عمیق پا به عرصه‌ی وجود گذاشته و مسائل بسیاری را به مجموعه‌ی مسائلی که توسط ماشین‌ها قابل حل است اضافه کرده است. در این نوشتار سعی می‌کنیم مسئله‌ی ساختن یک سامانه‌ی گفتگوگر را به یادگیری عمیق بسپاریم تا از نوشتن تعداد بسیار زیادی قانون و قناعت به یک سامانه با دقت عملکرد پایین جلوگیری کنیم. سپس با بررسی نقاط قوت و ضعف این سامانه و نحوه‌ی رفع مشکلات آن نوشتار را به پایان می‌رسانیم.

در فصل اول مفاهیم مقدماتی مورد نیاز برای ادامه‌ی بحث را بررسی خواهیم کرد. در فصل دوم به بررسی سامانه‌های گفتگوگر موجود خواهیم پرداخت و بهترین سامانه‌ی گفتگوگر موجود را به اجمال بررسی می‌کنیم. در فصل سوم به بیان روش پیشنهادی بر اساس یادگیری عمیق پرداخته و

---

[1] Alan Turing
[2] Hugh Loebner

در فصل چهارم این روش را با استفاده از سوال و جواب با سامانه به آزمایش می‌گذاریم. در نهایت، قسمت نتیجه گیری خلاصه‌ای از مباحث نوشتار ارائه خواهد کرد.

فهرست مطالب

فصل ۱

مفاهیم مقدماتی

۱.۱ پردازش زبان طبیعی

۱.۱.۱ One-hot Encoding

برای تبدیل داده‌های رسته‌ای[۱] به بردار ویژگی از این نوع رمزگذاری استفاده می‌شود. به طور مثال تصور کنید که قرار است کلمات فرهنگ لغت را تبدیل به بردارهایی با طول ثابت کنید و این بردارها را به یک الگوریتم یادگیری وارد کنید. دلیل انجام این کار نیز عدم توانایی بسیاری از الگوریتم‌ها در گرفتن ورودی با طول متغیر می‌باشد. بنابراین برای انجام این کار از رمزگذاری با نام One-hot استفاده می‌کنیم.

فرض کنید که به هر کلمه در فرهنگ لغت مورد نظر یک عدد طبیعی مانند i اختصاص داده‌ایم. حال برای ساختن بردار هر کلمه ابتدا برداری به طول تعداد کلمات موجود در فرهنگ لغت ساخته و سپس تمام عناصر آن به جز عنصر iام را برابر با صفر و عنصر iام را برابر یک قرار می‌دهیم. به این روش از ساختن بردار One-hot encoding گوییم.

۱.۱.۲ Word2vec

استفاده از One-hot encoding سامانه را با محدودیت‌های جدی روبرو می‌کند. به طور مثال تصور کنید که یک الگوریتم یادگیری ماشین داریم که قرار است جملات مختلف را دسته‌بندی کند. حال اگر به این الگوریتم جمله‌ای نشان دهیم که در آن از کلمه گربه استفاده شده و در ادامه همان جمله را در مرحله‌ی تست به الگوریتم نشان دهیم اما این بار کلمه گربه را با سگ جابجا کنیم الگوریتم احتمالا در دسته‌بندی جمله با مشکل روبرو می‌شود. دلیل این امر هم این است که در رمزگذاری با

[۱] Categorical data

استفاده از الگوریتم One-hot بردارهای به وجود آمده از کلماتی که شباهت بالایی به یکدیگر دارند ممکن است کاملا متفاوت باشند.

از طرف دیگر بردارهای ساخته شده به وسیله‌ی One-hot encoding طول برابر با طول فرهنگ لغت دارند که معمولا عدد بزرگی محسوب می‌شود و این امر باعث می‌شود تا الگوریتم مجبور باشد در بعد بسیار بالا کار کند و با مسئله‌ی نفرین بعد[۲] روبرو شود.

از این رو در سال ۲۰۱۳ توماس میکُلُو[۳] موفق به ساخت الگوریتمی به نام word2vec شد که کلمات را به یک فضای برداری با ویژگی‌های بسیار مناسب می‌نگارد. این نگاشت به قسمی است ارتباط معنایی و نحوی موجود میان کلمات را حفظ می‌کند. به بیان ساده‌تر اگر میان دو کلمه ارتباط معنایی و یا نحوی خاصی موجود باشد این ارتباط درون فضای برداری نیز منعکس می‌شود[۱]. به طور مثال فاصله‌ی میان کلمات تهران و ایران به فاصله‌ی میان کلمات پاریس و فرانسه بسیار نزدیک است.

امروزه در بسیاری از مدل‌های یادگیری ماشین و یادگیری عمیق از این الگوریتم به منظور تبدیل کلمات به بردارهایی با طول ثابت استفاده می‌شود. البته این الگوریتم یک رقیب جدی به نام Glove [۲] دارد که در بسیاری از مقالات و مدل‌ها استفاده می‌شود.

AIML ۲.۱

زبان برنامه‌نویسی Artificial Intelligence Markup Language یا به اختصار AIML زبانی بر پایه‌ی XML است که برای طراحی سامانه‌های گفتگوگر طراحی شده است. سامانه‌ی گفتگوگر A.L.I.C.E که در سال‌های ۲۰۰۰، ۲۰۰۱ و ۲۰۰۴ موفق به دریافت جایزه‌ی لوبنر شد از این زبان برای نگهداری داده‌هایش استفاده می‌کند. در ادامه ساختار این زبان را بررسی خواهیم کرد. برای این منظور ابتدا برچسب‌های مهم این زبان را وصف کرده و کاربرد آن‌ها را با مثال‌هایی نمایش خواهیم داد. تمام مطالب این قسمت از منبع شماره [۳] انتخاب شده‌اند.

<aiml>

شروع کننده‌ی هر سند AIML یک برچسب <aiml> است. بدین وسیله کامپایلر زبان AIML متوجه وجود کدی از جنس AIML می‌شود. توجه کنید که این زبان مبتنی بر XML بوده و برچسب های این زبان عموما جفت خود را داشته و در جایی تمام می‌شوند. برچسب انتهایی شکلی مانند: <tag name/> دارد. به طور مثال برچسب انتهایی برای برچسب <aiml> به شکل <aiml/> خواهد بود.

<category>

هر سند AIML از تعداد زیادی برچسب <category> تشکیل شده است. این برچسب

---

[۲] Curse of dimension

[۳] Tomas Mikolov

عملا نگه دارنده‌ی اطلاعات سامانه‌ی گفتگوگر است. به این واحدهای اطلاعاتی AIML Object می‌گویند. به بیان ساده‌تر هر برچسب <category> شامل یک برچسب متناظر با مکالمه‌ی کاربر و یک پاسخ برای آن مکالمه‌ی احتمالی است.

هر برچسب <category> باید شامل یک برچسب <pattern> و یک برچسب <tem-plate> باشد. اطلاعات دیگری نیز میتواند به این برچسب اضافه شود که در ادامه شرح داده خواهند شد.

<pattern>

اولین برچسب درون هر برچسب <category> یک برچسب <pattern> است. این برچسب مکالمه‌ی احتمالی کاربر را درون خود نگه می‌دارد. این مکالمه می‌تواند شامل Wild cardها نیز باشد. برای مثال:

```
<pattern>Hello *</pattern>
```

با عباراتی همچون Hello there و Hello Mohammad تطبیق خواهد یافت.

<template>

از این برچسب برای نگهداری پاسخ سامانه به <pattern> مورد نظر استفاده می‌شود. این برچسب می‌تواند داده ذخیره کند، برنامه‌ی دیگری را صدا بزند، جواب مشروط دهد و کار را دست <category> های دیگر بسپارد[۴]. برای مثال برای <pattern> تعریف شده در بالا می‌توان برچسب زیر را در نظر گرفت:

```
<template>Hi, how are you?</template>
```

<that>

فرض کنید که میخواهید به ورودی Yes پاسخ دهید. اما باید بدانید کاربر به چه سوالی جواب مثبت داده است. برای ایجاد یک <category> که جواب قبلی سامانه را هم بررسی می‌کند از برچسب <that> استفاده میکنیم. به طور مثال:

```
<category>
        <pattern>YES</pattern>
        <that>Do you have a rabbit?</that>
        <template>God, I love rabbits.</template>
</category>
```

<star/>

اگر بخواهیم مقداری که کاراکتر ستاره درون برچسب <pattern> با آن تطبیق یافته است را درون برچسب <template> استفاده کنیم می‌توانیم از برچسب <star/> استفاده کنیم. یک مثال از استفاده از این برچسب در زیر قابل مشاهده است:

```
<category>
        <pattern>My name is *</pattern>
        <template>Hi <star /> </template>
</category>
```

دقت کنید که این برچسب دارای جفت نمی‌باشد و به صورت تنها به کار می‌رود. در ادامه مثالی از وجود چند کاراکتر ستاره خواهد آمد.

```
<category>
        <pattern>A * can be a *</pattern>
        <template>
        <star index='1'/> is not always <star index='2'/>
         </template>
</category>
```

<srai> گاهی نیاز به تقلیل یک <pattern> به <pattern> دیگر داریم. به بیان ساده‌تر گاهی نیاز است تا تمام الگوهای مشابه به یک الگو نگاشته شوند. در این موارد میتوان در الگوهای فرعی از برچسب <srai> استفاده کرده و در برچسب اصلی <template> پاسخ اصلی را نوشت.

```
<category>
        <pattern>Who are you?</pattern>
        <template><srai>What is your name?</srai><template>
</category>

<category>
        <pattern>What is your name?</pattern>
        <template>My name is UTBot.</template>
</category>
```

<random>

یکی از مواردی که باعث تشخیص ماشین از انسان می‌شود جواب یکسان ماشین به سوال یکسان است اما می‌دانیم که انسان‌ها چنین عمل نمی‌کنند. پس به منظور تنوع بخشیدن به جواب‌های سامانه می‌توان از برچسب <random> استفاده‌ی فراوانی کرد. درون این برچسب هر جواب احتمالی با یک برچسب <li> مشخص می‌شود.

```
<category>
        <pattern>Who are you?</pattern>
        <template><srai>What is your name?</srai><template>
```

```
</category>

<category>
        <pattern>What is your name?</pattern>
        <template>
                <li>My name is UTBot.</li>
                <li>I am UTBot. Who are you?</li>
                <li>They call me UTBot. Nice to meet you.</li>
        </template>
</category>
```

<set> و <get/>

تصور کنید که کاربر به سامانه رنگ مورد علاقه خود را معرفی میکند و چند دقیقه‌ی بعد رنگ مورد علاقه خود را از سامانه می‌پرسد. در چنین شرایطی باید بتوان اطلاعات را به عنوان متغیر در سامانه ذخیره کرد. برچسب‌های <set> و <get/> به همین منظور ایجاد شده‌اند.

```
<category>
        <pattern>* is *</pattern>
        <template>
                Ok, thanks. I will remember that.
                <set name=<star index='1'/>><star index='2'></
                    set>
        </template>
</category>

<category>
        <pattern>What is *?</pattern>
        <template>
                The <star/> is <get name=<star/>/>
        </template>
</category>
```

۳.۱ شبکه عصبی

۱.۳.۱ مفاهیم پایه

پرسپترون

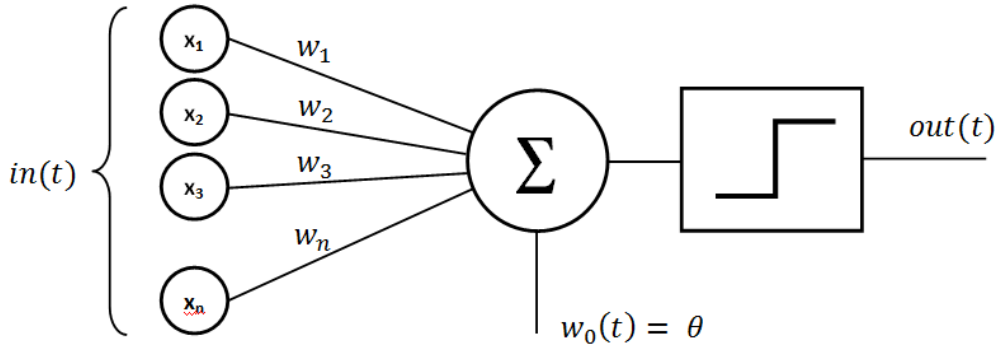
یک نورون طبیعی با ساده‌ترین نگرش ممکن یک سامانه‌ی غیر خطی است که توسط دندریت‌هایش که نقش ورودی نورون را بر عهده دارند جریان الکتریکی را جمع کرده و در صورت عبور مقدار جریان جمع شده از حد آستانه‌ی نورون، پالسی از آکسون‌هایش به سمت دندریت نورون‌های دیگر

شلیک می‌کند. لازم به ذکر است که میزان تأثیر هر نورون به نورون بعدی میتواند تغییر کند و همین امر است که یادگیری را ممکن می‌سازد.

فرض کنید قصد مدل سازی یک نورون طبیعی را داریم. هر نورون باید دارای چندین ورودی وزن‌دار و خروجی باشد. نورون مصنوعی هر ورودی را در وزن متناظرش ضرب کرده و در خود جمع می‌زند و مقدار آن را با یک حد آستانه مقایسه می‌کند. در صورت بیشتر بودن این مقدار از مقدار آستانه نورون مقدار یک و در غیر این صورت مقدار صفر برمی‌گرداند. اگر ورودی را با بردار x و خروجی را با بردار w نمایش دهیم نورون مورد نظر به صورت یک تابع مانند f قابل ارائه می‌باشد.

$$f(x) = \begin{cases} 1 & \text{if} \quad w.x + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{۱.۱}$$

اگر با دقت به معادله‌ی ۱.۱ بنگریم خواهیم دید که این نورون فضای نمونه‌ها را با یک ابر صفحه به دو قسمت تقسیم می‌کند. به این ابر صفحه مرز تصمیم گفته می‌شود و از نورون ساخته شده جهت دسته‌بندی نمونه‌ها استفاده می‌شود. توجه داشته باشید که در صورت نبود b ابر صفحه‌ی تعریف‌شده توسط نورون به عنوان مرز تصمیم همواره از مرکز عبور کرده و دسته‌بند ساخته‌شده قابلیت دسته‌بندی بسیاری از مسائل را از دست می‌دهد. به نورونی که به صورت بالا و با معادله‌ی ۱.۱ ساخته شده باشد پرسپترون گوییم. یک نمونه پرسپترون را در شکل ۱.۱ مشاهده می‌کنید.



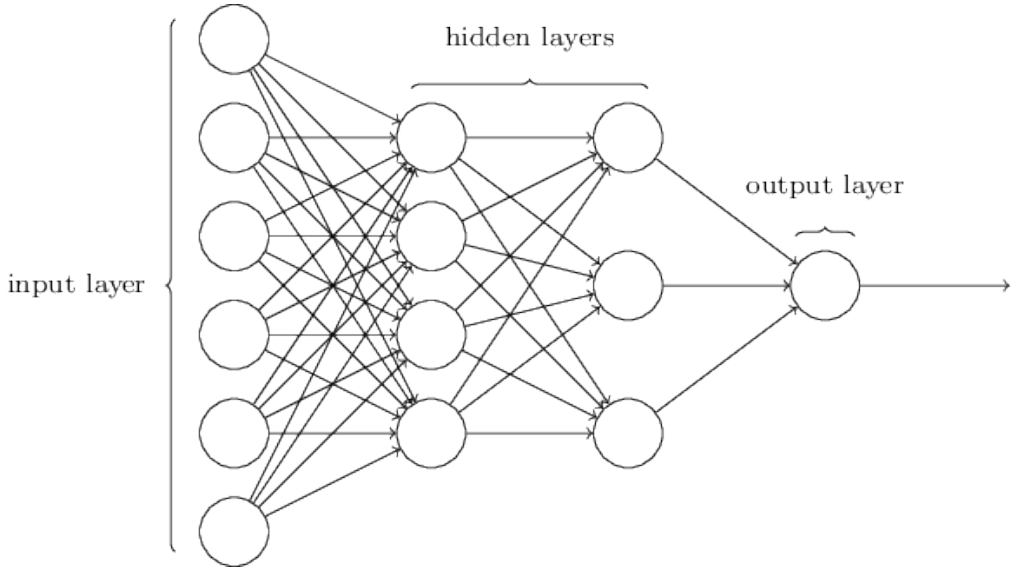
شکل ۱.۱: نمایی از یک پرسپترون

پرسپترون چندلایه

در ادامه میخواهیم پرسپترون‌ها را به یکدیگر متصل کنیم تا مرز تصمیم پیچیده‌تری بدست آوریم. فرض کنید که یک شبکه از پرسپترون‌ها با i لایه ساخته‌ایم، شبکه‌ی $i+1$ لایه‌ای را با قرار دادن

n_{i+1} نورون در لایه $i+1$ و اتصال خروجی تمامی نورون های لایه‌ی i ام به ورودی تمام نورون های لایه $i+1$ بدست می‌آوریم. دقت کنید که در هر لایه هیچ نورونی به نورون مجاور خود متصل نیست و اتصالات تنها از هر لایه به لایه‌ی بعد هستند. به اولین لایه و آخرین لایه به ترتیب لایه‌ی ورودی و لایه خروجی و به لایه‌های غیر از این دو لایه،لایه‌های مخفی اطلاق می‌شود.

حال اگر تعداد زیادی از پرسپترون ها را به قسمی که در بالا توضیح داده شد به یکدیگر متصل کنیم یک شبکه عصبی خواهیم داشت که به پرسپترون چندلایه [۴] معروف است. به صورت نمونه ساختار یک شبکه پرسپترون چندلایه را در شکل ۲.۱ مشاهده می‌کنید.



شکل ۲.۱: نمایی از یک شبکه‌ی پرسپترون چندلایه

۲.۳.۱ شبکه عصبی پیشخور

شبکه‌های عصبی پیشخور [۵] شبکه‌هایی هستند که در آن‌ها هیچ گونه حلقه و یا دوری موجود نیست و نتیجتاً جریان داده در این نوع شبکه به صورت کاملاً یک جهته و از لایه‌ی ورودی به لایه‌ی خروجی است. شبکه عصبی پرسپترون چندلایه که در بالا بررسی شد نوعی از شبکه‌های عصبی پیشخور است.

شبکه‌های عصبی پیشخور برای پردازش داده‌هایی که وابسته به زمان نیستند مناسب هستند. به عنوان مثال تصاویر ورودی‌های مناسبی برای شبکه‌های عصبی پیشخور محسوب می‌شوند. اما

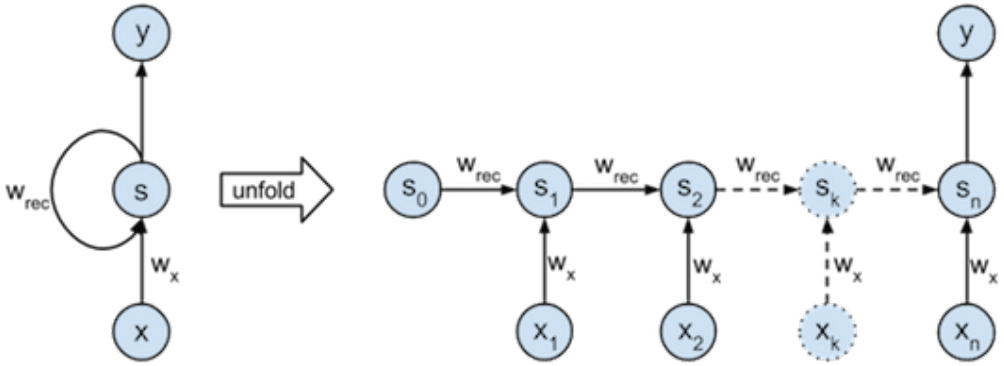
[۴] Multilayer perceptron (MLP)
[۵] Feedforward Neural Network

متن، ویدئو و صوت به دلیل وابسته بودن به زمان عموما ورودی‌های مناسبی برای شبکه های عصبی پیشخور به شمار نمی‌آیند.

۳.۳.۱ شبکه عصبی بازگشتی

از مطالبی که پیشتر گفته شد چنین برمی‌آید که شبکه‌های عصبی پیشخور در حل بسیاری از مسائل با مشکل مواجه هستند زیرا مقدار قابل توجهی از داده‌ها با مفهوم زمان در ارتباط بوده و با حذف عنصر زمان از آن‌ها کیفیت این دسته از داده از بین می‌رود. لذا در مقابل شبکه های عصبی پیشخور نوعی دیگر از شبکه‌های عصبی به نام شبکه های عصبی بازگشتی[۶] مطرح می‌شوند.

شبکه عصبی بازگشتی نوعی شبکه عصبی است که ارتباط بین واحدها تشکیل یک دور جهت‌دار می‌دهد. این ویژگی شبکه‌های عصبی بازگشتی باعث ایجاد درکی از زمان برای شبکه می‌شود و این به این معناست که شبکه قابلیت پردازش داده‌های دارای توالی را بدست می‌آورد. شکل ۳.۱ یک شبکه عصبی بازگشتی را نشان می‌دهد. همان‌طور که مشاهده می‌کنید این نوع شبکه به دو صورت قابل نمایش است. نمایش سمت راست نمایش بازشده[۷]ی نمایش سمت چپ است. توجه داشته باشید که w_{rec} با ایجاد ارتباط میان لایه‌ی وضعیت که با s نشان داده شده است با خودش باعث Recurrent شدن شبکه می‌شود.



شکل ۳.۱: شبکه عصبی بازگشتی با دو نمایش متفاوت

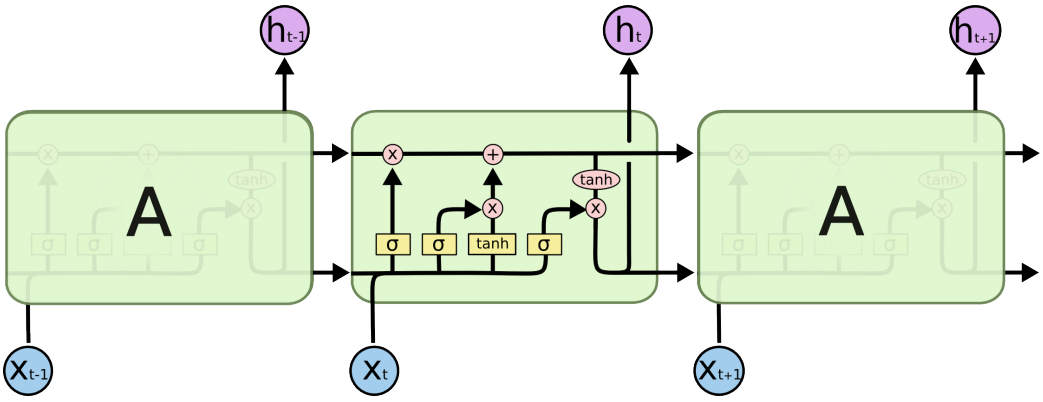
۴.۳.۱ شبکه عصبی LSTM

شبکه‌های عصبی پایه که در بالا گفته شد به منظور ایجاد قابلیت اثرگذاری موارد پیشین در تصمیم حال حاضر شبکه به وجود آمدند. اما در واقع این شبکه‌ها قابلیت به‌خاطرسپاری اطلاعاتی که

[۶] Recurrent Neural Network (RNN)
[۷] unfolded

مدت زیادی از نشان دادنشان به شبکه میگذرد را ندارند. اما در بسیاری از موارد هدف ما دقیقا به‌خاطرسپاری دنباله‌هایی با طول زیاد است. به طور مثال تصور کنید که میخواهیم شبکه‌ای داشته باشیم که هر بار کلمه‌ی بعدی را در متن داده شده حدس بزند و فرض کنید پاراگرافی با طول ۲۰۰ کلمه که با جمله‌ی "زبان فارسی که..." شروع شده و با جمله‌ی "در کشورهایی همچون افعانستان، پاکستان و ...زبان رسمی می‌باشد." خاتمه می‌یابد به شبکه نشان داده شده است. شبکه برای حدس زدن کلمه‌ی ایران در جمله‌ی انتهایی نیاز به یادآوری کلمه "فارسی" در جمله‌ی ابتدایی دارد و فاصله‌ی این دو کلمه چیزی در حدود ۲۰۰ کلمه است. شبکه‌های بازگشتی ساده که در بالا معرفی شدند قادر به یادآوری دنباله‌های با طول زیاد نیستند بنابراین نیاز به نوعی دیگر از شبکه داریم که قابلیت به‌خاطرسپاری دنباله‌هایی با طول زیاد را داشته باشد. یک نمونه شبکه که چنین قابلیتی را فراهم می‌آورد و در زمان نگارش این مقاله با محبوبیت بالایی روبروست به شبکه عصبی LSTM معروف است. دلیل این که یک شبکه RNN ساده اطلاعاتی که در گذشته دور به شبکه نشان داده شده‌اند را فراموش می‌کند به صورت کامل در منبع شماره [۵] قابل مطالعه است اما به طور مختصر می‌توان گفت که یادگیری در شبکه‌های عصبی غالبا با استفاده از گرفتن گرادیان یک تابع انجام می‌شود و در این مقدار هر چه فاصله زمانی میان وضعیت حال حاضر و وضعیتی که مورد نیاز است بیشتر می‌شود در بسیاری از موارد کاهش می‌یاد و این کاهش تا جایی ادامه پیدا می‌کند که مقدار گرادیان تابع عملا به مقداری نزدیک به صفر می‌رسد و این مقدار در خوش‌بینانه ترین حالت آن‌قدر کوچک است که توانایی تغییر وزن‌ها را نداشته و یادگیری اتفاق نمی‌افتد.

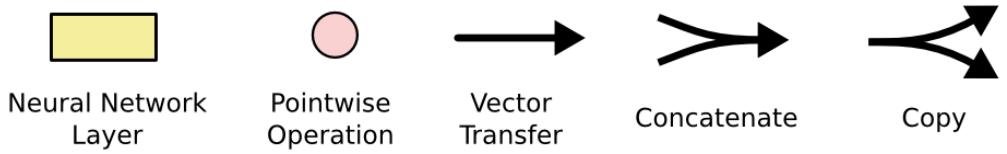
شبکه عصبی Long Short-Term Memory یا به اختصار LSTM نوعی خاص از شبکه عصبی بازگشتی است که با اضافه کردن دروازه‌های مختلف به هر نورون امکان به‌خاطرسپاری دنباله‌هایی با طول زیاد را برای شبکه فراهم می‌آورد. شکل ۴.۱ نمایی کلی از یک نورون LSTM به دست می‌دهد.



شکل ۴.۱: یک نورون از شبکه عصبی LSTM که در طول زمان نشان داده شده است

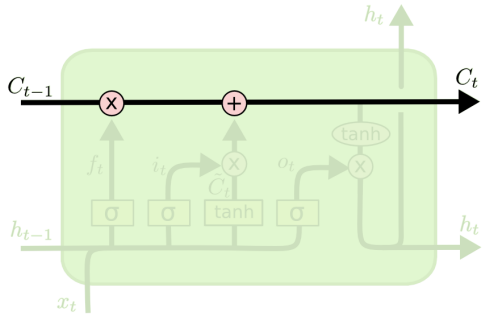
ابتدا اجازه دهید نمادگذاری موجود در شکل ۴.۱ را بررسی کنیم. همان طور که در شکل ۵.۱

مشاهده میکنید هر خط جهت‌دار نمایانگر جابجایی یک بردار از ابتدای خط به انتهای آن، هر دو خطی که به هم می‌پیوندند نمایانگر تلفیق و هر دو خط که از یکدیگر جدا می‌شوند نمایانگر نسخه برداری هستند. دایره‌های صورتی نشان‌دهنده‌ی عملگر نقطه‌ای و مستطیل‌ها نیز نشان‌دهنده‌ی یک لایه شبکه عصبی هستند.



شکل ۵.۱: نمادگذاری موجود در سلول LSTM

در شکل پیچیده‌ی نورون LSTM چیزی که باید مورد توجه قرار گیرد خطی است که از بالای نورون عبور کرده و از نورون بیرون می‌رود. این خط نمایانگر وضعیت سلول[۸] است. که بعد از تغییراتی از سلول خارج میشود. این بردار را میتوانید در شکل ۶.۱ مشاهده کنید.



شکل ۶.۱: وضعیت سلول در نورون LSTM

هر سلول LSTM دارای سه دروازه[۹] است. هر دروازه از یک لایه شبکه عصبی با تابع فعال‌سازی سیگموید[۱۰] و یک ضرب نقطه‌ای تشکیل شده است. سه دروازه‌ی نورون LSTM شامل دروازه‌های فراموشی، دروازه ورودی و دروازه خروجی هستند که در زیر به تفصیل شرح داده خواهند شد.

---

[۸] Cell state
[۹] Gate
[۱۰] Sigmoid

دروازه‌ی فراموشی

این دروازه وظیفه‌ی از بین بردن اطلاعاتی که دیگر مورد نیاز نیستند را بر عهده دارد. بنابراین مقدار خروجی در گام زمانی[11] پیشین به علاوه‌ی ورودی در گام زمانی کنونی را گرفته و برداری با مقادیر بین صفر و یک تولید می‌کند و چون مقدار خروجی دروازه ها بردارهایی با عناصر بین صفر و یک هستند اگر مقدار دروازه فراموشی را در مقدار وضعیت سلول ضرب کنیم از هر واحد از وضعیت سلول تنها مقدار دلخواه باقی خواهد ماند. مقدار دروازه فراموشی از رابطه‌ی ۲.۱ بدست می‌آید.

$$f_t = Sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f) \qquad (۲.۱)$$

در ادامه این مقدار باید در مقدار وضعیت سلول به صورت نقطه‌ای ضرب شود.

دروازه‌ی ورودی

قدمی بعدی این است که تعیین کنیم چه اطلاعاتی قرار است به سلول اضافه کنیم برای این عمر نیاز به دو کار داریم: اول این که تعیین کنیم هر واحد از وضعیت سلول قرار است چقدر تغییر کند و در قدم دوم باید تعیین کنیم به هر واحد باید چه مقداری اضافه شود. قسمت اول را می‌توانیم با یک دروازه کنترل کنیم. به این دروازه دروازه ورودی گفته می‌شود. دروازه‌ی ورودی مقدار خروجی در گام زمانی قبلی و مقدار ورودی در گام زمانی کنونی را گرفته و برداری با المان‌های بین صفر و یک تولید می‌کند. برای قسمت دوم هم می‌توانیم از یک لایه شبکه عصبی با تابع فعال‌سازی تانژانت هیپربولیک[12] استفاده کنیم. ورودی این شبکه عصبی هم همانند دروازه‌های قبلی و بعدی خروجی سلول در گام زمانی قبلی و ورودی سلول در گام زمانی کنونی خواهد بود. مقدار دروازه‌ی ورودی را با i_t و مقدار تولید شده به عنوان نامزد جایگزینی وضعیت سلول را با $\widetilde{C}_t$ نمایش می‌دهیم. مقادیر ذکر شده از رابطه‌های ۳.۱ و ۴.۱ بدست می‌آیند.

$$i_t = Sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad (۳.۱)$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \qquad (۴.۱)$$

به روز رسانی وضعیت سلول

حال با استفاده از دروازه‌ی ورودی و مقدار نامزد جایگزینی وضعیت سلول، مقدار جدیدی برای وضعیت سلول انتخاب خواهیم کرد. این مقدار را در معادله‌ی ۵.۱ مشاهده میکنید.

$$C_t = f_t * C_{t-1} + (1 - f_t) * \widetilde{C}_t \qquad (۵.۱)$$

[11] Time step
[12] tanh

دروازه‌ی خروجی

در نهایت برای تولید خروجی نیاز به یک دروازه و یک لایه شبکه عصبی داریم. دروازه‌ی ذکر شده تعیین می‌کند که چه مقدار از هر واحد از سلول باید به بیرون از سلول انتقال پیدا کند و لایه‌ی ذکر شده مقدار وضعیت سلول را گرفته و خروجی هر واحد را تعیین می‌کند. مقدار دروازه خروجی در گام زمانی t را با o_t و مقدار خروجی در گام زمانی t را با h_t نمایش می‌دهیم. روابط ۶.۱ و ۷.۱ این مقادیر را نشان می‌دهند.

$$o_t = Sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o) \qquad (۶.۱)$$

$$h_t = o_t * tanh(C_t) \qquad (۷.۱)$$

برای اطلاعات بیشتر در مورد نحوه‌ی عملکرد شبکه‌های LSTM به منبع شماره [۵] مراجعه فرمایید.

۵.۳.۱ مدل‌های دنباله به دنباله

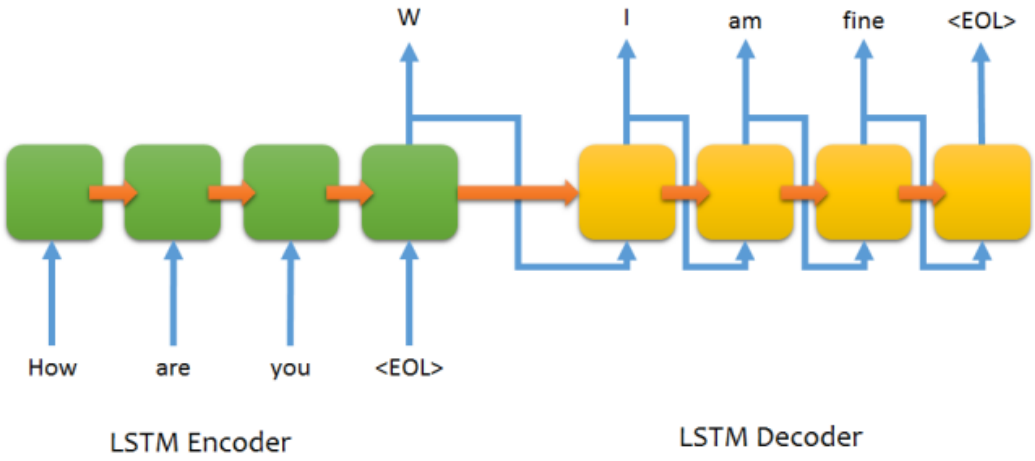
پیشتر نحوه‌ی ایجاد یک سلول LSTM مورد بررسی قرار گرفت. اما همواره می‌توان سلول‌های LSTM را پشت به پشت یکدیگر قرار داده تا تشکیل یک شبکه پیچیده‌تر دهند. به شبکه‌ای که دارای تعداد زیادی لایه‌ی مخفی باشد شبکه عصبی عمیق[۱۳] اطلاق می‌شود. بنابراین با قرار دادن تعدادی سلول LSTM پشت به پشت هم میتوان یک شبکه عصبی عمیق تشکیل داد. شبکه‌های عصبی عمیق در بسیاری از مسائل دنیای امروزی موفقیت چشمگیری کسب کرده‌اند اما این شبکه‌ها تا قبل از مطرح شدن مدل دنباله به دنباله توسط Ilya Sutskever و همکارانش قادر به حل مسائلی که دنباله‌ها را به دنباله‌ها می‌بردند نبودند[۶].

مدل دنباله به دنباله از دو شبکه LSTM تشکیل شده است. شبکه‌ی اول که به رمزگذار[۱۴] معروف است وظیفه‌ی رمزگذاری دنباله ورودی یا به بیان دیگر تبدیل دنباله‌ای از بردارها به یک بردار و شبکه‌ی دوم که به رمزگشا معروف است وظیفه‌ی رمزگشایی یا به بیان دیگر تبدیل بردار تولید شده توسط شبکه‌ی اول به دنباله‌ی خروجی را بر عهده دارد. در شکل ۷.۱ یک نمونه از مدل دنباله به دنباله را مشاهده می‌کنید.

---

[۱۳] Deep Neural Network (DNN)

[۱۴] Encoder



شکل ۷.۱: مدل دنباله به دنباله

همان‌طور که در شکل مشخص است خروجی رمزگذار غالبا دور ریخته شده و تنها وضعیت سلول است که نگه داشته می‌شود. در انتهای دنباله ورودی وقتی علامت انتهای خط[۱۵] به رمزگذار داده می‌شود خروجی رمزگذار نگه داشته می‌شود و به عنوان ورودی به رمزگشا داده می‌شود. توجه کنید که وضعیت سلول رمزگشا از صفر آغاز نشده و وضعیت سلول رمزگذار به عنوان وضعیت سلول ابتدایی رمزگشا در نظر گرفته می‌شود. این مقدار در واقع همان بردار با طول ثابتی است که انتظار داریم رمزگذار تولید کند. حال از این به بعد هر بار خروجی رمزگشا به خودش داده می‌شود و انتظار می‌رود که عنصر بعدی دنباله تولید شود. اگر دقت کنید رمزگشا عملا یک مدل زبانی است که با وضعیت سلول رمزگذار شرطی شده باشد. وضعیت سلول انتهایی رمزگذار را بردار تفکر[۱۶] نیز می‌نامند.

۶.۳.۱ سازوکار توجه

فرض کنید که می‌خواهیم یک مترجم زبان فارسی به انگلیسی طراحی کنیم. برای این کار می‌توانیم از یک مدل دنباله به دنباله که در بالا توضیح داده شد استفاده کنیم. می‌دانیم که رمزگذار در نهایت دنباله‌ی ورودی را به یک بردار می‌نگارد. بدیهی است که اگر این بردار بخواهد تمام اطلاعات مربوط به دنباله‌ی ورودی را در خود حفظ کند باید طول زیادی داشته باشد. از طرف دیگر رمزگشا در هر گام زمانی معمولا در حال ترجمه‌ی یکی از کلمات دنباله‌ی ورودی است. اما بدست آوردن

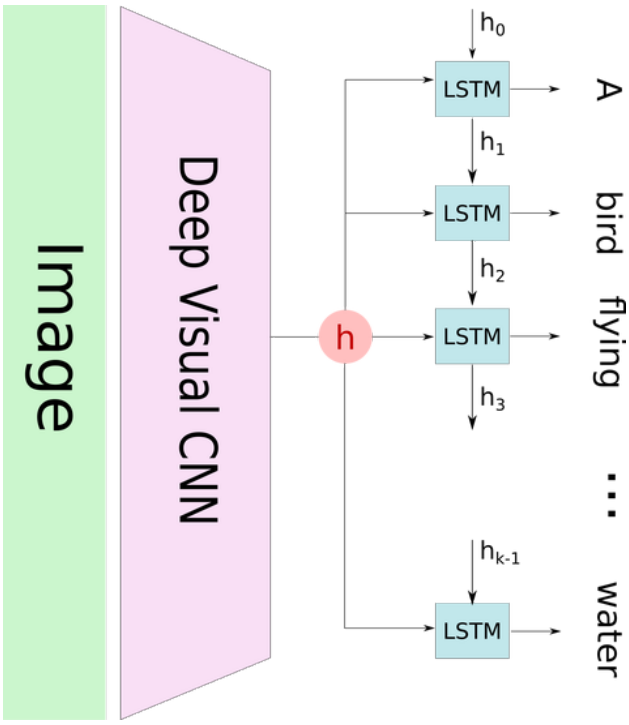
[۱۵] EOL
[۱۶] Thought vector

یک کلمه از بین بردار با طول زیاد بهترین راه حل به نظر نمی‌رسد. راه حل بهتر پیاده سازی سیستمی است که توجه خود را در طول زمان به نقاط مختلفی از دنباله‌ی ورودی معطوف کند.

به عنوان مثالی دیگر فرض کنید که قصد پیاده سازی سامانه‌ای را داریم که وجود و یا عدم وجود یک شیء خاص درون عکسی با اندازه‌ی بسیار بالا را بیان می‌کند. بدیهی است که پردازش کل عکس کاری غیر بهینه و در برخی موارد کاری نشدنی به حساب می‌آید. همچنین پژوهش‌های انجام شده در زمینه‌ی علوم اعصاب نشان می‌دهد که انسان‌ها نیز در مواجهه با مسائل توجه خود را به قسمتی از مسئله معطوف میکنند.

مثال‌هایی مانند مثال‌های بالا باعث به وجود آمدن سازوکاری به نام توجه شد. این سازوکار در دو نوع نرم و سخت موجود است که اجمالا توصیف خواهند شد.

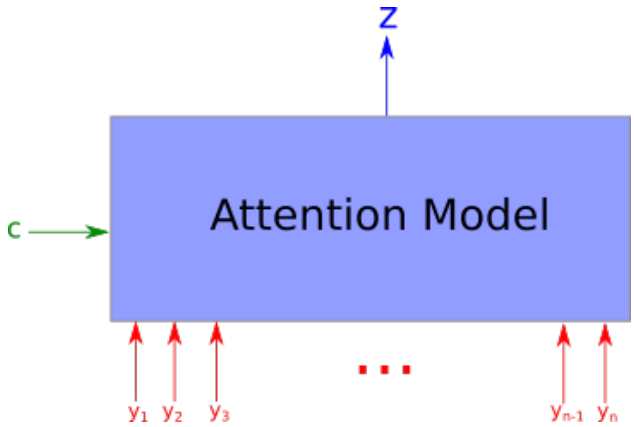
توجه نرم



شکل ۸.۱: مدل کلاسیک عنوان‌گذاری تصویر

فرض کنید که قصد پیاده سازی سامانه‌ای را داریم که با دیدن هر عکس بتواند صحنه‌ی موجود در عکس را به صورت متنی توصیف کند. در حالت کلاسیک با قرار دادن یک شبکه عصبی

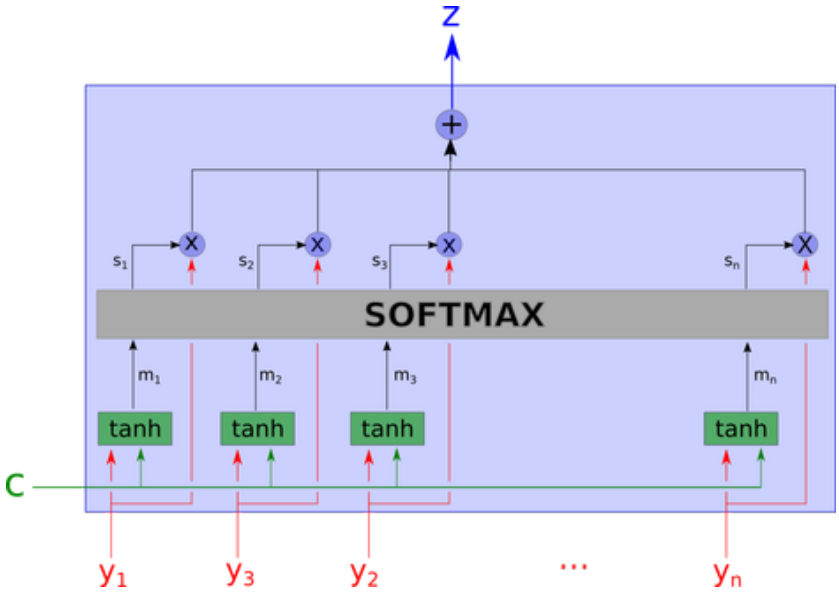
کانولوشن[۱۷] برداری مانند h تولید شده که به یک شبکه عصبی LSTM وارد شده و رمزگشایی میشود تا متن خروجی را تولید کند (شکل ۸.۱). اما در هر لحظه از تولید خروجی شبکه در حال توصیف یک قسمت از تصویر است. پس بهتر است این قسمت بیشتر مورد توجه قرار گیرد.



شکل ۹.۱: مدل کلی توجه

ابتدا اجازه دهید مکانیز کلی توجه را مورد بررسی قرار دهیم. همان‌طور که در شکل ۹.۱ مشاهده میکنید، توجه در حالت کلی n قسمت ورودی یا $y_1, \ldots, y_n$ را همراه با یک «زمینه» یا c دریافت میکند و z که خلاصه‌ای از ورودی‌ها میباشد خارج می‌کند. به بیان بهتر مکانیز توجه میانگین حسابی y_iها را که با مقدار شباهت y_i با c وزن‌دار شده‌اند برمی‌گرداند. حال به بررسی دقیق تر ساختار توجه می‌پردازیم.

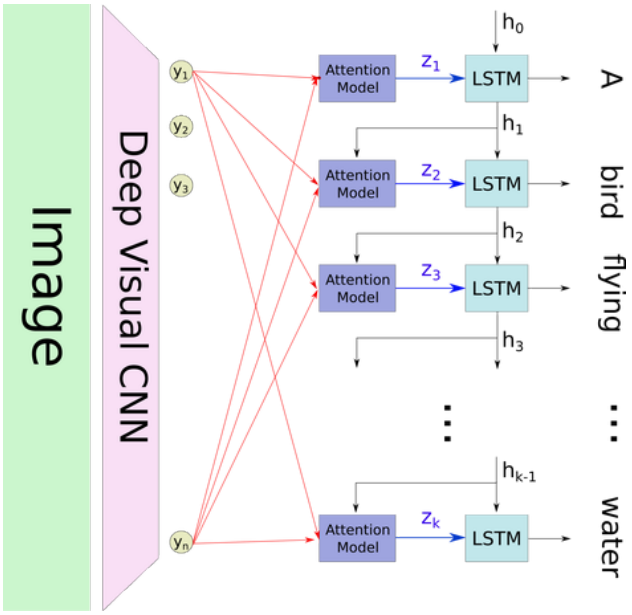
[۱۷] Convolutional Neural Network



شکل ۱۰.۱: جزئیات مدل کلی توجه نرم

ابتدا به ازای هر i، ورودی متناظر با قسمت i یعنی y_i با c تجمیع شده و از یک لایه تانژانت هیپربولیک عبور می‌کند و مقداری بین 1 و -1 به سمت بالا هدایت می‌شود. به این مقدار m_i می‌گوییم. توجه داشته باشید که این تابع می‌تواند غیر از تابع تانژانت هیپربولیک باشد. به طور مثال می‌توان از ضرب نقطه‌ای استفاده کرد. سپس تمام m_i ها به یک تابع Softmax وارد می‌شوند تا مجموع آن‌ها برابر عدد یک شود. مقادیر خارج شده از Softmax را با s_i نشان می‌دهیم. حالا s_i ها که عملا میزان توجه به هر قسمت هستند در مقدار همان قسمت یعنی y_i ضرب می‌شوند. در نهایت تمام مقدار بدست آمده با یکدیگر جمع شده تا خروجی نهایی را بدست دهند.

حال تصور کنید که می‌خواهیم مسئله گفته شده را توسط توجه حل کنیم. همان‌طور که در شکل ۱۱.۱ مشاهده می‌کنید، در هر گام زمانی وضعیت سلول LSTM در گام زمانی قبل به عنوان زمینه (c) وارد لایه‌ی توجه شده و مقدار ورودی LSTM در گام زمانی کنونی بدست می‌آید.

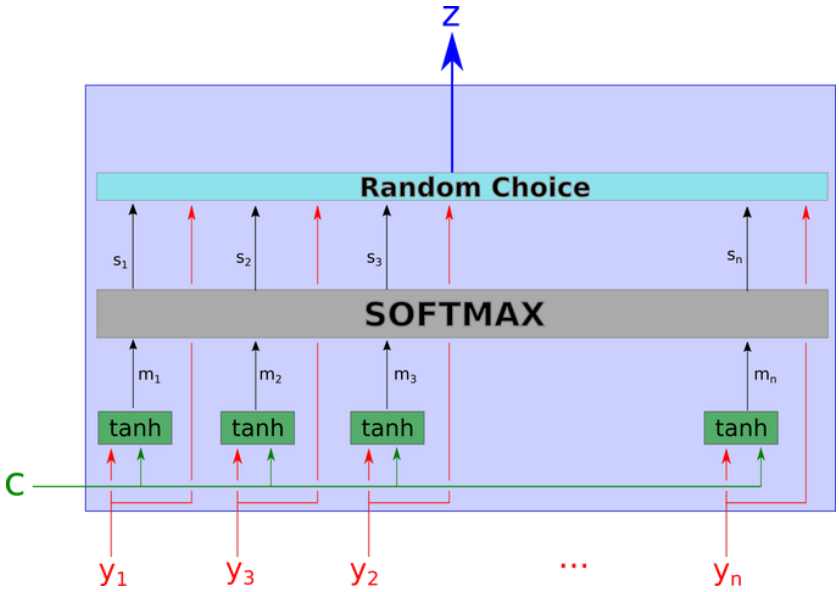


شکل ۱۱.۱: مدل عنوان‌گذاری تصویر همراه با سازوکار توجه

توجه سخت

تنها تفاوت توجه سخت با توجه نرم این است که توجه سخت به جای گرفتن میانگین حسابی یک قسمت را بر اساس میزان احتمالی که به آن اختصاص یافته انتخاب می‌کند (شکل ۱۲.۱). بنابراین توجه سخت یک فرآیند تصادفی است.

نکته‌ی جالبی که در مورد توجه نرم وجود دارد مشتق پذیر بودن کل فرآیند است. به همین دلیل سامانه‌ای که توجه نرم را درون خود پیاده سازی کند می‌تواند از ابتدا تا انتها با یک فرآیند بهینه‌سازی کار را ادامه دهد و نیاز به تکه تکه آموزش دادن سامانه نیست. اما این مسئله در مورد توجه سخت برقرار نیست.



شکل ۱۲.۱: جزئیات مدل کلی توجه سخت

۴.۱ آزمایش تورینگ و جایزه‌ی لوبنر

آلن تورینگ[۱۸] به منظور ایجاد معیاری برای محققان هوش مصنوعی در راستای سوال آیا ماشین می‌تواند فکر کند بازی تقلید[۱۹] را مطرح می‌کند. بازی تقلید با شرکت یک مرد یک زن و یک بازجو شروع می‌شود. بازجو یکی را به عنوان X و دیگری را به عنوان Y می‌شناسد و باید سعی کند تشخیص دهد کدام مرد و کدام زن هستند. در ادامه تورینگ یکی از دو نفر را با یک ماشین جابجا می‌کند و از بازجو می‌پرسد کدام عامل انسان و کدام ماشین است. بنابراین سوال «آیا ماشین می‌تواند فکر کند» به سوال «آیا ماشین می‌تواند بازجو را فریب دهد تا باور کند که در حال صحبت با انسان است» تقلیل می‌یابد.

از سال ۱۹۹۰ جایزه‌ی صد هزار دلاری به اولین کامپیوتری که جواب‌هایی دهد که از جواب‌های انسان قابل تفکیک نیست پیشنهاد می‌شود. متاسفانه تا کنون هیچ‌کس موفق به ساخت چنین کامپیوتری نشده است. اما از سال ۱۹۹۱ هر سال مبلغ دو هزار دلار و یک مدال برنز به کامپیوتری که نسبت به بقیه‌ی رقبا جواب‌های شبیه‌تری به انسان تولید کند اهدا می‌شود. به سامانه‌های گفتگوگری که به این مدال دست پیدا می‌کنند گفتگوگر برنده‌ی جایزه لوبنر[۲۰] اطلاق می‌شود.

---

[۱۸] Alan Turing
[۱۹] Imitation game
[۲۰] Loebner Prize

فصل ۲

بررسی سامانه‌های گفتگوگر موجود

در این بخش به بررسی سامانه‌های گفتگوگر برنده‌ی جایزه لوبنر می‌پردازیم و روش‌های آن ها را به صورت دقیق بررسی میکنیم. روند بررسی به ترتیب تاریخی خواهد بود.

A.L.I.C.E ۱.۲

Artificial Linguistic Internet Computer Entity یا به اختصار ALICE در سال ۱۹۹۵ توسط دکتر ریچارد والاس[۱] پیاده سازی شد. دانش ALICE درون فایل‌های AIML ذخیره میشود. این فایل‌ها همان‌طور که در قسمت‌قبل اشاره شد قانون‌هایی را در خورد ذخیره می‌کنند که در صورت تطابق ورودی کاربر با این قوانین خروجی مورد نظر تولید می‌شود. اما زبان طبیعی قانون‌مند نبوده و علی‌رغم تلاش‌های گسترده زبان طبیعی را نمی‌توان با استفاده قوانین محدود کرد. از این رو این سامانه با مشکلاتی روبرو بوده و نمی‌تواند تست تورینگ را با موفقیت رد کند.

Ella ۲.۲

سامانه‌ی گفتگوگر Ella آگوست سال ۲۰۰۰ توسط کوین کاپل [۲] طراحی و ساخته شد. خاص بودن این سامانه از استفاده از پردازش زبان طبیعی نشأت می‌گیرد. این سامانه از توانایی‌هایی از جمله بازی کردن و داشتن شخصیت خاص خود برخوردار است. در سال ۲۰۰۱ این سامانه در مسابقه‌ی جایزه‌ی لوبنر مقام دوم را کسب کرده و در سال ۲۰۰۲ برنده‌ی این مسابقه شد[۷].

---

[۱] Richard S. Wallace

[۲] Kevin Copple

Jabberwacky ۳.۲

این سامانه همواره سعی در شبیه‌سازی مکالمه انسانی به شکل جذاب، سرگرم‌کننده و طنزآمیز دارد. قسمت متفاوت Jabberwacky این است که این سامانه یاد می‌گیرد و این یادگیری از نگاهی شبیه به یادگیری قوانین، زبان، حقایق و متن توسط انسان است. توجه کنید که این سامانه از هیچ گونه قانون به طور دستی اضافه شده توسط انسان استفاده نمی‌کند و همین مسئله است که آن را از دیگر سامانه‌هایی که تا کنون بررسی شد متمایز می‌کند. هوش عمومی این سامانه هر چیزی که هر کسی گفته است را ذخیره می‌کند و از طریق تطبیق الگوی متنی نزدیک ترین عبارت را به عبارت دریافت شده پیدا می‌کند. اگر با آن با زبان خارجی صحبت کنید زبان شما را یاد گرفته و اگر اطلاعات کافی به او بدهید می‌تواند به زبان خواسته شده سخن بگوید. مهم نیست که زبان سخن گفتن شما زبان عامیانه یا بازی با لغت یا طنز باشد این سامانه آن را یاد خواهد گرفت. می‌توان به آن به عنوان یک ویکی‌پدیای گفتگو کننده نگاه کرد. اما علی‌رغم برتری نسبی نسبت به سامانه‌هایی که در گذشته بررسی شد این سامانه هنوز از نقطه ضعف هایی رنج می‌برد. از جمله این نقاط ضعف می‌توان به در نظر نگرفتن دنباله‌ی گفتگو و عدم کنترل روی مکالمه اشاره کرد[۸].

Ultera Hal ۴.۲

Ultera Hal یک سامانه‌ی گفتگوگر است که به منظور خدمت‌رسانی به عنوان یک دستار ساخته شده است. این سامانه از طریق یک رابط زبان طبیعی به همراه شخصیت‌های متحرک و تلفیق گفتار برای ارتباط با کاربران استفاده می‌کند. کاربران می‌توانند با برنده‌ی جایزه‌ی لوبنر سال ۲۰۰۷ از طریق نوشتن و یا بازشناسی گفتار گفتگو کنند. این سامانه از واژه‌نامه‌ی لغوی شبکه‌ی لغت[۳] استفاده میکند[۹].

Elbot ۵.۲

Elbot برنده‌ی جایزه باهوش‌ترین و سرگرم‌کننده‌ترین سامانه‌ی گفتگوگر در مسابقات چالش Chat-terbox سال ۲۰۰۳، موفق به دریافت جوایز بسیار دیگری نیز شده است. در مسابقه‌ی جایزه لوبنر سال ۲۰۰۸ این سامانه موفق به دریافت جایزه برای متقاعد کردن ۳ داور از مجموع ۱۲ داور به انسان بودنش شد. جالب اینجاست که اگر این سامانه موفق به فریب دادن تنها یک داور دیگر می‌شد این مقدار از ۳۰ درصد داوران که مرز تعیین شده توسط آلن تورینگ برای بازی تقلید بود فراتر می‌رفت و سامانه جایزه بزرگ را از آن خود می‌کرد.

سازندگان این سامانه از طریق یک فناوری که خودشان توسعه داده‌اند موفق به ساخت این سامانه شده‌اند. هدف از ساخت این فناوری که تعامل زبان طبیعی[۴] نام‌گذاری شده توانا کردن فناوری به

---

[۳] WordNet

[۴] Natural Language Interaction

فکر کردن بوده است. سازندگان قصد ایجاد انقلابی در ارتباط میان انسان و دستگاه‌های ساخته شده توسط بشر از جمله رایانه‌ها تلفن‌های همراه و حتی تلوزیون‌های هوشمند را داشته‌اند[۱۰]. متاسفانه اطلاعات دیگری از نحوه‌ی عملکرد این فناوری و یا نحوه‌ی عملکرد Elbot موجود نیست.

Do-Much-More ۶.۲

Do-Much-More طراحی شد تا باهوش‌تر، آگاه‌تر و طبیعی‌تر از بقیه سامانه‌های گفتگوگر ظاهر شود و این ویژگی‌ها تنها با استفاده از ریخت‌شناسی زبان انگلیسی، مقداری دانش عمومی و مقداری آگاهی از نحوه‌ی استفاده از لغات تحقق پیدا کرد[۱۱]. ویژگی مهم این سامانه عدم شرکت در مکالمات تخصصی و نگه داشتن بحث در زمینه غیر تخصصی است و همین امر باعث می‌شود سامانه نیاز به دانستن مطالب تخصصی نداشته باشد. این سامانه همواره سعی می‌کند تا بحث را جذاب نگه داشته تا مخاطب همواره مشتاق به ادامه‌ی بحث باقی بماند.

نسخه‌ای از Do-Much-More که موفق به دریافت جایزه لوبنر سال ۲۰۰۹ شد با نسخه‌ی عادی آن تفاوت خاصی نمی‌کند و این به این معناست که نویسندگان این سامانه از دانش هیچ متخصصی برای افزایش اطلاعات سامانه استفاده نکرده‌اند و Do-Much-More همواره با مهارت خود در زمینه‌ی نگهداری بحث در حالت غیر تخصصی مخاطب را مشغول نگه می‌دارد.

Suzette و Rosette ۷.۲

در سال ۲۰۱۰ مجموع امتیازهای A.L.I.C.E بیشتر از Suzette بود اما Suzette برنده شد زیرا توانست یک داور را فریب دهد. از ویژگی‌های منحصر به فرد و هیجان‌انگیز این سامانه میتوان به شبیه‌سازی احساسات اشاره کرد. از این رو Suzette میتواند ناراحت و یا رنجانده شود و حتی گاها مکالمه‌ی خود را به دلیل رنجانده شدن از دست بازجویش تمام می‌کند[۱۲].

Suzette دانش خود را درون فایل‌های AIML ذخیره می‌کند. اما به اینجا محدود نمی‌شود و دادگانی از متغیرها سه‌گانه‌ها و مفاهیم نگهداری میکند[۱۳].

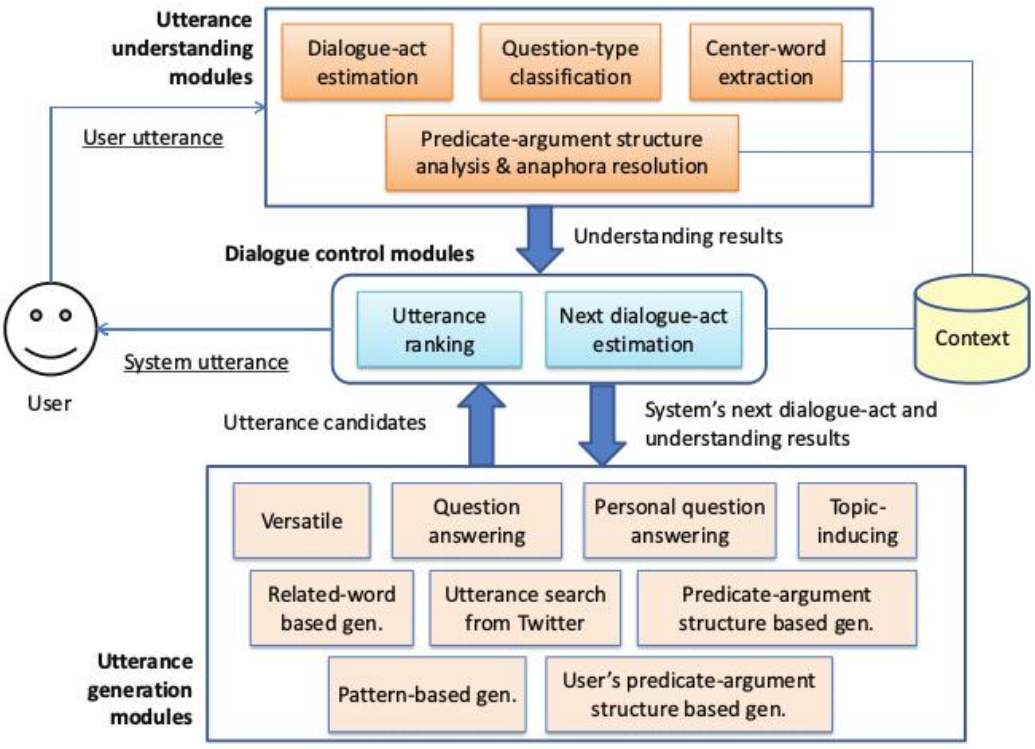
علی‌رغم این که Rosette برنده‌ی جایزه‌ی لوبنر در سال ۲۰۱۱ است، متاسفانه اطلاعات زیادی از آن در دسترس نیست. اما اطلاعاتی موجود است که بیان میکند ساختار و نحوه‌ی عملکرد آن شبیه به Suzette است.

Mitsuku ۸.۲

این سامانه برنده‌ی جایزه‌ی لوبنر در سال‌های ۲۰۱۳ و ۲۰۱۶ بوده در حال حاضر بهترین سامانه‌ی گفتگوگر موجود در دنیا به حساب می‌آید. سامانه‌ی پیشنهاد شده کاملاً از ماژول‌های مبتنی بر تکنیک‌های پردازش زبان طبیعی ساخته شده است. نتایج آزمایشگاهی نشان می‌دهد که این سیستم

از سیستم‌هایی با تعداد ۱۴۹ هزار قانون دست نویس هم به مراتب بهتر عمل می‌کند. همچنین لازم به ذکر است که این سامانه دامنه‌باز[۵] بوده و به سوالات هر زمینه می‌تواند پاسخگو باشد. بدیهی است که سامانه‌ی دامنه‌باز سامانه‌ی قوی‌تری از سامانه‌ی وظیفه‌محور[۶] به حساب می‌آید. در سامانه‌های وظیفه‌محور سازنده می‌تواند اطلاعاتی مربوط به وظیفه‌ی مورد نظر را گردآوری کرده و به سامانه ارائه دهد اما این کار در سامانه‌های دامنه‌باز امکان‌پذیر نمی‌باشد[۱۵].

Mitsuku بر این باور است که با توجه به حجم انبوه اطلاعات موجود در سطح وب می‌توان کار نگهداری اطلاعات را به وب سپرده و مطمئن بود که حداقل اطلاعاتی برای هر ورودی کاربر موجود است. البته این اطلاعات به سختی بدست می‌آیند. دلیل این امر پرآشوب[۷] بودن اطلاعات مخصوصا در شبکه اجتماعی‌هایی همچون توییتر است. شکل ۱.۲ ساختار سامانه‌ی Mitsuku را نشان می‌دهد. این ساختار در ادامه توضیح داده خواهد شد.



شکل ۱.۲: ساختار سامانه‌ی گفتگوگر Mitsuku

[۵] open-domain
[۶] task-oriented
[۷] noisy

ساختار Mitsuku تشکیل شده از سه قسمت فهم سخن، کنترل گفتگو و ایجاد سخن است. نظریه گفتمان[۸] بیان میدارد که مهم‌ترین ارکان مکالمه خیال (ساختار ذهنی)، مبحث (وضعیت توجه) و محتوا (ساختار زبانی) هستند. سازندگان Mitsuku سعی در ساخت سیستمی که توانایی فهم و تولید این سه مفهوم را داشته باشد داشته‌اند.

۱.۸.۲ واحد فهم سخن

همان طور که در شکل ۱.۲ مشخص است این واحد از زیر واحدهای تخمین گفتگو_عمل[۹]، دسته‌بندی نوع سوال[۱۰]، استخراج واژه‌ی مرکزی[۱۱] و تحلیل ساختار پیش‌فرض_استدلال[۱۲] تشکیل شده است که در ادامه هر یک را بررسی می‌کنیم.

تخمین گفتگو_عمل

برا فهم نوع گفتگو از این روش استفاده می‌شود. در این سامانه از یک ماشین بردار پشتیبان برای حل این مسئله به صورت نظارت‌شده استفاده شده است. ویژگی‌های مورد استفاده نیز غالبا n-gram ها بوده‌اند.

دسته‌بندی نوع سوال

مسئله‌ی دسته‌بندی نوع سوال مسئله تخصیص یک برچسب به هر سوال ورودی کاربر است این مسئله در طول تاریخ به روش‌های مبتنی بر قانون، مبنتی بر یادگیری ماشین و ترکیبی حل شده است[۱۴]. برچسب تخصیص داده شده به هر سوال معمولا از بین مجموعه‌های درشت دانه و یا ریزدانه‌ی انتخاب شده توسط سازنده‌ی سیستم انتخاب می‌شود.

در این سامانه از یک دسته‌بند مبتنی بر رگرسیون منطقی برای دسته‌بندی نوع سوال استفاده شد که به دقت ۵.۹۲ بسنده کرد.

استخراج واژه‌ی مرکزی

کلمه‌ی واژه‌ی مرکزی به یک عبارت اسمی[۱۳] در جمله اطلاق می‌شود که موضوع مکالمه را مشخص می‌کند. این طور فرض می‌شود که هر سخن بیشتر از یک واژه‌ی مرکزی ندارد.

---

[۸] Discourse theory
[۹] Dialog-act estimation
[۱۰] Question-type classification
[۱۱] Center-word extraction
[۱۲] predicate-argument structure analysis
[۱۳] Noun phrase

در این سامانه برای واکشی واژه‌ی میانی از یک میدان تصادفی[۱۴] شرطی استفاده شد تا مسئله به صورت نظارت‌شده حل شود. در این روش دیگر نیازی به ساختن درخت تجزیه[۱۵] نبوده و عبارات اسمی به صورت مستقیم با دنباله‌هایی از کلمات به میدان تصادفی شرطی داده شدند. برای استخراج ویژگی از ویژگی‌هایی مانند کلمات به خودی خود و برچسب بخش گفتار[۱۶] استفاده شد.

تحلیل ساختار پیش‌فرض-استدلال

در این قسمت پیش‌فرض ها و استدلال‌هایشان شناخته میشوند. پیش‌فرض می‌تواند یک فعل یک صفت و یا یک فعل مشترک باشد. استدلال‌ها هم عبارات اسمی متناظر با موارد در دستورزبان موردی[۱۷] می‌باشند.

۲.۸.۲ واحد کنترل گفتگو

این واحد از دو زیر واحد تخمین دیالوگ-عمل بعدی[۱۸] رتبه بندی سخن[۱۹] تشکیل شده است که به اختصار بیان خواهند شد.

تخمین دیالوگ-عمل بعدی

برای تخمین دیالوگ-عمل بعدی سه دیالوگ-عمل قبلی به دسته‌بند داده شده و از آن خواسته می‌شود تا دیالوگ عملی بعدی را حدس بزند. در این سامانه دقت دسته‌بند تخمین دیالوگ-عمل بعدی ۲۸ درصد گزارش شده که با وجود پایین بودن با توجه به دقت ۱۵ درصدی سیستم پایه، منطقی به نظر می‌رسد.

رتبه‌بندی سخن

سازندگان Mitsuku بر این باورند که استفاده از میزان ارتباط و وابستگی منطقی‌ترین گذینه‌ی ممکن برای بدست آوردن خروجی محسوب میشود. از این رو آن‌ها دادگانی با نمونه‌های واقعی و جعلی از گفتگوها ساخته و به یک ماشین بردار پشتیبان نشان میدهند تا بتواند تفاوت بین پاسخ درست و غلط را تشخیص دهد. برای استخراج ویژگی در این سامانه از ارتباط دو به دوی کلمات سخن‌های پیشین با کلمات سخن کنونی استفاده شده است. برای مثال اگر سخن کنونی دارای m کلمه و سخن قبل دارای n کلمه میباشد یک بردار $m * n$ به ماشین بردار پشتیبان داده می‌شود.

---

[۱۴] Conditional random field
[۱۵] Parse tree
[۱۶] POS tag
[۱۷] Case grammer
[۱۸] Next dialogue-act estimation
[۱۹] Utterance ranking

با این اوصاف سیستم مورد نظر به دقت ۷.۶۶ رسیده است در حالی که دقت سیستم پایه ۵۰ درصد می‌باشد.

۳.۸.۲ واحدهای تولید سخن

۹ واحد برای تولید سخن در این سامانه موجود است. همه کاره[۲۰]، واحد جواب‌دادن به سوالات[۲۱] و واحد جواب دادن به سوالات شخصی[۲۲] واحدهایی بودند که بر پایه‌ی دیالوگ_عمل و نوع سوال عمل می‌کردند. از طرف دیگر واحدهای القاء موضوع[۲۳]، کلمه‌ی مرتبط[۲۴]، توییتر[۲۵] و تحلیل ساختار پیش‌فرض استدلال[۲۶] بر پایه‌ی واژه‌ی میانی و در نهایت الگو[۲۷] و تحلیل ساختار پیش‌فرض استدلال کاربر[۲۸] بر پایه‌ی متن خام و تحلیل ساختار پیش‌فرض استدلال ورودی کاربر کار می‌کردند. توجه کنید که در تمامی این واحدها از دیالوگ_عمل بعدی کاربر برای تحلیل استفاده شده است.

با توجه به طولانی شدن این قسمت و توصیفات زیاد ۹ واحد ذکر شده در بالا در مقاله اصلی[۱۵] برای مطالعه‌ی بیشتر در مورد هر واحد خواننده را به این مقاله ارجاع می‌دهیم.

---

[۲۰] Versatile
[۲۱] Question answering
[۲۲] Personal question answering
[۲۳] Topic-inducing
[۲۴] Related-word
[۲۵] Twitter
[۲۶] PAS
[۲۷] Pattern
[۲۸] User PAS

فصل ۳

روش پیشنهادی

شاید بعد از بررسی سامانه‌های گفتگوگر موجود این موضوع واضح به نظر برسد که استفاده از قوانین دست‌نویس کاری سخت و طاقت فرساست و در نهایت امر به دلیل دقت پایین نتیجه نهایی، روش مناسبی محسوب نمی‌شود. از طرف دیگر دیدیم که سامانه‌ای مانند Mitsuku به راحتی از روش‌های موجود در پردازش زبان طبیعی و داده‌های موجود در وب استفاده کرده و سیستمی با کیفیت بسیار بالا تحویل می‌دهد. اما حتی Mitsuku هم سامانه‌ی پیچیده‌ای محسوب میشود. سوال اصلی اینجاست که آیا می‌توان با استفاده از روش‌های مبتنی بر یادگیری ماشین سامانه‌ای حتی ساده‌تر و قدرتمندتر پیاده سازی کرد؟

این روزها استفاده از روش‌های مبتنی بر یادگیری ماشین برای حل انواع مسائلی که ماشین‌ها تا کنون موفق به حل آن ها نبودند متداول شده است. دلیل این امر به وجود آمدن سخت‌افزارهای قدرتمند و به طور خاص واحد پردازش گرافیکی[۱] و ظهور داده‌های بسیار است که در گذشته موجود نبودند. شبکه‌های عصبی که نیازمند داده‌های زیاد و حجم بالای محاسبات هستند تا قبل از این دوره قابلیت عمیق شدن نداشتند. اما امروزه شبکه‌های عصبی بسیار عمیق متداول شده‌اند و رشته‌ی مطالعاتی جدیدی به نام Deep learning به وجود آمده است که توجه خود را بر روی این نوع از شبکه‌ها معطوف می‌کند.

برای مسائلی که نیاز به وجود مفهوم زمان در ورودی آن‌ها احساس می‌شود LSTM ها نامزدهای مناسبی به شمار می‌آیند. مسئله‌ی مکالمه نیز ورودی‌هایی دارد که وابسته به زمان هستند. اگر هر کلمه را عضوی از یک دنباله تصور کنیم یک ورودی کاربر را میتوان با یک دنباله معرفی کرد. از طرف دیگر خروجی مورد نظر نیز باید دنباله‌ای از کلمات و در نتیجه یک دنباله باشد. بنابراین باید از یک مدل دنباله به دنباله که در قسمت ۵.۳.۱ معرفی شد استفاده کنیم. اما به دلایلی که در قسمت ۶.۳.۱ بیان شد نیاز به وجود یک مکانیزم توجه احساس می‌شود.

بنابراین در ساده‌ترین حالت ممکن برای پیاده‌سازی این سامانه ابتدا دادگان موجود در دانشگاه

[۱] Graphics Processing Unit

Cornell که در مورد دیالوگ‌های فیلم‌های تاریخ سینماست دریافت و برای ورود به الگوریتم آماده شد. سپس الگوریتم Word2vec روی داده‌های موجود تعلیم داده شد تا هر کلمه به یک بردار تبدیل شود. سپس کلمات وارد مدل دنباله به دنباله همراه با مکانیزم توجه شده و خروجی‌های مورد نظر به صورت دنباله‌ای از بردارهای کلمات بدست آمد. سپس هر بردار خروجی به نزدیک‌ترین کلمه‌ی موجود در فرهنگ لغت نگاشته شد تا دنباله‌ی کلمات خروجی بدست آید. این مدل از مرجع شماره [۱۶] الهام گرفته است.

به سامانه‌هایی که به صورت یک واحد عمل کرده و قابل آموزش به صورت یک جا می‌باشند سیستم‌های پشت‌سرهم[۲] گفته می‌شود. سامانه‌ی پیشنهادی یک سامانه‌ی پشت‌سرهم بوده و به عنوان یک واحد قابل آموزش می‌باشد. توجه کنید که در سامانه‌هایی که در گذشته وجود داشت این اتفاق رخ نمی‌داد و هر جزء باید به تنهایی آموزش می‌دید. از جمله‌ی این سامانه‌های تجمعی[۳] می‌توان به Mitsuku اشاره کرد که تعداد زیادی از واحدها را کنار یکدیگر جمع میکند.

اما نکته‌ای که در مورد تبدیل کلمات به بردار وجود دارد این است که نمی‌توان همه کلمات موجود در متن آموزشی را درون فرهنگ لغت قرار داد. دلیل این امر هم این است که کلمات با تکرار بسیار پایین نمی‌توانند جایگاه خود را در فضای برداری به درستی پیدا کنند و تبدیل آن‌ها به بردار کاری بی‌ثمر می‌باشد. پس باید این کلمات را کنار گذاشته و به تمام این نوع کلمه یک بردار خاص مانند بردار «شناخته‌نشده» تخصیص دهیم. اما این روشی نیز مشکل خاص خود را دارد. سامانه ممکن است جواب «شناخته‌نشده» دهد و یا از این عبارت در جواب خود استفاده کند.

برای مشکل بالا می‌توانیم راه حل خود را مقداری تغییر دهیم تا دیگر شاهد این مشکل نباشیم. بدین منظور به جای ساخت سامانه‌ای کلمه محور سامانه‌ای کاراکتر محور خواهیم ساخت. توجه کنید که سامانه‌ی ساخته شده توانایی نگاشت هر دنباله‌ای به هر دنباله‌ی دیگر را دارد و می‌دانیم که می‌توان به جای استفاده از دنباله‌ی کلمات از دنباله‌ی کاراکترها استفاده کرد. حسن این روش این است که دیگر نشانه‌ی شناخته نشده ای وجود نخواهد داشت. زیرا تعداد کاراکترها محدود و تعداد تکرار هر کاراکتر، حتی کاراکترهای کمیاب، بسیار زیاد است و هر کاراکتر جایگاه خود را درون فضای برداری خواهد یافت.

---

[۲] End-to-end

[۳] Ensembling

فصل ۴

آزمایشات

همان‌طور که در قسمت‌های پیش بیان شد آزمایشات مربوط به یک سامانه‌ی گفتگوگر به این ترتیب است که سامانه‌ی گفتگوگر در مقابل چند داور به نحوی که داور به صورت تصادفی با انسان و سامانه بازی می‌کند قرار می‌گیرد و سامانه باید بتواند انسان را فریب دهد تا باور کند در حال سخن گفتن با یک انسان دیگر است.

آزمایشات مربوط به سامانه‌ی گفتگوگر نوشته شده در این نوشتار به دلیل کمبود امکانات نمی‌توانست دقیقا مشابه به آزمایشات انجام شده دز مسابقه‌ی جایزه‌ی لوبنر باشد. از این رو دو مکالمه با سامانه انجام و نتیجه‌ی آن‌ها گزارش شد. مکالمات را در جداول ۱.۴ و ۲.۴ مشاهده میکنید.

در اینجا میتوان در مورد نقطه ضعف های این سامانه نیز چند نکته بیان کرد: نکته اول این است که مکالمات به لطف مکالمات موجود در فیلم‌های هالیوودی که دادگان ما را تشکیل میدادند به شدت دراماتیک هستند. بدین منظور می‌توان از دادگان دیگر نیز استفاده کرد. به طور مثال دادگاه مکالمات توییتر احتمالا دادگان مناسبی برای متعادل کردن این شرایط محسوب می‌شود. نکته دوم این است که این سامانه مکالمات قبلی را فراموش می‌کند و تنها به آخرین گفتگو اهمیت می‌دهد.

Table 4.1: Chatbot based on deep learning. Conversation 1.

Human	Chatbot
You are a monster!	I am...
I'm sorry	You don't need to boy
I'm a girl	You done?
Did you change your hair?	No.
No?	No, I won't.
Who?	Paul Owen

Table 4.2: Chatbot based on deep learning. Conversation 2.

Human	Chatbot
What	Don’t bother.
You’re being sassy	I need.
What do you need?	I need ya help ya.
Okay.	What do you want?
You’re just talking gibberish.	And what are you talking about?
About life.	And what about it?
life is beautiful	Is good.
Where do you live?	Up.
As in heaven?	In his apartment.

برای این که این اتفاق را شاهد نباشیم می‌توانیم هر بار وضعیت سلول قبلی رمزگذار را دور نریخته و در ابتدا وضعیت را به جای صفر برابر با وضعیت قبلی قرار دهیم. نکته‌ی سوم این است که سامانه شخصیت خاص خود را ندارد. برای این کار دو راه حل پیشنهاد می‌شود. روش اول هر بار به خروجی رمزگذار یک بردار ثابت حاوی اطلاعات شخصیتی سامانه اضافه می‌کند تا سامانه همواره شخصیت خود را به یاد داشته باشد. روش دوم به این ترتیب است که کلا سامانه را با استفاده از مکالمات یک شخصیت خاص آموزش می‌دهیم.

فصل ۵

نتیجه گیری

۱.۵ گذشته

در این نوشتار بهترین سامانه‌های گفتگوگر جهان از سال ۲۰۰۰ تا کنون بررسی شدند. بدیهی است که این سامانه‌ها هر سال در حال پیشرفت بوده و هر بار که یک سامانه بخواهد با سامانه‌های قبلی رقابت کند باید ایده‌ی جدیدی را وارد عرصه سامانه‌های گفتگوگر بکند. سامانه‌ی Ella با پیاده‌سازی شخصیت و استفاده از زبان طبیعی، سامانه‌ی Jabberwacky با استفاده از تکنیک‌های تطبیق الگو، سامانه‌ی Elbot با وارد کردن طنز، سامانه‌ی Suzette با وارد کردن احساسات و سامانه‌ی Mitsuku با استفاده گسترده از پردازش زبان طبیعی مستقل از عملکرد از جالب‌ترین سامانه‌های گفتگوگر به شمار می‌آیند.

۲.۵ حال

در ادامه روشی هوشمندتر و ساده‌تر برای پیاده‌سازی سامانه‌های گفتگوگر پیشنهاد شد. این روش بر پایه‌ی روش‌های یادگیری ماشین سعی در ساخت سامانه‌های به قدرت بهترین سامانه‌های موجود در این زمینه می‌کند. اما باید اعتراف کرد که هنوز این سامانه مشکلاتی را پیش روی خود می‌بینند. از جمله‌ی این مشکلات می‌توان به عدم نگهداری اطلاعات بیان شده توسط کاربر، پاسخ یکسان به ورودی یکسان، نداشتن شخصیت و نیاز به داده‌ی آموزشی فراوان اشاره کرد.

برای رفع مشکلات بالا می‌توان از چند گام نه چندان دشوار استفاده کرد. به طور مثال برای رفع مشکل داده‌های کم می‌توان از چند دادگان به طور همزمان استفاده کرد. البته متاسفانه در زبان فارسی چنین دادگانی موجود نیست. برای رفع مشکل نداشتن شخصیت دو راه پیشنهاد می‌شود: اول می‌توان هر بار یک بردار خاص حاوی مشخصات سامانه به خروجی رمزگذار اضافه کرد و دوم می‌توان تنها مکالمات یک فرد را برای آموزش سامانه استفاده کرد.

همچنین می‌توان گام‌های دیگری در راستای بهبود سامانه برداشت. از جمله این گام‌ها میتوان به توانا کردن سامانه در گرفتن بازخورد از طرف کاربر اشاره کرد.

۳.۵ آینده

در ادامه‌ی راه می‌توان از شبکه‌ای مانند شبکه‌ی قشر جلویی[۱] مغز استفاده کرد تا متغیرهای مختلف و میزان ارتباطشان با هم شناخته شده و در جواب سامانه وارد شوند. برای این منظور می‌توان هر بار که دو کلمه کنار یکدیگر ظاهر می‌شوند ارتباط میان آن‌ها را در شبکه‌ی شبیه‌ساز قشر جلویی مغز تقویت کرد و این تغییرات را در این شبکه انتشار داد. حال هر بار که شبکه جواب می‌دهد میتوان به واحدهای فعال در شبیه‌ساز نگاه کرده و اگر خروجی شبکه‌ی اصلی شامل این واحدها نبود شبکه را جریمه کرد.

---

[۱] cortex Prefrontal

Bibliography

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[2] Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In EMNLP (Vol. 14, pp. 1532-1543).

[3] https://www.tutorialspoint.com/aiml/aiml_basic_tags.htm, access date: 2017/07/05

[4] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[5] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[6] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[7] https://www.chatbots.org/chatbot/ella, access date: 2017/07/07

[8] http://www.jabberwacky.com/j2about, access date: 2017/07/07

[9] https://en.wikipedia.org/wiki/Ultra_Hal_Assistant, acess date: 2017/07/07

[10] http://www.elbot.com/chatterbot-elbot/, acess date: 2017/07/07

[11] http://www.worldsbestchatbot.com/Do_Much_More, acess date: 2017/07/07

[12] https://pl.wikipedia.org/wiki/Suzette_(chatbot), access date: 2017/07/07

[13] Abdul-Kader, S.A. and Woods, J., 2015. Survey on chatbot design techniques in speech conversation systems. Int. J. Adv. Comput. Sci. Appl.(IJACSA), 6(7).

[14] Razzaghnoori, M., Sajedi, H. and Khani Jazani, I. Question Classification in Persian using Word Vectors and Frequencies. In Cognitive System Research (submitted)

[15] Higashinaka, R., Imamura, K., Meguro, T., Miyazaki, C., Kobayashi, N., Sugiyama, H., Hirano, T., Makino, T. and Matsuo, Y., 2014. Towards an open-domain conversational system fully based on natural language processing. In COLING (pp. 928-939).

[16] Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I. and Hinton, G., 2015. Grammar as a foreign language. In Advances in Neural Information Processing Systems (pp. 2773-2781).

Abstract

Today, a great number of researchers are solving the different kinds of problems which were impossible to solve by machines, using deep learning algorithms. The one problem which seems to be possible to be solved using deep learning is creating chatbots. A chatbot is a system that has the ability to chat with a human being using natural language. In other words, a chatbot receives input in the form of natural language and outputs a statement in natural language form. Previously, these chatbots were being made using hand-crafted rules, natural language processing, and classic machine learning techniques. Today, however, it might be the time to consider new techniques like deep learning as the science behind creating chatbots.

In this research, existing chatbots will be studied and the related information about them will be gathered. Finally, a chatbot will be created using deep neural networks and its strengths and flaws will be discussed.

College of Science
School of Mathematics, Statistics, and Computer Science

A Survey on Existing Chatbots and an Implementation of a Chatbot Based on Machine Learning

Mohammad Razzaghnoori

Supervisor: Dr. Hediyeh Sajedi

A thesis submitted to Graduate Studies Office
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

2017