بسم الله الرحمن الرحیم

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته علوم کامپیوتر

گرایش علوم تصمیم و مهندسی دانش

تشخیص اخبار جعلی در رسانه های اجتماعی

استادان راهنما

دکتر مینا مسعودی فر و دکتر داوود ذبیح زاده

استاد مشاور

دکتر مرتضی جعفر زاده

پژوهشگر:

زهرا رستمی

شهریور۱۴۰۰

باسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان‌نامه کارشناسی ارشد

فرم ۱۱۳-ت

جلسه دفاع از پایان‌نامه آقای /خانم **زهرا رستمی** دانشجوی رشته علوم کامپیوتر گرایش علوم تصمیم و مهندسی دانش به شماره دانشجویی ۹۷۱۳۱۸۵۰۹۳ با عنوان:

تشخیص اخبار جعلی در رسانه های اجتماعی

در مورخه ..... در دانشکده ریاضی و علوم کامپیوتر تشکیل و توسط هیات داوران مورد ارزشیابی قرار گرفت و نمره ..... برابر درجه ..... برای آن تعیین گردید .

به‌این ترتیب از این تاریخ آقای/ خانم زهرا رستمی به عنوان کارشناس ارشد در رشته مذکور شناخته می‌شود .

موارد ارزشیابی	موارد	حداکثر نمره	نمره کسب شده
۱- کیفیت نگارش	رعایت اصول نگارش انسجام در تنظیم بخشهای مختلف ،کیفیت تصاویر ، جداول و اشکال ،تنظیم فهرست ها ، منابع و ماخذ	٤	
۲- کیفیت علمی	بررسی تاریخچه و سابقه تجربی و نظری موضوع انسجام منطقی دربخش های مختلف پایان نامه ،ابتکار و نوآوری ،اهمیت و ارزش علمی پایان نامه ،استفاده از منابع معتبر و جدید ،کیفیت تجزیه و تحلیل یافته ها و نتیجه گیری ،روشن بودن روش کار ،هدف ها و فرضیه های تحقیق ،جدید بودن روش تحقیق	۱۰	
۳-کیفیت ارایه در جلسه دفاع	تسلط بر موضوع و بیان واضح وتفهیم آن ،توانایی در پاسخگویی به سوالات مطرح شده در جلسه ، رعایت زمان ارایه , روش ارائه	٤	
٤-ارزشیابی گزارشات	گزارش های دوره ای پیشرفت کار(حداقل ٤مورد)	۱	
٥- خروجی پایان نامه	مقاله مستخرج از پایان نامه: این نمره به صورت زیر اختصاص می یابد ۱) چکیده کنفرانسی هر مورد ۰/۲۵ نمره تا سقف۰/۵ نمره ۲) مقاله کامل در مجموع مقالات همایشهای معتبر یا مقاله در مجلات علمی-ترویجی معتبر پذیرفته شده یا چاپ شده هر مورد ۰/۵ نمره تا سقف ۱نمره ۳)مقاله پذیرفته شده یا چاپ شده در مجلات علمی پژوهشی معتبر ۱ نمره ٤)مقاله ارسال شدهبه مجلات علمی پژوهشی معتبرهر مورد ۲۵/. نمره تا سقف ۰/۵ نمره ٥) دستگاه ساخته شده دارای گواهی ثبت اختراع یا به سفارش سازمان ها تا سقف ۱ نمره ٦) دستگاه ساخته شده کاربردی که به تایید رئیس دانشکده رسیده باشد تا سقف ۰/۵ نمره	۱	
جمع			

درجه معادل کسب شده: (از ۱۹تا ۲۰ عالی☐ از۱۸تا ۱۸/۹۹بسیارخوب ☐ از ۱٦تا ۱۷/۹۹خوب ☐ از ۱٤ تا ۱۵/۹۹قابل قبول☐ کمتر از ۱٤ غیر قابل قبول☐)

مشخصات هیات دوران

ردیف	نام و نام خانوادگی	سمت	مرتبه علمی	محل کار	امضـــــــــــا
۱	دکتر مینا مسعودی فر	استاد راهنما	استادیار	دانشگاه حکیم سبزواری	
۲	دکتر داوود ذبیح زاده	استاد راهنما	استادیار	دانشگاه حکیم سبزواری	
۳	دکتر مرتضی جعفر زاده	استاد مشاور	استادیار	دانشگاه حکیم سبزواری	
۴	دکتر...	استاد داور	استادیار	دانشگاه حکیم سبزواری	
۵	دکتر غلامرضا مقدسی	نماینده تحصیلات تکمیلی	استادیار	دانشگاه حکیم سبزواری	

امضا

مدیر گروه

امضا

رئیس دانشکده

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک، کوشش خویش و بهره گیری از دانش استادان و سرمایه‌های مادی و معنوی این مرز و بوم، توشه‌ای از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد می‌کنم که در به کارگیری دانش خویش، همواره بر راه راست ودرست گام بردارم. خداوند بزرگ، شما شاهدان، دانشجویان و دیگر حاضران را به عنوان داورانی امین گواه می گیرم که ازهمه دانش و توان خود برای گسترش مرزهای دانش بهره گیرم و از هیچ کوششی برای تبدیل جهان به جایی بهتر برای زیستن، دریغ نورزم. پیمان می‌بندم که همواره کرامت انسانی را در نظر داشته باشم و همنوعان خود را در هر زمان و مکان تا سر حد امکان یاری دهم. سوگند می خورم که در به کارگیری دانش خویش به کاری که با راه و رسم انسانی، آیین پرهیزگاری، شرافت و اصول اخلاقی برخاسته از ادیان بزرگ الهی، به ویژه دین مبین اسلام، مباینت دارد دست نیازم. همچنین در سایه اصول جهان شمول انسانی و اسلامی، پیمان می‌بندم از هیچ کوششی برای آبادانی و سرافرازی میهن و هم میهنانم فروگذاری نکنم و خداوند بزرگ را به یاری طلبم تا همواره در پیشگاه او و در برابر وجدان بیدار خویش و ملت سرافراز، بر این پیمان تا ابد استوار بمانم.

نام و نام خانوادگی: زهرا رستمی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب زهرا رستمی به شماره دانشجویی ۹۷۱۳۱۸۵۰۹۳ دانشجوی رشته علوم کامپیوتر مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: زهرا رستمی

تاریخ و امضا:

مجوز بهره برداری از پایان‌نامه

بهره برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره برداری از این پایان‌نامه برای همگان بلامانع است.

☐ بهره برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استادان راهنما: دکتر مینا مسعودی فر

دکتر داوود ذبیح زاده

تاریخ و امضا:

تقدیم به:

پدر و مادر عزیزم

و

همسر مهربانم

سپاس خداوندگار حکیم را که با لطف بی کران خود، آدمی را زیور عقل آراست.

در آغاز وظیفه خود می دانم از زحمات بی دریغ استاد راهنمای خود، سرکار خانم دکتر مینا مسعودی فر و جناب آقای دکتر داوود ذبیح زاده صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمی رسید.

از جناب آقای دکتر مرتضی جعفرزاده که زحمت مطالعه و مشاوره این رساله را تقبل فرمودند و در آماده سازی این رساله، به نحو احسن اینجانب را مورد راهنمایی قرار دادند، کمال امتنان را دارم.

همچنین لازم می دانم از گروه پارسی‌لاتک در پاسخگویی به مشکلات کاربران کمال قدردانی را داشته باشم.

در پایان، بوسه می زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

زهرا رستمی

شهریور۱۴۰۰

فهرست مطالب

فهرست جداول

فهرست تصاویر

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:رستمی	نام:زهرا	ش. دانشجویی:۹۷۱۳۱۸۵۰۹۳
استادان راهنما: دکتر مینا مسعودی فر و دکتر داوود ذبیح زاده		
استاد مشاور: دکتر مرتضی جعفر زاده		
دانشکده ریاضی و علوم کامپیوتر	رشته: علوم کامپیوتر	گرایش: علوم تصمیم و مهندسی دانش
مقطع: کارشناسی ارشد	تاریخ دفاع:شهریور۱۴۰۰	تعداد صفحات: ۱۱۰
عنوان پایان‌نامه: تشخیص اخبار جعلی در رسانه های اجتماعی		
کلید واژه‌ها:اخبار جعلی، رسانه های اجتماعی،یادگیری ماشین،داده کاوی		

چکیده: دنیای دیجیتال به سرعت در حال گسترش است و علاوه بر مزایای آن، معایبی نیز به‌همراه دارد .مسائل مختلفی در دنیای دیجیتال وجود دارد که اخبار جعلی یکی از آن هاست.چون نظارتی در این دنیای دیجیتال وجود ندارد، افراد به راحتی می توانند اخبار جعلی را برای حمایت از یک سازمان یا شخص و یا برعلیه یک شخص یا یک سازمان منتشر کنند و شهرت آن را زیر سوال ببرند. هدف این پژوهش، تشخیص این‌گونه اخبار است که امروزه به یکی از مهم‌ترین و پرخطرترین مسائل در جوامع تبدیل شده‌است. تشخیص اخبار جعلی به‌چندین روش امکان‌پذیر است. روش‌های مبتنی بر یادگیری ماشین که از روش‌های سنتی استفاده می‌کنند و روش‌های مبتنی بر یادگیری‌عمیق که از شبکه‌های عصبی استفاده می‌کنند. در این پژوهش، روش های مبتنی بر یادگیری عمیق مورد مطالعه و در مجموعه‌داده مربوطه مورد بررسی و آزمایش قرار می‌گیرد.

پیش‌گفتار

در دنیای امروز اخبار با سرعت نور در حرکت است، رقابت زیادی بین مردم در تولید اخبار وجود دارد که منجر به تولید اخبار ساختگی یا همان اخبار جعلی می شود. در گذشته سرعت انتشار اخبار به‌دلیل استفاده از روش‌های سنتی مانند روزنامه و تلویزیون پایین‌تر بوده اما امروزه به‌دلیل استفاده از رسانه های اجتماعی مانند اینترنت سرعت انتشار بالا است، این اخبار عمدا و برای گمراه کردن خوانندگان نوشته شده‌است. هدف این پایان نامه شناسایی اخبار جعلی است. راه‌های زیادی برای شناسایی این اخبار وجود دارد که در این پایان‌نامه به معرفی برخی از آن‌ها پرداخته می‌شود.

این پایان نامه شامل ۴ فصل می‌باشد:

در فصل ۱، مقدمه ای در رابطه با اخبار جعلی و کاربردهای آن آورده شده و در ادامه توضیحاتی در رابطه با شبکه های عصبی ANN, CNN, RNN,LSTM ارائه شده است. همچنین کلیاتی در رابطه با مفاهیم به‌کاررفته در پژوهش و تحقیقات صورت گرفته در این زمینه بیان شده است. در ادامه نیز توصیفی در رابطه با مجموعه داده مورد استفاده و معیارهای ارزیابی ارائه شده است.

در فصل ۲، روش های مختلف برای شناسایی اخبار را مورد بررسی قرار داده و پژوهش‌ها و تحقیقاتی که در گذشته در این زمینه انجام شده است ارائه می‌شود.

در فصل۳، دو روش ساختار سلسله مراتبی سطح گفتمان و شبکه عصبی تمایزگر رویداد را مورد بررسی قرار داده و همچنین در رابطه با ابزارهای مورد استفاده توضیحاتی ارائه شده است.

در فصل ۴، به بیان نتایج تحقیق و روش های ارزیابی برای بررسی ساختار آن پرداخته می‌شود. همچنبن نتایج مقایسه دو روش مورد بررسی با ۲۰۰ تکرار اراهئه شده است.

فصل ۱

مقدمه

۱-۱ اخبار جعلی

رسانه های اجتماعی [۲] سهم عمده ای در آگاه سازی مردم از وقایع جهان دارد. استفاده از اینترنت امکان پخش اخبار جعلی را به همراه دارد، موضوع اخبار جعلی توجه مردم و جوامع آکادمیک را جلب کرده است، چنین اطلاعات نادرستی پتانسیل تاثیرگذاری بر افکار عمومی را دارد و فرصتی برای احزاب مخرب فراهم می‌کند تا نتایج رویدادهای عمومی مانند انتخابات را دستکاری کنند.با افزایش محبوبیت شبکه‌های اجتماعی، بیشتر مردم به جای رسانه‌های سنتی از رسانه‌های اجتماعی استفاده می‌کنند. بر همین اساس از رسانه های‌اجتماعی به دلیل کم‌هزینه تر بودن و دسترسی آسان‌تر برای پخش اخبار جعلی [۳] استفاده می‌شود [۱۲].

یک سند شامل دو بخش است: ناشر و محتوا.

- ناشر شامل مجموعه ای از ویژگی‌ها برای توصیف نویسنده ی اصلی ازجمله نام و سن و..است.
- محتوا شامل مجموعه ای از ویژگی هایی است که بیانگرمقاله‌ی خبری است و شامل عناوین،متن و... است.

درروش های سنتی برای تشخیص اخبار جعلی، تعدادی ویژگی دست ساز معرفی می‌شود و از روی آنها طبقه بند سنتی اقدام به شناسایی نوع خبر میکند، اما امروزه بااستفاده از روش های یادگیری عمیق [۴] ویژگی ها به طور خودکار از متن خبر استخراج می‌شود. برای تشخیص اخبار جعلی از روی ۱) متن خبر ۲) کاربری که خبر را انتشار داده ۳) نحوه انتشار خبر ۴) عکس الحاقی به خبر،استفاده می‌شود و یا می‌توان موارد فوق را به صورت ترکیبی استفاده کرد [۱۳].

---

[2]Social media [3]Fake news [4]Deep Learning

۱-۱-۱ تعریف مسئله

کشف اخبار جعلی در رسانه‌های اجتماعی چندین مشکل تحقیقاتی چالش برانگیز را به وجود می‌آورد. اول، اخبار جعلی به صورت عمدی برای گیج کردن خوانندگان نوشته می‌شوند، به‌گونه‌ای‌که شناسایی اخبار براساس محتوای اخبار ممکن نیست. بنابراین شناسایی اخبار با تکیه بر ویژگی های متنی مناسب نیست. دوم ، اطلاعات کمکی خاص مانند پایگاه دانش و مشارکت‌های اجتماعی کاربر، نیز باید برای بهبود روند تشخیص اخبار اضافه شوند. بهره برداری از این اطلاعات کمکی منجر به یک چالش مهم در کیفیت داده ها می‌شود اگرچه این اطلاعات باعث کمک به تشخیص اخبار می‌شود اما چگونگی استخراج ویژگی ها از هر روش و ترکیب آن ها خود یک مسئله چالش برانگیز است. مسئله را می توان به فرم زیر شرح داد : فرض کنید m مقاله خبری که شامل اطلاعات متن و تصویر است وجود دارد و داده ها به‌عنوان مجموعه‌ای از متن و تصویر به صورت رابطه (۱-۱) نشان داده می‌شود :

$$A = (A_j^T, A_j^I)_i^m \tag{۱-۱}$$

در این مسئله هدف این است که مشخص شود آیا مقالاتی که در A وجود دارد جعلی است یا خیر ؟ برچسب دهی نوع خبر به صورت y=۰،۱ است که ۱ نشان دهنده اخبار جعلی و ۰ نشان دهنده اخبار واقعی است. ویژگی های متنی و تصویری مربوط به مقالات خبری مجموعه $A = (A_j^T, A_j^I)_i^m$ به ترتیب به صورت X_i^T،X_i^I نمایش داده می‌شود. هدف از مسئله تشخیص اخبار جعلی ساخت مدل $f = (A_j^T, A_j^I)_i^m \in X \rightarrow Y$ برای استنباط نوع برچسب خبر مقالات خبری مجموعه A است. همان‌طور که گفته شد وظیفه مدل کشف اخبار جعلی این است که بررسی کند مقالات خبری A جعلی است یا خیر؟ و به صورت رابطه (۱-۲) نمایش داده می‌شود [۱۴، ۱۵].

$$f(A) = \begin{cases} ۰ & \text{if a is a piece of fake news} \\ ۱ & \text{otherwise} \end{cases} \tag{۲-۱}$$

۱-۲ توصیف اخبار جعلی

برای کار روی اخبار جعلی ابتدا باید مفهوم اخبار جعلی و راه‌های تشخیص آن را درک کرد، برای ساخت مدل‌های تشخیص اخبار ابتدا باید به توصیف آن پرداخت. اخبار جعلی تمایل به دستکاری و تنوع در موضوعات و سبک‌ها را دارند. بنابر گزارش ویکی پدیا می‌توان گفت اخبار جعلی عبارت است از یک نوع روزنامه‌نگاری یا تبلیغات دروغین که شامل اطلاعات غلط عمدی است که یا از طریق رسانه‌های سنتی چاپ‌شده و یا از طریق رسانه‌های اجتماعی آنلاین منتشر می‌شوند. اخبار جعلی از دو بخش تشکیل‌شده‌اند: اصالت و قصد. اصالت شامل حقایق نادرست محتوای اخبار جعلی است و قصد به این معنی

است که اطلاعات غلط باهدف فریب خواننده نوشته‌شده است [۱۴]. اخبار جعلی شامل رسانه های خبری سنتی و رسانه های اجتماعی آنلاین است که در ادامه به معرفی هر یک پرداخته می‌شود.

- **اخبار جعلی رسانه‌های خبری سنتی:** در طول زمان، اخبار جعلی ابتدا از روزنامه به تلویزیون و سپس به اخبار آنلاین و رسانه‌های اجتماعی تبدیل‌شده است. اخبار جعلی سنتی، ابتدا مخاطبین خود را با استفاده از آسیب‌پذیری آن‌ها مورد هدف قرار می‌دهد. دو عامل مهم که مخاطبین را نسبت به اطلاعات غلط آسیب‌پذیر می‌کند عبارت‌اند از:
  - واقع‌گرایی نمادین: مخاطبین تنها دیدگاه خود را به‌عنوان حقیقت قبول دارند [۱۶].
  - تعصب اطلاعاتی: مخاطبین تمایل دارند از دیدگاه‌های فعلی مطلع شوند. این تعصبات در انسان ذاتی است و دوست دارد اخبار جعلی را واقعی تلقی کند [۱۷].
- **اخبار جعلی رسانه‌های اجتماعی آنلاین:** به دلیل هزینه پایین تشکیل حساب کاربری رسانه‌های اجتماعی، خیلی از کاربران این رسانه‌ها ممکن است انسان واقعی نباشند و ربات باشند. این ربات‌ها که امروزه خیلی مورداستفاده قرار می‌گیرند حساب‌هایی در رسانه‌های اجتماعی هستند که توسط یک الگوریتم کامپیوتری پشتیبانی می‌شوند [۱۸].

۱-۳ روش های شناسایی اخبار جعلی

با توجه به آن چه در مقالات مروری آمده است، می‌توان گفت شناسایی اخبار جعلی به چند روش صورت می‌گیرد. یکی از آن ها روش‌های مبتنی بر یادگیری ماشین است، انواع مختلفی از الگوریتم های یادگیری ماشین وجود دارند که شامل الگوریتم های نظارت شده، بدون نظارت و یادگیری ماشین تقویتی هستند. این الگوریتم ها ابتدا توسط یک مجموعه داده آموزشی، آموزش داده می‌شوند سپس برای انجام کارهای مختلف مورد استفاده قرار می‌گیرند. برخی از روش های مبتنی بر یادگیری ماشین از روش های سنتی مانند: svm [۱۹] Naïve Bayes [۲۰] K-Nearest Neighbor [۲۱] و ... استفاده می کنند. روش های جدید بر مبنای یادگیری عمیق عمل می‌کنند، که می‌توان برای مثال شبکه های RNN [۲۲] CNN [۲۳] و... را نام برد. در ادامه، تعاریف مقدماتی لازم برای شناسایی اخبار جعلی با استفاده از روش‌های یادگیری عمیق مورد مطالعه قرار می‌گیرد.

۱-۴ تعاریف

کیسه کلمات : جملات را به صورت چند کلمه نشان می دهد و همه کلمات ذخیره می شوند، اما ترتیب نادیده گرفته می‌شود یعنی هرگونه اطلاعات مکانی در مدل کلمات، مانند وقوع مشترک کلمات کلیدی ثبت نمی‌شود. مدل کیسه های کلمات

معمولاً در روش های طبقه‌بندی اسناد استفاده می‌شود. روش آن به این‌صورت است که سندها به فرم برداری تبدیل می‌شوند. در صورت وجود کلمه خاص در جمله یک و درغیر این صورت صفر می‌شود به این صورت کوله ای از کلمات در ماتریس ساخته می‌شود، این ماتریس می‌تواند به الگوریتم های بعدی برای عملیاتی مانند طبقه بندی و خوشه بندی داده شود [۲۴].

n-gram : الگوریتم با توجه به کلمات داخل متن اقدام به شناخت کلمات می کندو می تواند از روی مجموعه آموزشی اقدام به یادگیری کند و داده های جدید را به صورت خودکار برچسب بزند. مثلا کلمه ی خوبی به تنهایی مثبت است اما اگر به صورت ترکیب کلمات باشد ممکن است دیگر مثبت نباشد برای مثال در جمله ی (آدم خوبی نیست) اگر از gram-۲ استفاده کند کلمه ی خوبی بار منفی دارد، در این جا می توانی از n-gram استفاده کرد یعنی مجموعه ای از n کلمه را باهم ترکیب کرد.که در این جا بسته به مقدار n تعداددویژگی یا بعد نیز تغییر می‌کند و هر ترکیب کلمات یک عدد در ماتریس ویژگی می‌شود.

TF-IDF : همان طور که می‌دانیم هر کدام از کلمات دارای وزن خاصی هستند که میزان اهمیت آن کلمه را در سند مشخص می‌کند. TF-IDF [۱] به معنای فراوانی وزنی کلمه کلیدی است و هدف آن نشان دادن اهمیت کلمه کلیدی مورد نظر از طریق مقایسه تعداد تکرار کلمات در متن با تعداد تکرار کلمات در تمام مستندات می‌باشد. به عبارت دیگر می‌توان گفت :

TF: به آن فرکانس مدت نیز می‌گویند و تعداد تکرار کلمات در متن را اندازه گیری می کند

TF(t) =تعداد دفعات حضور عبارت t در یک سند/تعداد کل اصطلاحات سند

IDF: به آن فرکانس سند معکوس نیز گفته می‌شود و میزان اهمیت یک اصطلاح را مشخص می‌کند

IDF(t)= (تعداد کل اسناد/تعداد اسنادی که عبارت t درآن قرار دارد) log [۲۵].

نرخ یادگیری : نرخ یادگیری [۲] اغلب با نماد α و گاهی اوقات با نماد η نمایش داده می‌شود و بیانگر سرعت (گام) به‌روزرسانی وزن‌ها است، به عبارت دیگر می‌توان گفت این نرخ، تعیین می‌کند که تا چه میزان اطلاعات بدست آمده ی فعلی بر اطلاعات قبل ترجیح داده شود. نرخ یادگیری که بیش از حد بالا باشد ممکن است باعث شود یادگیری ناپایدار باشد و بیش‌برازش اتفاق بیفتد یا اینکه یادگیری پایین‌تر از حد مطلوب باشد، نرخ یادگیری هایی که مقدار کمی داشته باشند نیز ممکن است باعث شود فرایند یادگیری طولانی شود و همگرایی شبکه برای مدت زیادی طول بکشد.مقدار صفر یعنی همان اطلاعات قبلی باقی بماند و مقدار یک یعنی اطلاعات جدید را ملاک قرار داده شود.

محوشدگی گرادیان : شبکه های عصبی مصنوعی در طی استفاده از روش های یادگیری مبتنی برگرادیان دچار مشکل محو شدگی گرادیان می شوند. در این روش ها به منظور به روز رسانی پارامترهای شبکه عصبی از گرادیان استفاده می شود، هر پارامتر با توجه به میزان تاثیری که در نتیجه نهایی شبکه داشته است تغییر می یابد به این صورت که مشتق جزیی تابع خطا نسبت به هر پارامتر در هر بار تکرار فرایند آموزش محاسبه می شود. مقادیر گرادیان با حرکت به سمت ابتدای شبکه به مرور کوچک می شود تا جایی که تغییرات وزن به صورت ناچیزی صورت می گیرد و به همین علت فرایند آموزش کند می شودو در حالات شدیدتر منجر به توقف فرایند آموزش می شود. این مشکل به دلیل عمق زیاد شبکه اتفاق می افتد. عمق زیاد در شبکه های عصبی پیش خور [۳] مانند (MLP,...) به دلیل لایه های زیاد است در حالی که در شبکه های عصبی بازگشتی به

[1] Term Frequency -Inverse Document Frequency [2] Learning Rate [3] feed forward neural network

دلیل گام های زمانی زیاد است چرا که در این شبکه ها هر گام زمانی مطابق با یک لایه ی شبکه های پیش خور است.

تفکیک جملات : اولین قدم در پردازش متن تفکیک جملات [1] است. یعنی جدا کردن کلمات جمله از هم و تبدیل هر کدام از این جملات به یک سری کلمه جداست.

one-hot-encoding: برای تبدیل و تغییر کلمات به فرمی که برای کامپیوتر قابل فهم باشد از one-hot-encoding استفاده می شود.در این روش، هر کلمه با یک بردار با طول تعداد لغات نشان‌داده می‌شود. در این بردار ها تمام عناصر صفر هستند، به غیر از آن عنصری که به آن کلمه اختصاص داده‌شده‌است.

تعبیه کلمات : در پردازش زبان طبیعی به مجموعه ای از تکنیک های یادگیری ویژگی و مدل سازی زبان تعبیه کلمات [2] می‌گویند. در این تکنیک ها کلمات و عبارات را به بردارهای عددی تبدیل می‌کند تا مدل یادگیری ماشین قادر به فهم و پردازش زبان طبیعی باشد. هر مجموعه ای از اعداد یک بردار کلمه است که به تنهایی سودمند نیست، مجموعه ای از این بردار کلمات سودمند است که معنای کلمات، ارتباط بین آن ها و محتوای کلمات مختلف را همان طور که به صورت طبیعی استفاده می‌شود بدست اورده باشند [۲۶].

با توجه به این‌که در این پایان نامه روش های شناسایی اخبار جعلی به کمک یادگیری عمیق مورد مطالعه قرار گفته است، در ادامه برخی تعریفات و ملزومات در این زمینه ارائه می گردد.

۱-۵ یادگیری عمیق

یادگیری عمیق، زیرمجموعه ای از یادگیری ماشین [3] است که از شبکه های عصبی چند لایه برای انتقال داده ها از ورودی به خروجی و ارائه دقت بالاتر در کارهایی مانند تشخیص گفتار ،چهره و .. استفاده می کند. یادگیری عمیق با روش‌های یادگیری ماشین سنتی متفاوت است، آن‌ها می‌توانند به طور خودکار ویژگی‌هایی چون متن، ویدئو و یا تصاویر را بدون دخالت انسان و یا به طور ضمنی، یاد بگیرند. معماری‌های شبکه عصبی عمیق مستقیما از داده‌های خام یاد می‌گیرند و می‌توانند دقت پیش‌بینی خود را در زمانی که اطلاعات بیشتر ارائ ه می‌شود، بهبود بخشند.

باتوجه به این که یادگیری عمیق مبتنی بر شبکه عصبی است، در ادامه ابتدا، مروری بر ساختار شبکه عصبی خواهیم داشت، سپس، دو نمونه از شبکه های عصبی که در یادگیری عمیق استفاده می شود معرفی می‌شوند.

۱-۵-۱ انواع شبکه عصبی

- شبکه عصبی مصنوعی [4]
- شبکه عصبی پیچشی [5]

[1] tokenization [2] Word embedding [3] Machine Learning [4] Artificial Neural Networks
[5] convolutional neural network

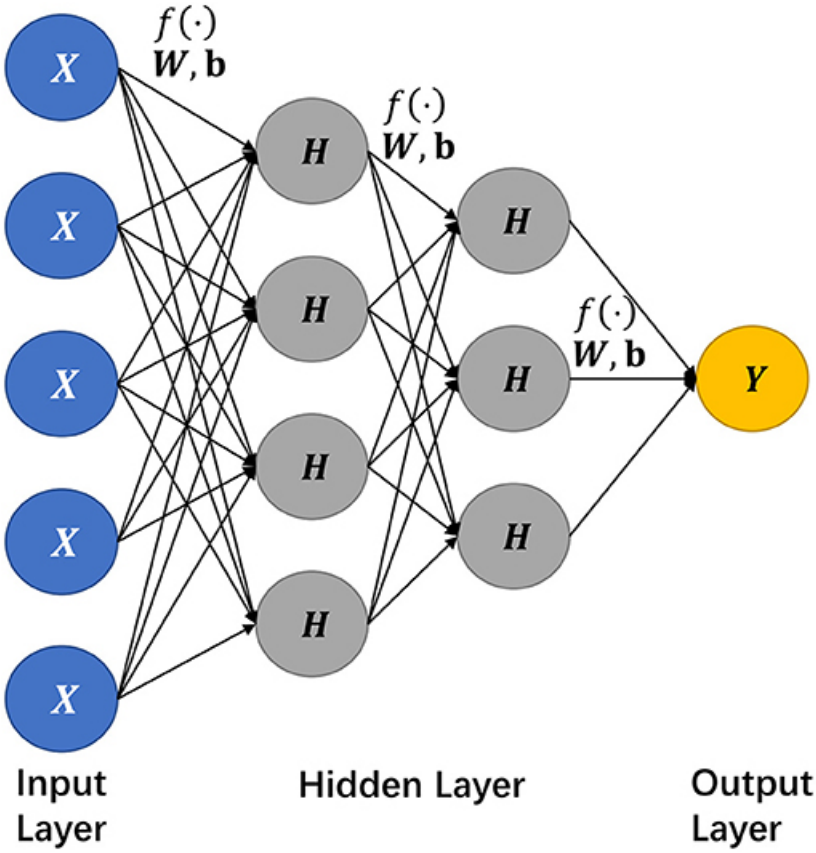
- شبکه عصبی بازگشتی [1]

۱-۵-۲ ساختار شبکه عصبی مصنوعی ANN

شبکه های عصبی مانند مغز انسان از تعداد زیادی نورون تشکیل شده که به یکدیگر متصل هستند، درواقع نورون، کوچک ترین واحد پردازش اطلاعات است، هر نورون به بخش‌های ورودی و تابع فعالیت تقسیم می‌شود. شبکه‌عصبی شامل یک لایه‌ی ورودی و یک لایه‌ی خروجی و چندین لایه‌ی میانی است که در هرلایه تعدادی نورون وجود دارد. ساختار آن به‌گونه ای است که تمام نورون‌های یک لایه به نورون‌های لایه‌ی دیگر متصل است. لایه‌ی ورودی شامل یک یا چند متغیر ویژگی (یا متغیرهای ورودی یا متغیرهای مستقل) است که به‌صورت $X1, X2, X3, \ldots$ نشان داده می‌شود، لایه‌ی میانی (پنهان) شامل یک یاچند گره مخفی است و لایه‌ی خروجی نیز شامل یک یا چند گره خروجی است.در شبکه عصبی اگر انتقال اطلاعات از سمت ورودی شروع و بعد لایه‌های پنهان و در نهایت به لایه‌ی خروجی برسد به آن شبکه عصبی پیشخور یا ایستا [2] می‌گویند، در مقابل اگر انتقال اطلاعات علاوه بر مسیر اصلی از لایه‌ی خروجی یا لایه‌ی پنهان به خود لایه یا لایه‌ی ورودی باشد به آن شبکه عصبی پسخور یا بازگشتی می‌گویند [۲۷]. تمامی نورون های ورودی شامل وزن های اولیه است، وزن در شبکه‌های عصبی در واقع پارامتری است که داده های ورودی را در لایه‌های پنهان شبکه تغییر می‌دهد، همان‌طور که در شکل (۱-۱) نشان داده شده‌است، شبکه عصبی مجموعه‌ای از نورون‌ها و گره‌ها است که درون هر گره مجموعه‌ای از متغیرهای ورودی و وزن‌ها w و مقدار بایاس b وجود دارد. شبکه‌عصبی همواره در پی یافتن بهترین وزن w است. نورون‌های ورودی شبکه که هرکدام وزن اولیه دارند وارد لایه‌ی اول شده و در تابع فعالیت موجود در آن قرارگرفته می‌شود، در اینجا جریان اطلاعات از ورودی به خروجی است بنابراین خروجی این لایه ورودی لایه بعدی خواهد بود و همین روند ادامه دارد و هر لایه تابع فعالیت مخصوص خود را دارد [۲۸، ۲۹].

---

[1] Recurrent Neural Networks [2] Feed Forward



شکل ۱-۱: شمای کلی یک شبکه عصبی [۱]

۱-۵-۲-۱ تابع هزینه

تابع ضرر که با نام تابع هزینه هم شناخته می‌شود، در واقع میزان خطا را در هربار اجرای شبکه عصبی برای مجموعه داده آموزشی نمایش می‌دهد. این تابع روشی برای ارزیابی کارکرد مدل است و تفاوت بین مقدار جواب واقعی و مقدار جواب پیش بینی شده مدل شبکه عصبی را محاسبه می‌کند، اگر جواب پیش بینی شده از جواب واقعی خیلی فاصله داشت یعنی مقدار تابع هزینه زیاد است و با کمک روش هایی مانند به‌روز رسانی کردن وزن‌ها و ..می‌توان این مقدار خطا را کاهش داد. هرچقدر جواب پیش بینی شده توسط شبکه به جواب واقعی ما نزدیک تر شود یعنی مقدار تابع هزینه کاهش یافته است.
چند تابع هزینه مختلف وجود دارد، که در ادامه به آن‌ها اشاره می‌شود :

- میانگین قدرمطلق خطا (mean absolute error)

- میانگین مربعات خطا (mean squared error)

- میانگین خطای ساده (mean bias error)

- اختلاف باینری آنتروپی (binary cross entropy)

- اختلاف دسته‌ای آنتروپی (categorical cross entropy)

۱-۵-۲-۲ تابع بهینه ساز

در شبکه‌های عصبی، هدف کمینه کردن تابع هزینه است، تابع بهینه‌ساز کمک می کند تا با بهبود وزن‌ها هزینه مدل کاهش یابد.از توابع بهینه‌ساز معروف می‌توان به روش سریع‌ترین شیب، سریع‌ترین شیب تصادفی، و هم چنین روش‌های توسعه داده شده این روش مانند RMSprop، Adam، Adagrad اشاره کرد. اکثر این توابع، به‌خوبی کار می‌کنند و نمی‌توان برتری یکی از آن‌ها را نسبت به دیگری بیان کرد، این توابع کارایی متفاوتی در انواع مختلف شبکه عصبی دارد [۲۸].

۱-۵-۲-۳ تابع فعالیت

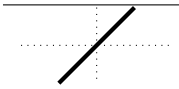
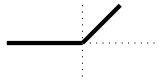
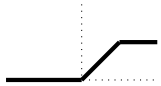
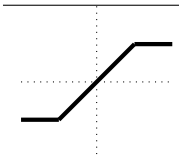
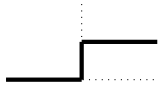
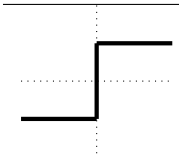
در شبکه‌های عصبی تابع فعالیت [1] از اهمیت زیادی برخوردار است و معمولا به‌صورت توابع ریاضی غیرخطی هستند، به زبان ساده‌تر می‌توان گفت که تابع فعالیت درباره‌ی فعال بودن یک نورون در شبکه عصبی تصمیم می‌گیرد. وجود این توابع برای یادگیری [2] و درک چیزهای واقعا پیچیده درون یک شبکه عصبی مصنوعی [3] بسیار حائز اهمیت است. هر لایه‌ی شبکه عصبی به صورت مجزا یک تابع فعالیت دارد، هر تابع فعالیت در شبکه های عصبی یک عدد را به عنوان ورودی دریافت می‌کند و سپس یک سری عملیات ریاضی بر روی این اعداد اعمال می‌شود ،خروجی این تابع عددی است که در بازه ی ۰ تا ۱ یا در بازه ی ۱ تا ۱- قرار می‌گیرد. در شکل زیر یک نورون با تابع فعالیت f مشاهده می‌شود. همان‌طور که می‌بینید در این جا مجموع حاصل ضرب ورودی ها که شامل نورون x و وزن های w است را با مقدار بایاس b جمع می‌شود و نتیجه به صورت فرمول (۱-۳) محاسبه می‌شود [۲۷، ۲۹].

$$u = \sum_{i=۱}^{n} x_i w_i + b \qquad (۳-۱)$$

تابع فعالیت با استفاده از مرز تصمیم‌گیری داده‌های مختلف را از یکدیگر جدا می‌کند، مرز تصمیم گیری در ابعاد مختلف متفاوت است برای مثال در یک فضای دو بعدی یک خط و درفضای سه بعدی یک صفحه و در ابعاد بالاتر یک ابر صفحه خواهد بود [۳۰، ۲۷].

[1] Activation Function [2] Learning [3] Artificia Neural Network

جدول ۱-۱: انواع توابع فعالیت

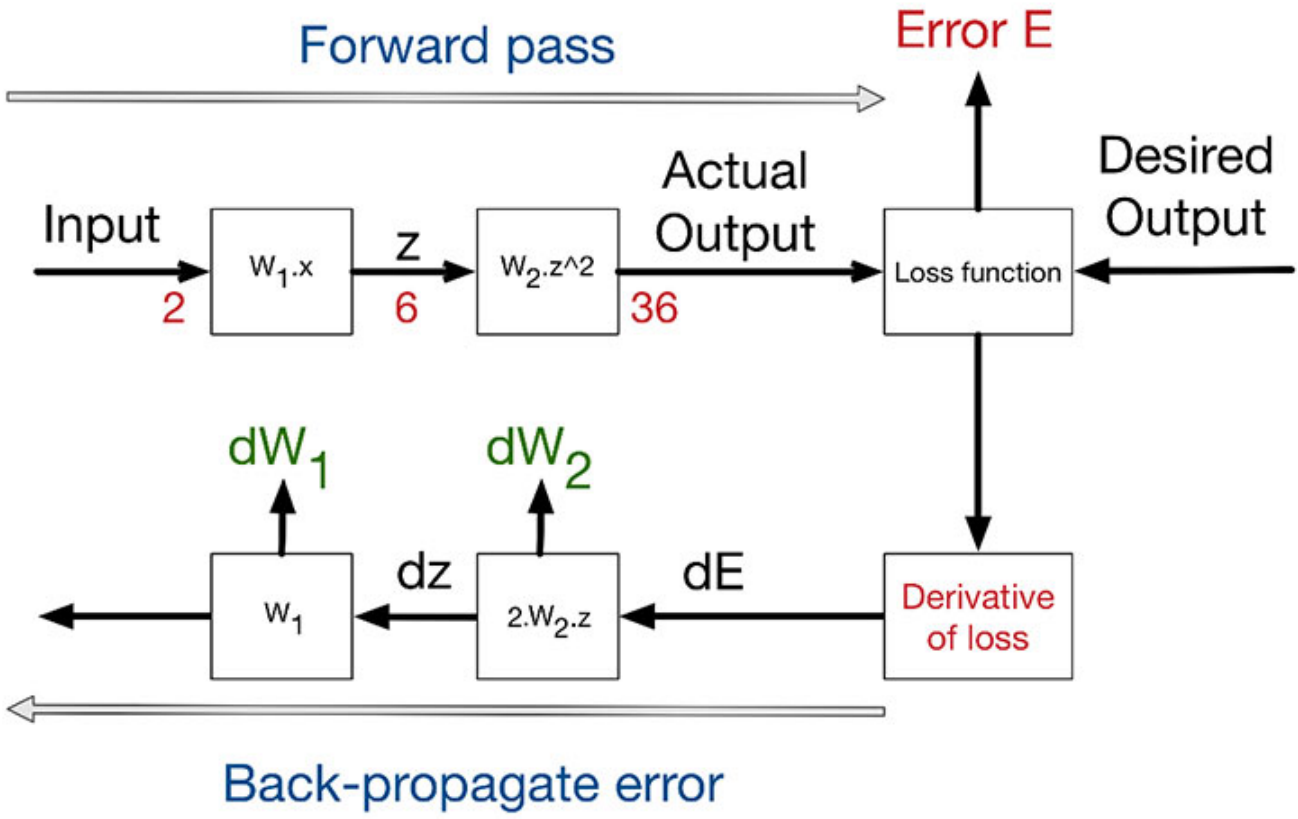
ردیف	نام	تعریف تابع	نمودار
۱	همانی (identity)	$f(x) = x$	
۲	یک‌سو ساز (Relu)	$f(x) = \begin{cases} ۰ & \text{if } x < ۰ \\ x & \text{if } x \geq ۰ \end{cases}$	
۳	آستانه‌ای خطی	$f(x) = \begin{cases} ۰ & \text{if } x < ۰ \\ x & \text{if } ۰ \leq x \leq ۱ \\ ۱ & \text{if } x > ۱ \end{cases}$	
۴	آستانه‌ای خطی متقارن	$f(x) = \begin{cases} -۱ & \text{if } x < -۱ \\ x & \text{if } -۱ \leq x \leq ۱ \\ ۱ & \text{if } x > ۱ \end{cases}$	
۵	آستانه‌ای دو مقداره	$f(x) = \begin{cases} ۰ & \text{if } x < ۰ \\ ۱ & \text{if } x \geq ۰ \end{cases}$	
۶	آستانه‌ای دو مقداره متقارن	$f(x) = \begin{cases} -۱ & \text{if } x < ۰ \\ ۱ & \text{if } x \geq ۰ \end{cases}$	

۱-۵-۲-۴ پس انتشار خطا

میزان خطا در طی آموزش یک شبکه عصبی با مقایسه ی میزان خروجی مدل و میزان واقعی محاسبه می شود. همان طور که در بخش های قبل مشاهده شد در شبکه عصبی، یادگیری از طریق به‌روز رسانی وزن ها w ومقدار بایاس b است. یادگیری در شبکه‌های عصبی با تکرار انجام می‌شود به این معنی که داده‌های مجموعه داده مان راچندین بار به الگوریتم می‌دهد و این الگوریتم با به روز رسانی وزن ها و مقدار بایاس می‌تواند میزان خطا را کمینه کند. در روش پس انتشار خطا [1] در هر تکرار دو مرحله وجود دارد، مرحله ی اول همان حرکت رو به جلو [2] است که با ضرب داده های ورودی در وزن و سپس جمع با مقدار بایاس یک مقدار خروجی به دست می آورد که با خروجی واقعی متفاوت است که این مقدار بیان‌گر میزان خطا است، حالا

[1] Back propagation of error [2] (feed forward)

با در نظر گرفتن میزان خطا، وارد مرحله ی دوم تکرار شده، در این مرحله می توان به عقب بازگشته و وزن ها و مقدار بایاس را به روز رسانی کرد تا در تکرار بعدی نتیجه ی بهتری به دست آورد، این تکرار آنقدر انجام می‌شود تا میزان خطا کمینه شود و خروجی شبکه به نزدیک ترین مقدار واقعی خود برسد شکل (۱-۲) بیان‌گر مراحل پس انتشار خطا است [۲۸، ۳۱].



شکل ۱-۲: پس‌انتشارخطا در شبکه‌عصبی[۳]

رابطه ای که در زیر مشاهده می‌کنید برای شبکه عصبی با یک خروجی است اما این الگوریتم به دلیل استفاده مداوم از قاعده مشتق زنجیره ای و قانون توان قابلیت این را دارد که روی یک شبکه با هر تعداد خروجی پیاده سازی شود.ابتدا فرضیات زیر را در نظر بگیرید :

- w_{ij}^k: وزن گره j در لایه ی l_k برای گروه وارد شونده i
- o_i^k: خروجی گره i در لایه l_k
- b_i^k: بایاس گره i در لایه l_k

- r_k: تعداد گره های لایه l_k
- a_i^k: مجموع خاصلضرب به علاوه بایاس (فعالیت) برای گره i در لایه l_k
- g: تابع فعالیت گره های لایه پنهان
- g_o: تابع فعالیت گره های لایه خروجی

مقدار ورودی و خروجی به صورت (x_i, y_i) که در آن y_i مقدار خروجی واقعی و $\overline{y_i}$ خروجی مدل است. تا بع خطا به صورت رابطه (۱-۴) محاسبه می‌شود :

$$E(X,T) = \frac{۱}{۲N}\sum_{i=۱}^{N}(\overline{y_i} - y_i)^۲ \qquad (۱-۴)$$

در نتیجه تابع هزینه E برحسب مقادیر خروجی y از رابطه (۱-۵) محاسبه می‌شود.

$$y_j = f(a_j) \qquad (۱-۵)$$

هم چنین مقدار y برحسب a محاسبه شده و همچنین مقدار a برحسب w_{ij}

$$a_j = \sum_{i=۱}^{N} y_i w_{ij} \qquad (۱-۶)$$

مشتق،نسبت به وزنw و قاعده زنجیره ای محاسبه می‌شود که به فرم رابطه (۱-۷) زیر است:

$$\frac{\partial L(z, y_۱)}{\partial \omega} = \frac{\partial (z, y_۱)}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial \omega} \qquad (۱-۷)$$

و در نتیجه وزن ها به صورت رابطه (۱-۸) به روزرسانی می‌شود:

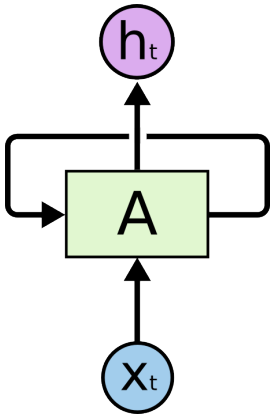
$$w \leftarrow\ w - \alpha\frac{\partial L(z, y_۱)}{\partial \omega} \qquad (۱-۸)$$

۱-۵-۳ شبکه عصبی بازگشتی

شبکه‌های عصبی بازگشتی برای اولین بار در سال ۱۹۸۰ به‌وجود آمدند اما اخیرا مورد استفاده قرار گرفته و گسترش یافته‌است. بخش عظیمی از داده‌ها مانند داده‌های سری زمانی [1] ،گفتار،متن و...به صورت سریال (سری [2]) است.
شبکه‌های عصبی بازگشتی یک خانواده از شبکه‌های عصبی است که برای پردازش داده های سری یا دنباله ها استفاده می‌شود.

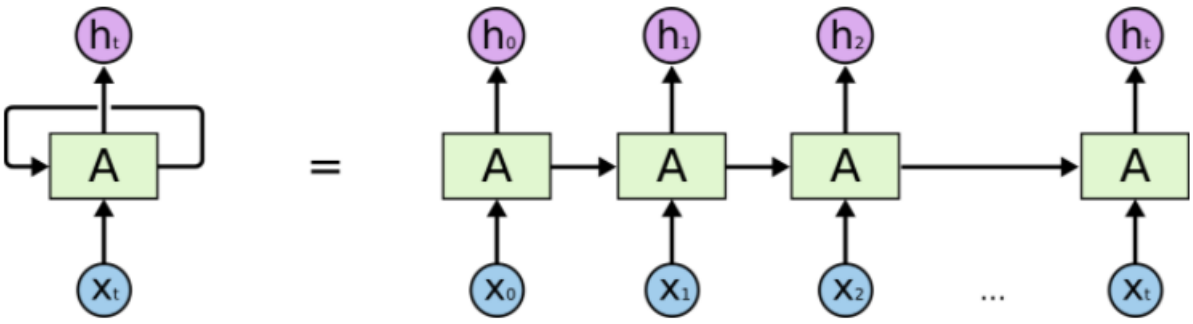
[1] time series [2] sequential

همان‌طور که گفته شد در شبکه‌های عصبی قدیمی فرض بر این است که تمام مقادیر ورودی از یک دیگر مستقل هستند اما این فرضیه همیشه صادق نیست، به‌عبارت دیگر اگر در یک جمله کلمه بعدی را بخواهد پیش بینی کند باید کلمات قبلی را بداند، شبکه‌های عصبی بازگشتی برای برطرف کردن این مشکل طراحی شده اند. درحقیقت شبکه‌های عصبی بازگشتی همان طور که از اسم آن مشخص است تکرار شونده است بنابراین داخل خودیک حلقه بازگشتی دارند که منجر به حفظ و نگهداری اطلاعات قبلی در شبکه می‌شود. در شکل زیر x_t به عنوان مقدار ورودی دریافت وبه بخش A منتقل می‌شود و مقدار h_t را به خروجی می‌برد. این حلقه ها باعث می‌شود که اطلاعات در هر مرحله به شبکه بعدی منتقل شود.



شکل ۱-۳: شبکه عصبی بازگشتی دارای حلقه[۴]

شبکه های عصبی بازگشتی را می‌توان به صورت چند کپی از یک شبکه عصبی در نظر گرفت. با توجه به شکل زنجیره مانند شبکه های عصبی می‌توان فهمید که این شبکه ها به دنباله ها و لیست ها مرتبط است و برای کار با چنین داده هایی انتخاب شده اند. در شکل (۱-۴) حالت زنجیره وار شبکه های عصبی بازگشتی را مشاهده می‌کنید [۳۲، ۳۳، ۳۴].



شکل ۱-۴: شبکه عصبی بازگشتی باز شده[۴]

نحوه اجرای شبکه عصبی به این صورت است که ابتدا تمام توابع فعالیت مستقل را به وابسته تبدیل می‌کند، سپس برای کاهش پیچیدگی پارامتر های RNN برای لایه ها وزن و بایاس یکسانی در نظر می‌گیرد. درهرمرحله خروجی لایه ی قبلی به

عنوان ورودی لایه ی بعد در نظر گرفته می‌شود. رابطه (۱-۹) را در نظر بگیرید :

$$h_t = f(h_{t-1}, x_t) \tag{۱-۹}$$

- h_t: حالت فعلی
- h_{t-1}: حالت قبلی
- x_t: حالت ورودی

بعد از اعمال تابع فعالیت tanh رابطه به صورت (۱-۱۰) است :

$$h_t = tanh(w_{hh}h_{t-1} + w_{xh}x_t) \tag{۱-۱۰}$$

که در آن :

- w_{xh}: وزن نورون ورودی
- w_{hh}: وزن نورون در حالت مخفی

برای محاسبه مقدار خروجی از رابطه ی (۱-۱۱) استفاده می‌شود :

$$y_t = w_{hy}h_t \tag{۱-۱۱}$$

نحوه آموزش از طریق شبکه عصبی بازگشتی :

۱. این شبکه یک گام زمانی از ورودی دریافت می‌کند.

۲. حالت فعلی از ورودی فعلی و حالت قبلی محاسبه می‌شود.

۳. خروجی حالت قبل تبدیل به ورودی حالت فعلی می‌شود.

۴. برای n گام مختلف این روند تکرار می‌شود وبه این صورت کل شبکه به هم متصل می‌شود.

۵. بعد از تکمیل مراحل، مرحله آخر محاسبه‌ی مقدار خروجی است.

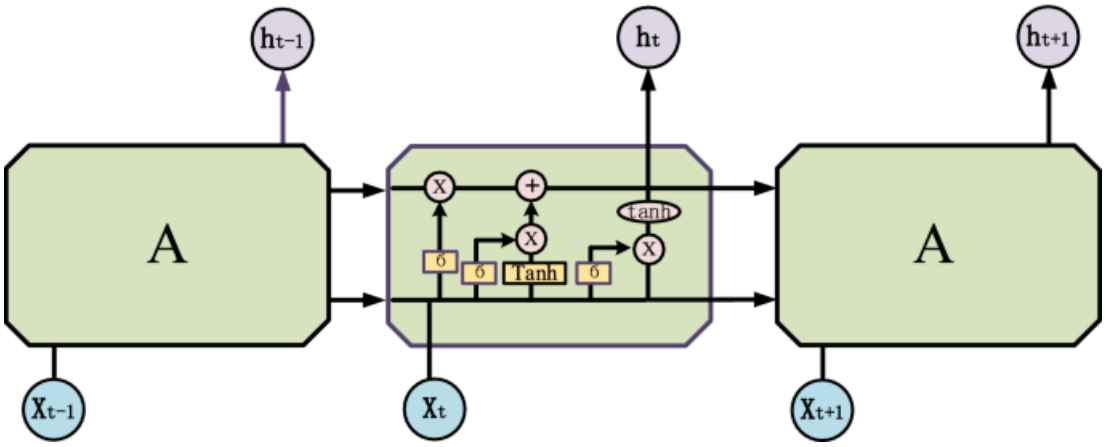
۶. میزان اختلاف بین خروجی واقعی و خروجی مدل خطای شبکه را نشان می‌دهد.

۷. اگر میزان خطا زیاد بود خطا به شبکه برگشت داده می‌شود تا با تنظیم وزن ها میزان خطا کاهش یابد.

از کاربردهای این نوع شبکه می‌توان به تشخیص گفتار، برچسب زنی خودکار، و تحلیل احساسات اشاره کرد [۳۳، ۳۴].

LSTM ۱-۳-۵-۱

شبکه های LSTM خلاصه ی عبارت Long Short Term Memory است.این شبکه ها توانایی یادگیری وابستگی های بلند مدت را دارند و هدف از طراحی آن ها حل مشکل وابستگی بلندمدت است. ساختار شبکه های LSTM به‌این‌صورت است که اطلاعات دور را به‌خاطر می‌سپارند و به طور ذاتی این ویژگی را در خود دارند. این شبکه ها مانند شبکه های بازگشتی ساختار زنجیره مانند دارند ولی ماژول تکرار شونده آن ساختار متفاوتی دارد و به جای داشتن یک لایه ی شبکه عصبی دارای ۴ لایه هستند که طبق ساختار ویژه ای با یکدیگر در ارتباط هستند.



شکل ۱-۵: شبکه LSTM [۴]

معنی هر کدام از علامت‌هایی را که درشکل (۱-۵) استفاده می‌شود به‌صورت زیر است :

در این شبکه ها مفاهیم جدیدی وجود دارد که در شبکه‌های عصبی بازگشتی معمولی وجود نداشتند، برای مثال این شبکه ها سه دروازه [۱] دارند که شبکه از طریق آن جریان داده درون خود را کنترل می‌کند. این سه دروازه عبارتند از :

- دروازه فراموشی یا Forget gate
- دروازه ورودی یا Input gate
- دروازه خروجی یا Output gate

[1] gate

علاوه بر این سه دروازه یک عنصر اصلی به نام سلول حالت [1] دارند که به صورت یک خط افقی است که در بالای شکل قرار دارد و دارای ساختار ساده ای است که تغییرات زیادی در آن اتفاق نمی‌افتد و با C_t نمایش داده می‌شود. شبکه های lstm از طریق دروازه ها می‌توانند اطلاعات جدیدی به سلول حالت اضافه کنند یا اطلاعات را حذف کنند درواقع دروازه ها راهی برای ورود اختیاری اطلاعات هستند که از یک لایه ی شبکه عصبی sigmoid به همراه یک عملگر ضرب نقطه ای تشکیل شده اند. خروجی لایه sigmoid عددی بین صفر و یک است که بیان‌گر این است که چه مقدار ورودی باید به خروجی ارسال شود.مقدار صفر یعنی هیچ اطلاعاتی ارسال نشود و مقدار یک یعنی تمام اطلاعات ورودی به خروجی ارسال گردد [۳۵].

رابطه (۱-۱۲) بیان‌گر دروازه فراموشی است. در روابط زیر h_{t-1}, x_t مقادیر ورودی هستند. اندیس f در W_f مخفف forget می‌باشد. در این دروازه اطلاعات غیرضروری گذشته فراموش می‌شود.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (۱-۱۲)$$

رابطه (۱-۱۳) بیان‌گر دروازه ورودی است. اندیس i در W_i مخفف input می‌باشد.دروازه ورودی، بررسی می‌کند که اطلاعات به‌دست آمده از لحظه جاری (t) ارزش ذخیره در حافظه بلندمدت را دارند یا خیر؟

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (۱-۱۳)$$

یک لایه تانژانت هایپربولیک است که برداری از مقادیر، به نام $\tilde{C}_t$ می‌سازد که می‌توان آن‌ها را به سلول حالت اضافه کرد.

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \quad (۱-۱۴)$$

با ترکیب مرحله فراموشی و مرحله ورودی مقدار سلول حالت به‌روزرسانی می‌شود.

$$C_t = (f_t \bigodot C_{t-1} + i_t \bigodot \tilde{C}_t) \quad (۱-۱۵)$$

رابطه (۱-۱۶) بیان‌گر دروازه خروجی است. اندیس o در W_o مخفف output می‌باشد.دروازه خروجی برای این بود که همه اطلاعات موجود در C_t به خروجی h_t منتقل نشود.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (۱-۱۶)$$

در رابطه (۱-۱۷) خروجی تولیدشده از دروازه خروجی o_t باید در خروجی تابع سیگموئید ضرب شود تا آنقدری که نیاز

[1] cell state

است به خروجی h_t منتقل شود.

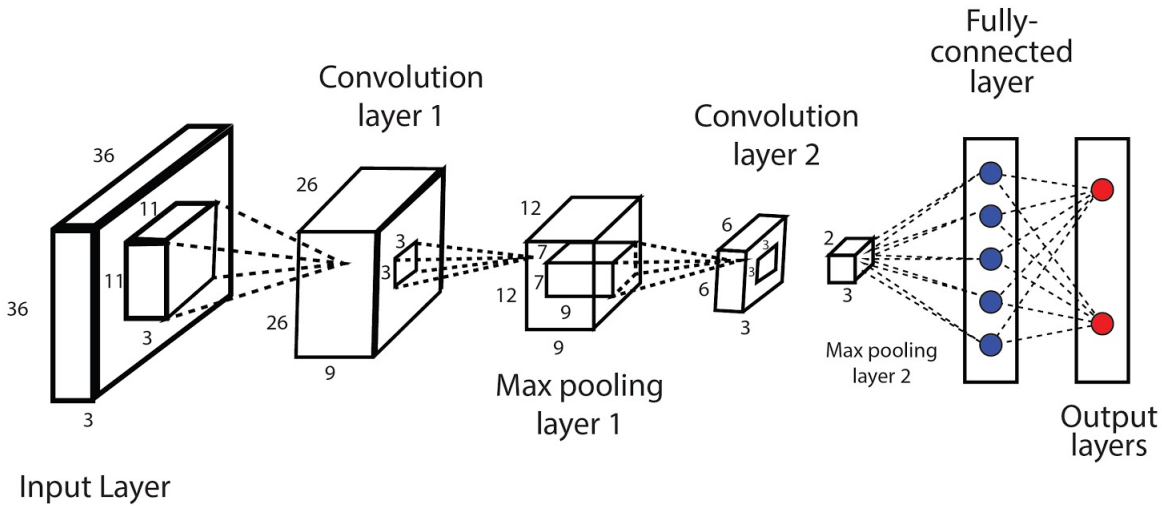
$$h_t = o_t \bigodot tanh(C_t) \tag{۱-۱۷}$$

۱-۵-۴ شبکه های پیچشی

این عملیات پیچیدگی را در لایه های خاصی انجام می‌دهد از این رو، شبکه عصبی پیچشی [۱] نام دارد. معماری از حالت شبکه عصبی سنتی تغییر می‌کند، ورودی این نوع شبکه ها تصویر است، از لایه‌های نورونی با وزن و بایاس با قابلیت تنظیم تشکیل شده است،همچنین این نورون ها دارای سه بعد (عرض، ارتفاع، عمق) هستند. هر نورون تعدادی ورودی دریافت کرده و سپس حاصل ضرب وزن های نورون در ورودی ها را محاسبه کرده و با مقدار بایاس جمع می‌شود و در انتها با استفاده از یک تابع فعالیت خروجی را نمایش می‌دهد.معماری این نوع شبکه از پنج لایه تشکیل شده است: لایه ورودی، لایه پیچشی،لایه غیرخطی، لایه ادغام [۲] ، لایه تماما متصل [۳] که به طور معمول تابع غیر خطی، همراه با لایه‌پیچشی یک‌جا نشان داده می‌شود، همچنین، این نوع شبکه ها یک تابع هزینه [۴] مانند (SVM،Softmax) دارند و از لایه آخر که همان لایه تماما متصل است برای طبقه بندی برچسب کلاس استفاده می‌شود. همانند شکل (۱-۶) شبکه پیچشی از سه لایه اصلی که شامل لایه پیچشی، لایه ادغام و لایه تماما متصل است تشکیل شده است، در ادامه توضیح مختصری از سه لایه اصلی ارائه می‌شود.

---

[1]Convolution [2]pooling [3]fully connected [4]Loss function

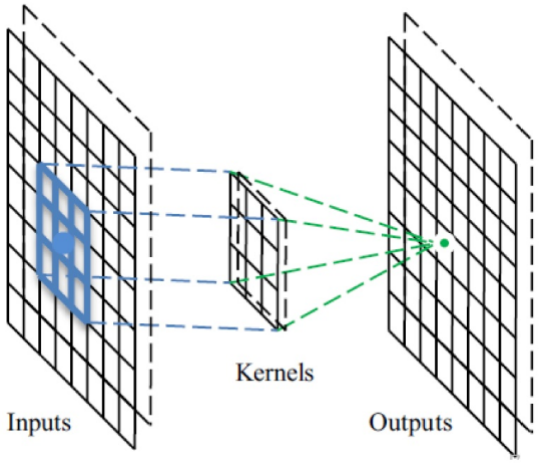


شکل ۱-۶: معماری شبکه عصبی پیچشی[۵]

- **لایه پیچشی :**

هسته اصلی یک شبکه CNN لایه پیچشی [۱] است که حجم عظیمی از محاسبات شبکه عصبی در این لایه انجام می‌شود.هر لایه پیچشی در شبکه عصبی پیچشی شامل مجموعه‌ای از فیلترها است و از پیچش بین فیلترها و لایه ورودی، خروجی شبکه ساخته می‌شود. به خروجی لایه پیچشی، نگاشت ویژگی [۲] گفته می‌شود.شکل (۱-۷) نشان دهنده لایه پیچشی است.

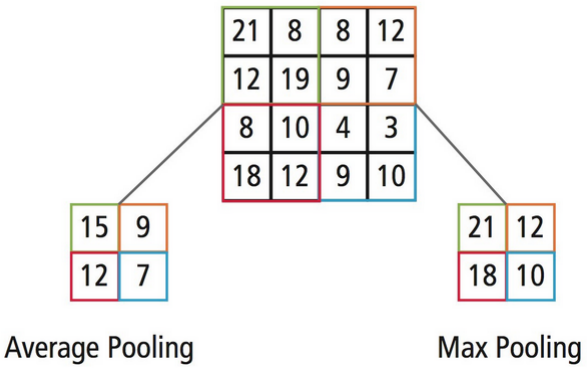
[1]Convolutional Layer [2]Feature Map



شکل ۱-۷: لایه پیچشی[۶]

- **لایه ادغام:**

این لایه عملکردی شبیه لایه پیچشی دارد و یک پنجره روی تصویر حرکت می‌کند، هدف لایه ادغام [۱] کاهش بعد مقدار خروجی لایه پیچشی است. لایه ادغام پارامتر قابل آموزش ندارد و شامل دو نمونه max pooling و average pooling است. در max pooling در هر پنجره بیشترین مقدار را انتخاب می‌کند و در average pooling میانگین را حساب می‌کند. در شکل (۱-۸) عملکرد لایه ادغام مشاهده می‌شود.



شکل ۱-۸: لایه ادغام[۷]

- **لایه تماما متصل:**

آخرین بخش یک شبکه عصبی پیچشی برای طبقه بندی را،لایه های تماما متصل [۲] تشکیل می‌دهند.یکی از کاربردهای اصلی این لایه در شبکه پیچشی، استفاده به عنوان طبقه بند است. یعنی مجموعه ویژگی‌هایی که از لایه پیچشی استخراج می‌شوند،

[1] pooling [2] fully connected

درنهایت تبدیل به یک بردار می‌شوند و این بردار ویژگی به یک طبقه بند تماما متصل داده می‌شود تا نوع برچسب را مشخص کند و برچسب درست را شناسایی کند [۳۶]. در شبکه هایی که از لایه تماما متصل برای کلاس بندی استفاده می‌کنند خروجی نگاشت ویژگی بعد از الحاق به یکدیگر، به تابع بیشینه فعال داده می‌شود. در شکل (۱-۶) لایه تماما متصل مشاهده می شود.

۱-۶ معیار ارزیابی

برای ارزیابی عملکرد الگوریتم های مربوط به مسائل جعل خبر، معیارهای مختلف ارزیابی استفاده شده‌است .در این قسمت معیارهایی که اغلب برای تشخیص خبر جعلی استفاده می‌شود مورد بررسی قرار می‌گیرد :

مثبت صادق TP : هنگامی که اخبار جعلی پیش بینی شده به درستی به عنوان اخبار جعلی مطرح می‌شوند.

منفی صادق TN : زمانی که اخبار واقعی پیش بینی شده در واقع به عنوان خبر واقعی مطرح می‌شوند.

منفی کاذب FN : هنگامی که مقادیر واقعی اخبار پیش بینی شده به عنوان اخبار جعلی مطرح می‌شوند.

مثبت کاذب FP : زمانی که اخبار جعلی پیش بینی شده، به عنوان خبر واقعی مطرح می‌شوند.

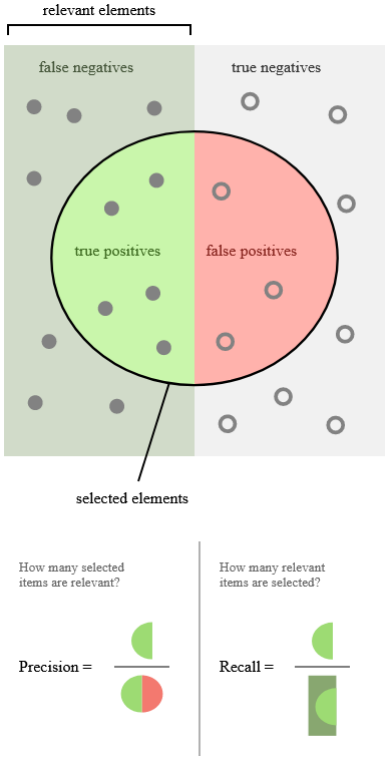
Precision : چند درصد از داده‌هایی که مدل طبقه بندی کرده درست بودند.

$$precision = \sum_{i=1}^{|C|} TP_j / \sum_{i=1}^{|C|} (TP_j + FP_j) \qquad (۱۸-۱)$$

Recall : چند درصد از داده‌های مربوطه به درستی طبقه‌بندی شده است.

$$recall = \sum_{i=1}^{|C|} TP_j / \sum_{i=1}^{|C|} (TP_j + FN_j) \qquad (۱۹-۱)$$

در شکل (۱-۹) معیارهای ارزیابی را مشاهده می‌کنید.



شکل ۱-۹: Precision ، Recall [۸]

Accuracy : دقت طبقه بندی را نشان می دهد، یعنی نسبت پیش بینی‌های درست به تعداد کل نمونه‌ها و به‌صورت رابطه (۱-۲۰) محاسبه می‌شود :

$$Accuracy = \sum_{i=1}^{|C|}(TP_j + TN_j) / \sum_{i=1}^{|C|}(TP_j + TNj + FP_j + FN_j) \qquad (۱-۲۰)$$

F score : میانگین Precision و Recall را نشان می دهد، این معیار در بهترین حالت ۱ و در بدترین حالت ۰ است و به صورت رابطه ۱-۲۱ محاسبه می‌شود [۱۴].

$$Fscore = ۲ * (Precision * Recall / Precision + Recall) \qquad (۱-۲۱)$$

۱-۷ مجموعه داده

برای تشخیص اخبار جعلی، از یک مجموعه داده [۱] استفاده می‌شود که در این بخش به تعریف و بررسی آن پرداخته می‌شود. اخبار به‌روز را می‌توان از منابع مختلف، از جمله صفحه‌های آژانس خبری، موتورهای جستجو و وب سایتهای اجتماعی، جمع آوری کرد .با این‌حال، تعیین صحت اخبار به صورت دستی یک کار چالش برانگیز است و نیازمند افراد متخصص در این حوزه است.

برخی از مجموعه داده های قابل دسترس در زیر لیست شده اند:

خبر Buzz Feed [۲]:

این مجموعه داده شامل نمونه ای کامل از اخبار منتشر شده در فیس بوک از۹ اژانس خبری در طی یک هفته نزدیک به انتخابات ۲۰۱۶ ایالات متحده ۱۹ تا۲۳ سپتامبر و از۲۶ تا۲۷ سپتامبر است. هر پست و مقاله مرتبط،توسط۵ خبرنگار BuzzFeed ادعا شده است [۳۷].

آشکار ساز BS [۳]:

این مجموعه داده ها از یک مرورگر به نام « آشکارساز BS » برای بررسی صحت خبر تهیه شده است .

CREDBANK[۴]:

این یک مجموعه داده ی گسترده ای است که تقریبا ۶۰میلیون توییت را شامل می‌شود. تمام توییت ها به بیش از۱۰۰۰ رویداد خبری مرتبط هستند [۱۴].

Twitter[۵]:

مجموعه داده های توییتر از یک معیار چند رسانه ای استفاده می‌کندکه برای کشف اخبار جعلی در توییتر کاربرد دارد.این مجموعه دارای دو بخش است : ۱)مجموعه آزمایش ۲) مجموعه آموزش. از داده‌های train به عنوان مجموعه آموزشی و از داده هایtest به عنوان مجموعه آزمایشی استفاده می‌شود. توییت های توییتر شامل متن، تصویر ویدئو و اطلاعات متنی دیگر است [۱۱].

Weibo[۶]:

این مجموعه داده‌ها، اخبار واقعی هستند که از منابع خبری موثق چین از جمله خبرگزاری شینهوا جمع‌آوری می‌شوند. این اخبار از ماه May 2012 تا January 2016 جمع‌آوری شده و توسط سیستم رسمی weibo تایید می‌شود. این سیستم کاربران مشترک را برای گزارش پست های مشکوک تشویق می‌کند و پست های مشکوک توسط تعدادی از کاربران مورد

[1]Data set [2]https://github.com/BuzzFeedNews/2016-10-facebook-fact-check/tree/master/data [3]https://github.com/bs-detector/bs-detector [4]http://compsocial.github.io/CREDBANK-data/ [5]https://github.com/MKLab-ITI/image-verification-corpus [6]https://github.com/yaqingwang/EANN-KDD18

اعتماد بررسی می‌شود. این سیستم به عنوان منبع معتبر برای جمع آوری اخبار شایعه عمل می‌کند، زمانی که این داده ها پیش‌پردازش می‌شود می توان ازآن ها در کار استفاده کرد. [۱۱].

۱-۸ استخراج ویژگی :

استخراج ویژگی‌های دست‌ساز برای مدل‌های تشخیص اخبارجعلی مبتنی بر یادگیری ماشین سنتی دشوار است. برای غلبه بر این محدودیت، در این‌جا به بررسی مدل یادگیری‌عمیق که برای شناسایی اخبار جعلی پیشنهاد شده‌است پرداخته می‌شود که استخراج ویژگی یه طور خودکار انجام می‌شود.استخراج ویژگی در مدل یادگیری‌عمیق مبتنی بر دو حالت است:استخراج ویژگی داده‌های تک حالت و استخراج ویژگی داده‌های چند حالت که در زیر به بررسی آن پرداخته شده‌است:

۱-۸-۱ استخراج ویژگی مبتنی بر داده های تک حالته

در این حالت تنها ویژگی‌های متنی در نظر گرفته می‌شود .ویژگی های متنی ویژگی هایی است که از محتوای متن پست ها استخراج می‌شود و در بسیاری از روش های کشف اخبار جعلی، مورد بررسی قرار گرفته است. برای مثال، می‌توان برای هر کلمه براساس میزان اهمیت کلمه در جمله وزن خاصی در نظر گرفت و کلمات با اهمیت‌تر را برای استخراج ویژگی انتخاب کرد. استخراج ویژگی های متنی از چند جزء اصلی تشکیل شده است :

منبع : این خبر از کجا می آید؟ چه کسی آن را نوشته است.

تیتر : برای جذب خوانندگان اخبار به طور خلاصه شرح داده می‌شود.

بدنه متن : محتوای اصلی مقاله خبری.

۱-۸-۲ استخراج ویژگی مبتنی بر داده های چند حالته

با پیشرفت فناوری چند رسانه ای اخبار از پست های متنی به پست های چند رسانه ای تصویر و ویدئو تبدیل می‌شوند بنابراین در این حالت علاوه بر ویژگی های متن ویژگی های بصری از جمله تصاویر و ویدئوها نیز در نظر گرفته می‌شود.ویژگی های بصری یک شاخص مهم برای کشف اخبار جعلی است. این کار باعث می شود نوجه خوانندگان جلب شود و از قدرت تصور بالاتری برخوردار شوند، البته این یک مزیت برای اخبار جعلی محسوب می‌شود، زیرا با دستکاری تصاویر باعث گمراه کردن خوانندگان و انتشار سریع خبر می‌شود. تصاویر جعلی به دو گروه طبقه بندی می‌شوند : یکی تصاویر دستکاری شده و دیگری تصاویر گمراه کننده. تصاویر دستکاری شده تصاویری است که به صورت دیجیتالی اصلاح شده اند برای مثال با نرم افزارهایی نظیر فوتوشاپ و.. تغییراتی در تصاویر به وجود می‌آورند.تصاویر گمراه کننده تصاویری است که دستکاری

نمی‌شوند اما مطالب گمراه کننده ای دارد مانند عدم تطابق متن و تصویر. همچنین می‌توان از بی کیفیت بودن تصاویر به عنوان علتی برای جعلی بودن خبر نام برد. چالش اصلی این است که چطور می‌توان این اخبار را از ویژگی ها شناسایی کرد؟ ویژگی ها را می‌توان از توئیت ها، تصاویر و ...استخراج کرد. [۱۱، ۹، ۳۸].

برای مثال شکل (۱-۱۰) یک تصویر دستکاری شده است که در آن پوتین را روی صندلی وسط اضافه کرده‌اند تا نشان دهند او نیز در میان یک بحث شدید در میان دیگر رهبران جهان بوده است.

شکل ۱-۱۰: تصاویر دستکاری شده[۹]

شکل ۱-۱۱: تصاویر گمراه کننده[۹]

در شکل (۱-۱۱) یک تصویر گمراه‌کننده است که عدم تطابق متن و تصویر را نشان می‌دهد، این تصویر به عنوان تصویر خورشید گرفتگی در ۲۰ مارس ۲۰۱۵ ارائه شده است در صورتی که در اصل تصویر یک اثر هنری است که طراحی شده است برای همین تشخیص آن از واقعیت دشوار است.

۱-۹ روش انجام تحقیق :

هدف از این تحقیق تشخیص اخبارجعلی در انواع رسانه های اجتماعی با استفاده از تکنیک های یادگیری عمیق می‌باشد.در این تحقیق با در اختیار داشتن یک مجموعه داده و ارائه یک روش مناسب با استفاده از تکنیک های یادگیری عمیق سعی در تشخیص و بالابردن دقت و سرعت انجام آن خواهد شد. در ادامه با مطالعه کارهای قبل و روش های به کاررفته در آن ها و انجام مقایسه بین روش های پیشین و روش پیشنهادی نتیجه گیری کلی انجام خواهد شد و روش مورد نظر از لحاظ معیارهای ارزیابی با دیگر روش ها موردنقد و بررسی قرار می‌گیرد.

مقالات اصلی مورد استفاده در این پایان نامه به قرار زیر است :

1. Wang, Yaqing, Ma, Fenglong, Jin, Zhiwei, Yuan, Ye, Xun, Guangxu, Jha, Kishlay, Su, Lu, and Gao, Jing. Eann: Event adversarial neural networks for multi-modal fake news detection. in *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, pp. 849–857, 2018.

2. Karimi, Hamid and Tang, Jiliang. Learning hierarchical discourse-level structure for fake news detection. *arXiv preprint arXiv:1903.07389*, 2019.

فصل ۲

مروری بر کار های گذشته

در این فصل در ابتدا به معرفی مقدمات مربوط به اخبار جعلی پرداخته می شود و سپس پژوهش های انجام شده در زمینه ی تشخیص اخبار جعلی بررسی می‌شود. درچند سال گذشته بسیاری اززمینه ها، تکامل زیادی را تجربه کرده‌اند که از جمله آن‌ها می‌توان علوم یادگیری عمیق و یادگیری ماشین را نام برد. اخبارجعلی برای مدت زمان بسیار طولانی وجود داشته، تقریبا از همان زمان اختراع چاپخانه در سال ۱۴۳۹ شایع بوده‌اند ولی با افزایش شبکه‌های اجتماعی سرعت انتشار بیشتری پیداکرده‌اند.نگاهی به اوضاع اجتماعی نشان می‌دهد که در مقایسه با رسانه های سنتی، شبکه‌های اجتماعی رشد بیشتری داشتند، به خصوص زمانی که بحرانی ایجاد شود رسانه های اجتماعی خبرهای گسترده تر و به‌روزتری در دسترس قرار می‌دهند [۱۴]. اما مشکلی که وجود دارد این که همه اخبار واقعی نیستند و احتمال تغییر و دستکاری اطلاعات واقعی توسط مردم به‌دلیل انگیزه های سیاسی، اقتصادی و اجتماعی وجود دارد. این اطلاعات دستکاری شده منجر به ایجاد اخباری می‌شودکه ممکن است کاملا درست نباشد و یا ممکن است کاملا غلط نباشد [۳۱]. تعریفی ساده‌تر از اخبار جعلی،مقالاتی خبری است که به‌طور عمدا نادرست هستند ومی‌توانند خوانندگان را گمراه کنند. توزیع اخبار در رسانه های اجتماعی امروزه بسیارمحبوب شده است. رسانه های اجتماعی به دلیل انتشار سریع، هزینه کم و دسترسی آسان، مزایایی را برای کاربران به همراه دارد و توجه زیادی را به خود جلب کرده است [۳۳].

دراین بخش از پایان نامه، چند نمونه از مطالعاتی که در این زمینه انجام شده مورد بررسی قرار گرفته است.همچنین آن‌ها به سه گروه تقسیم‌بندی می‌شود.

۲-۱ مبتنی بر ساختار

یانگ و همکاران در سال ۲۰۱۶ یک شبکه توجه سلسله مراتبی را برای طبقه‌بندی سند پیشنهاد کردند، این شبکه توجه به سطح کلمات و جملات اعمال می‌شود و بر اساس میزان اهمیت وزن خاصی به آن‌ها داده می‌شود، یکی از مزیت‌های این

روش این است که در هر مرحله ویژگی‌های مهم لایه قبل نیز مورداستفاده قرار می‌گیرد اما به دلیل این‌که فقط متن خبر را بررسی می‌کندنسبت به مدل هایی که حاوی متن و تصویر هستند دقت پایین‌تری دارد. این روش روی شش مجموعه‌داده Yelp2013,Yelp2014,Yelp2015,IMDB,Yahoo Answer,Amazon مورد بررسی قرار می‌گیرد و به ترتیب به دقت های، [۶۸.۲ % ، ۷۰.۵ % ، ۷۱ % ،۴۹.۴ % ،۷۵.۸ % ، ۶۳.۶ %] دست می‌یابد [۳۸].

حمید کریمی و همکاران در سال ۲۰۱۹ مدل ساختار گفتمان سلسله مراتبی را ارائه کردند، این مدل ساختار سطح گفتمان را برای مقالات خبری جعلی / واقعی به‌صورت خودکار و داده محور آموزش می‌بیند و می‌سازد. علاوه براین، ویژگی‌های ساختاریافته را شناسایی می‌کند، که می‌تواند ساختارهای کشف‌شده را توضیح دهد و اخبار جعلی را راحت‌تر بفهمند.در این روش پنج مجموعه‌داده مورد استفاده قرار می‌گیرد.دو مجموعه‌داده PolitiFact و Buzzfeed دو مجموعه‌داده بعدی از مجموعه‌داده اخبار جعلی آنلاین موجود در Kaggle استفاده می شود و در نهایت مجموعه‌داده ساخته‌شده توسط McIntire استفاده می‌شود و دقت % ۱۹.۸۲ را به‌دست آوردند [۱۰].

۲-۲ مبتنی بر نظرات کاربران

ناتالی روچانسکی و همکاران در سال ۲۰۱۷ مدلی مبتنی بر نظرات کاربران ارائه دادند که رفتار هر دو طرف،کاربران و مقالات، و رفتار گروهی از کاربران که اخبار جعلی را منتشر می‌کنند را ترکیب می‌کند. این مدل از سه ماژول ثبت، امتیاز، ادغام تشکیل‌شده است. بر روی دو مجموعه‌داده توییتر و ویبو به دقت های، % ۸۹،% ۹۵ دست یافتند [۱۳].

کای شو و همکاران در سال ۲۰۱۹ کشف قابل توضیح اخبار جعلی را موردمطالعه و بررسی قراردادند، یک همبستگی بین محتوای اخبار و نظرات کاربران وجود دارد، این مدل از شبکه‌های توجه برای میزان اهمیت و قابل توجیه بودن کلمات و جملات استفاده می‌کند و جملات و نظرات بااهمیت‌تر را انتخاب می‌کند و برای پیش‌بینی نوع خبر به طبقه بند می‌دهد. در این پژوهش از یک مجموعه‌داده شناسایی اخبار به نام FakeNewsNet استفاده می‌کند که این مجموعه‌داده از دو سیستم عامل GossipCop و PolitiFact تشکیل شده که حاوی محتوای اخبار مانند متن اصلی و تعاملات اجتماعی کاربران مانند نظرات کاربران است. دقت بر روی دو سیستم عامل GossipCop و PolitiFact به‌ترتیب % ۹۰ و % ۸۰ است [۴۱].

یی هان و همکاران در مقاله‌ای در سال ۲۰۲۰ بر تشخیص اخبار جعلی مبتنی بر انتشار تمرکز دارند.با توجه به قابلیت‌های شبکه‌های عصبی گراف GNN [1] در ارتباط با داده‌های غیر اقلیدسی از GNN برای تمایز بین الگوهای انتشار اخبار جعلی و واقعی در رسانه‌های اجتماعی استفاده کردند. از دو مجموعه‌داده PolitiFact GossipCop استفاده می‌شود و دقت آن % ۹۰ و % ۸۶ است [۴۵].

پاکا و همکاران در سال ۲۰۲۱ یک مجموعه داده توییتر با مقیاس بزرگ با برچسب های واقعی و جعلی به نام CTF [2] معرفی کردند. همچنین در این منبع Cross -SEAN را معرفی کردند که یک مدل توجه نیمه نظارتی است که مقدار زیادی از

[1] Graph Neural Networks [2] COVID-19 Twitter Fake News

داده های بدون برچسب را تحت تاثیر قرار می‌دهد. آن ها Cross -SEAN را با هفت روش جدید تشخیص اخبار جعلی مقایسه کردند و در مجموعه‌داده CTF به دقت ۹۵% دست یافتند [۴۷].

ژانگ و همکاران در سال۲۰۲۱ در گزارشی بیان می‌کنند که احساسات دوگانه بین اخبار جعلی و واقعی متمایز است. ویژگی‌های احساسات دوگانه شامل سه جزء است:۱. احساسات ناشر که از محتوا استخراج می شود ۲. احساسات اجتماعی که از نظرات کاربران استخراج می‌شود ۳. شکاف احساسی که نشان دهنده شباهت و تفاوت بین احساسات ناشر و احساسات اجتماعی است. از سه مجموعه‌داده RumourEval-19، Weibo-16، Weibo-20، در این پژوهش استفاده شده و به دقت های % ۷۵، ۹۱% ، ۹۳% دست یافتند [۴۸].

۲-۳ مبتنی بر متن و تصویر

فنگ کیان و همکاران در سال ۲۰۱۸ یک شبکه عصبی کانوولوشن دوسطحی با مولد پاسخ کاربر(TCNN-URG) ارائه دادند که در آن TCNN اطلاعات معنایی از متن مقاله را با ارائه آن در سطح کلمه و جمله دریافت می‌کند،URG یک مدل مولد پاسخ کاربر به متن را از بازخوردهای گذشته کاربر یاد می‌گیرد که می‌تواند از آن برای تولید پاسخ‌ها به مقالات جدید برای کمک به تشخیص اخبار جعلی استفاده کند.همچنین در این پژوهش بر روی مجموعه‌داده ویبو به دقت% ۸۹.۸ دست یافتند [۴۰].

یانگ یانگ و همکاران در سال ۲۰۱۸ مدلی به نام TI-CNN (اطلاعات متنی و تصویری مبتنی بر شبکه عصبی ارائه(CNN دادند، این مدل به‌طور خودکار ویژگی‌های متن و تصویر را استخراج کرده و به‌طور هم‌زمان با اطلاعات متن و تصویر آموزش داده می‌شود. این پژوهش بر روی مجموعه داده Kaggle به دقت% ۹۲ دست یافتند [۱۵].

یاکینگ وانگ و همکاران در سال ۲۰۱۸ مدل EANN را ارائه دادند که شامل سه جز اصلی است: استخراج‌کننده ویژگی چندوجهی، آشکارساز اخبار جعلی، و تفکیک‌کننده رویداد. در این مدل یک تمایز گر رویداد وجود دارد که ویژگی‌های خاص را حذف می‌کند و ویژگی‌های مشترک با طبقه بند نوع برچسب خبر می‌دهد،در این مدل ویژگی‌ها مستقل از نوع خبر است و هر مرحله فقط ویژگی‌های مهم همان مرحله را در نظر می‌گیرد.مجموعه داده مورد استفاده در این پژوهش توییتر و ویبو است و به دقت % ۷۱ و % ۸۲ دست یافتند [۱۱].

خطار و همکاران در سال ۲۰۱۹، یک شبکه رمزگذار خودکار متغیر چندوجهی MVAE را پیشنهاد می‌کند که از ترکیب یک رمزگذار خودکار دوبعدی با یک طبقه بند باینری برای تشخیص اخبار جعلی استفاده می‌کند. این مدل شامل سه جز اصلی، یک رمزگذار، یک رمزگشا و یک ماژول تشخیص دهنده اخبار جعلی است، که رمز گذار اطلاعات را از متن و تصویر به یک بردار مخفی رمزگذاری می‌کند، رمزگشا تصویر و متن اصلی را دوباره بازسازی می‌کند و تشخیص دهنده اخبار جعلی نیز نوع برچسب خبر را پیش‌بینی می‌کند. این مدل بر روی دو مجموعه‌داده توییتر و ویبو به‌ترتیب به دقت% ۷۴و % ۸۲ دست یافند [۴۳].

کالیار و همکاران در سال ۲۰۲۰ یک شبکه عصبی کانوولوشن FND NET را برای شناسایی اخبار جعلی پیشنهاد کردند،در این روش به‌جای استخراج ویژگی به‌صورت دست‌ساز از استخراج ویژگی خودکار برای طبقه‌بندی اخبار جعلی از طریق لایه‌های مخفی ساخته‌شده در شبکه عصبی عمیق استفاده می‌شود.برای استخراج ویژگی از شبکه‌های کانوولوشن در هر لایه استفاده می‌شود. در این مدل از مجموعه‌داده Benchmarked استفاده می‌شود و به دقت % ۹۸.۳ دست یافتند [۴۴].

یی جولو و همکاران در سال ۲۰۲۰ مشکل کشف اخبار جعلی را تحت یک سناریوی واقع‌گرایانه‌تر در رسانه‌های اجتماعی حل کردند. آن‌ها مدل G CAN [1] که یک مدل مبتنی بر شبکه عصبی است را پیشنهاد کردند که در آن با توجه به متن توییت و اظهارنظر کاربران توییت های جعلی را شناسایی می‌کند.از دو مجموعه‌داده Twitter15 و Twitter16 استفاده می‌شود و به دقت ۸۷% و % ۹۰ دست یافتند [۴۶].

از روش های ترکیبی که فقط محتوای متنی اخبار را بررسی می‌کنند ، ترکیب LSTM و CNN نتایج امیدوار کننده ای را نشان داده است. با این حال، تا کنون، LSTM ها برای ارائه جاسازی کلمات و CNN برای طبقه بندی نهایی استفاده شده است. در بخش های بعدی به تعریف و مقایسه دو روش که بر مبنای LSTM و CNN هستند پرداخته می شود.

[1] Graph-aware Co-Attention Networks

جدول ۲-۱: مقایسه روش های موجود

روش	سال انتشار	مجموعه داده	دقت	نوع طبقه بندی
HAN	۲۰۱۶	Yelp۲۰۱۳ Yelp۲۰۱۴ Yelp۲۰۱۵ IMDB Yahoo Answer Amazon	۶۸.۲ ۷۰.۵ ۷۱ ۴۹.۴ ۷۵.۸ ۶۳.۶	مبتنی بر ساختار
CSI	۲۰۱۷	Twitter Weibo	۸۹ ۹۵	مبتنی بر نظرات کاربران
TCNN-URG	۲۰۱۸	Weibo	۸۹.۸	مبتنی بر متن و تصویر
TI-CNN	۲۰۱۸	Kaggle	۹۲	مبتنی بر متن و تصویر
EANN	۲۰۱۸	Twitter Weibo	۷۱ ۸۲	مبتنی بر متن و تصویر
dEFEND	۲۰۱۹	GossipCop PolitiFact	۹۰ ۸۰	مبتنی بر نظرات کابران
HDSF	۲۰۱۹	Kaggle	۸۲.۱۹	مبتنی بر ساختار
MVAE	۲۰۱۹	Twitter Weibo	۷۴ ۸۲	مبتنی بر متن و تصویر
FNDNet	۲۰۲۰	Benchmarked	۹۸.۳	مبتنی بر متن و تصویر
GNN	۲۰۲۰	PolitiFact GossipCop	۹۰ ۸۶	مبتنی بر نظرات کاربرن
G CAN	۲۰۲۰	Twitter15 Twitter16	۸۷ ۹۰	مبتنی بر متن و تصویر
CTF	۲۰۲۱	CTF	۹۵	مبتنی بر نظرات کابران
Dual Emotion Features	۲۰۲۱	RumourEval-۱۹ Weibo-16 Weibo-20	۷۵ ۹۱ ۹۳	مبتنی بر نظرات کابران

فصل ۳

مقایسه دو روش ساختار سلسله مراتبی و تمایز گر رویداد

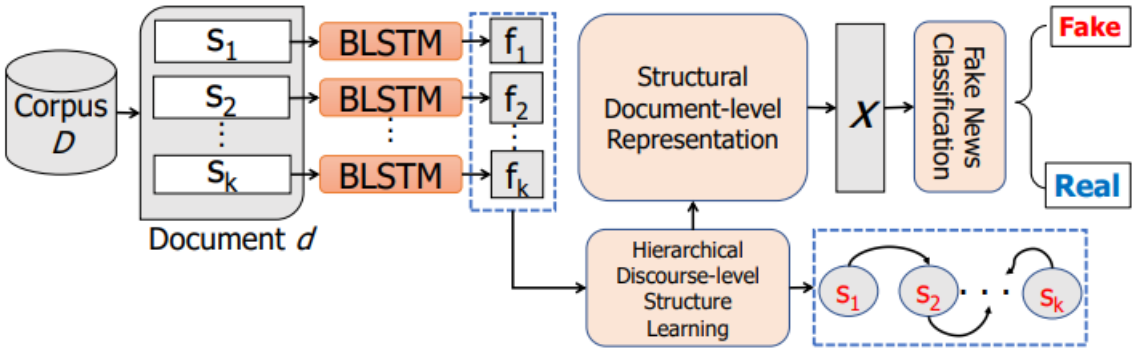
در این فصل در مورد روش های پیشنهادی برای شناسایی اخبار جعلی در شبکه‌های اجتماعی توضیحاتی ارائه می‌شود. در فصل‌های قبلی توضیح داده شد که با توجه به افزایش استفاده از شبکه‌های اجتماعی، برای جلوگیری از تشویش اذهان عمومی به‌خصوص در موارد ضروری از قبیل انتخابات سیاسی، حوادث طبیعی مثل زلزله و ... نیاز است نسبت به شناسایی اخبار جعلی اقدامات لازم صورت پذیرد. یکی از موارد می‌تواند شناسایی آنلاین اخبار جعلی در هنگام انتشار باشد. در این بخش از پایان نامه به دو روش اساسی که بازدهی خوبی در شناسایی اخبار جعلی دارند پرداخته و توضیحاتی ارائه شده است.

۳-۱ یادگیری ساختار سطح گفتمان سلسله مراتبی

در این پژوهش آقای حمید کریمی و همکاران بر محتوای اخبار متمرکز هستند و آموزش و ساخت یک ساختار سطح گفتمان برای مقالات خبری، جعلی و واقعی را موردبررسی قرار می‌دهند، در یک سند واحدهای گفتمان همان جملات هستند. اهمیت در نظر گرفتن ساختار سطح گفتمان سلسله مراتبی [۲] برای کشف اخبار جعلی به سه شکل است. اول، بررسی‌های انجام شده در ساختار سطح گفتمان برای کشف اخبار جعلی نشان می دهدکه روش اتصال دو واحد گفتمان یک سند می‌تواند در کشف اخبار جعلی مفید باشد. دوم این که اخبار جعلی معمولاً از خبرهای متفرقه به وجود می‌آیند وبرخلاف اخبار موثق و واقعی، این اخبار هیئت تحریریه دقیقی برای نظارت ندارد. بنابراین، با ترکیب ساختار سطح گفتمان سلسله مراتبی، می‌توان انسجام اسناد خبری جعلی، واقعی را بررسی کرد. سوم مطالعات نشان داده که استفاده از ساختار سلسله مراتبی بهتر و مفیدتر است. از آنجا که معمولاً کشف اخبار جعلی به عنوان یک مشکل طبقه‌بندی براساس کل متن در نظر گرفته می‌شود،

[2] Hierarchical Discourse-level Structure for Fake News Detection

استفاده از تجزیه و تحلیل گفتمان برای تشخیص اخبار جعلی مفید است. در این مدل یک ساختار سطح گفتمان برای مقالات خبری جعلی، واقعی به صورت خودکار و داده محور آموزش می‌بیند و می‌سازد. همچنین ویژگی‌های ساختار یافته را شناسایی می‌کند، که می‌تواند ساختارهای کشف شده را توضیح دهد و اخبار جعلی را راحت‌تر بفهمند. در این مدل، هدف ساخت یک ساختار سلسله مراتبی بین واحدهای گفتمان (یعنی جملات) می‌باشد. برای رسیدن به این هدف، رویکرد تجزیه وابستگی و ساختار سلسله مراتبی یک سند رابه صورت یک درخت وابستگی نمایش می‌دهد. در تجزیه وابستگی واحدهای گفتمان، باید تشخیص داده شود که آیا یک واحد گفتمان از نظر معنایی به واحدهای دیگر مرتبط است یا خیر. اگر مرتبط باشد، یک شاخه والد - فرزند (یا دنباله - سر) در درخت وابستگی می‌تواند ایجاد می‌شود. بنابراین، تا زمانی که ارتباط معنایی بین واحدهای گفتمان برقرار باشد، می‌توان یک درخت وابستگی گفتمانی ساخت. از آنجا که واحدهای گفتمان به عنوان جملات تعریف می‌شوند، ابتدا باید یک نمایش ثابت برای هر جمله وجود داشته باشد برای این منظور، از شبکه حافظه کوتاه مدت دو جهته BLSTM [1] استفاده می‌شود. شکل ۳-۱ بیانگر ساختار سلسله مراتبی اسناد است.



شکل ۳-۱: چارچوب پیشنهادی ساختار سطح گفتمان[۱۰]

همان طور که گفته شد در این روش از شبکه حافظه کوتاه‌مدت دو جهته (BLSTM)استفاده می‌شود. ابتدا هر کلمه در یک جمله s_j با یک کلمه نگاشت اندازه ثابت نشان داده می‌شود، و سپس شبکه BLSTM در هر گام زمانی $t \in [1, T_j]$ تابع‌های زیر را اجرا می‌کند:

$$\overrightarrow{h}_t = \mathcal{F}(\overrightarrow{h}_{t-1}, w_{t-1}) \qquad (۱-۳)$$

$$\overleftarrow{h}_t = \mathcal{F}(\overleftarrow{h}_{t-1}, w_{T_j-t+1}) \qquad (۲-۳)$$

$\mathcal{F}$ یک تابع LSTM است و $\overrightarrow{h}_t$ و $\overleftarrow{h}_t$ به ترتیب خروجی‌های شبکه‌های روبه‌جلو و رو به عقب LSTM در گام زمانی t هستند. سپس، یک نمایش ثابت برای جمله S_j ، که به‌صورت f_j نشان داده می‌شود، به‌عنوان میانگین آخرین خروجی

[1] Bidirectional Long Short-Term Memory

شبکه‌های روبه‌جلو و رو به عقب LSTM تعریف می‌شود:

$$f_j = \frac{[\overrightarrow{h}_{T_j} + \overleftarrow{h}_{T_j}]}{۲} \quad (۳-۳)$$

به‌طور مشابه، شبکه BLSTM برای تمام جملات یک سند اعمال می‌شود و یک دنباله از نمایش‌های جمله‌ای به‌صورت $f_۱, f_۲, ... f_k$ به دست می‌آید؛(شکل ۳-۱ را ببینید)

این مدل به دنبال استخراج اطلاعات مفید و کاربردی از ساختار برای نمایش تفاوت بین اسناد جعلی و واقعی است و برای نمایش آن سه ویژگی زیر بررسی می‌شود:

۱. تعداد گره‌های برگ: هر چقدر تعداد گره‌های برگ بیشتر باشد واحدهای گفتمان دارای ارتباط درونی کمتری خواهند بود بنابراین انسجام کم‌تر است، در واقع احتمالات را برای ریشه و برگ در نظر می‌گیرد وقتی برگ‌ها زیاد باشد درخت عمق ندارد و فقط تعداد برگ‌های آن زیاد شده است و احتمال جعلی بودن خبر زیاد است.

۲. تفاوت فاصله بین موقعیت الان و حالت اصلی: هر جمله در جای مناسب خود قرار می‌گیرد. اگر درجایی غیر مرتبط بود فاصله حالتی که در آن هست با حالتی که بهتر است باشد اندازه گرفته می‌شود. هرچه این فاصله کم‌تر باشد یعنی جمله در جای مناسبی قرار گرفته است و منسجم‌تر است بنابراین برچسب واقعی می‌خورد.

۳. فاصله گره ریشه و برگ: این ویژگی مجموع فاصله موقعیتی بین گره‌های برگ و ریشه را در حالتی که در جای مناسب قرار گرفته باشد می‌سنجد، اگر این فاصله کم باشد یعنی جملات در جایگاه خود است و به ترتیب پشت سر هم آمده‌اند و اگر با گره ریشه فاصله داشت یعنی انسجام خوبی ندارد و برچسب نوع خبر جعلی است. در این مدل دو مشاهده کلیدی وجود دارد:

- این مدل این واقعیت را نشان می‌دهد که ساختارهای اسناد اخبار جعلی در سطح گفتمان کاملاً متفاوت از اسناد واقعی هستند.

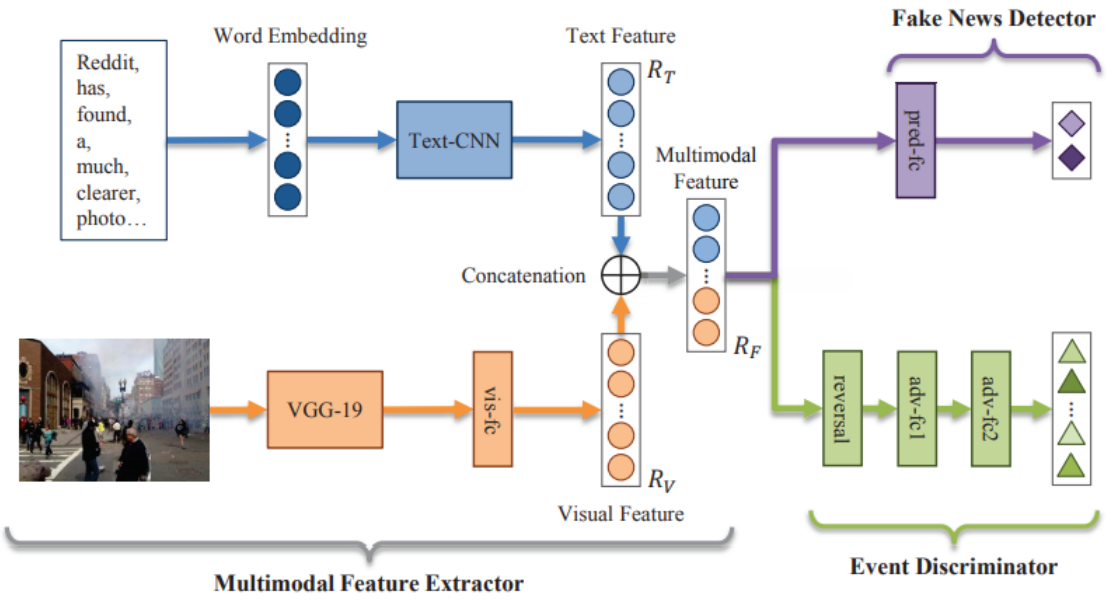
- هر چه انسجام جملات کم‌تر باشد یعنی جملات به یکدیگر مرتبط نیستند و نوع خبر جعلی است و هرچه انسجام بیشتر باشد یعنی نوع خبر، واقعی است.

۳-۲ شبکه‌های عصبی تمایزگر رویداد برای تشخیص اخبار جعلی چند حالت

روش دیگری که در اینجا موردتوجه قرار می‌گیرد چارچوب شبکه‌های عصبی تمایزگر رویداد [1] است که توسط آقای یانگ وانگ ارائه شده است. در مقایسه با اخبار متنی، اخباری که حاوی تصاویر و ویدیو هستند بهتر می‌توانند توجه خوانندگان را جلب کنند. در این روش بیشتر روی ویژگی‌های مشترک رویدادها متمرکز می‌شود تا از طریق حفظ ویژگی‌های ثابت بتواند

[1] EANN:Event Adversarial Neural Network

به تشخیص اخبار جعلی کمک کند و ویژگی‌های خاص را حذف می‌کند.این مدل سه مؤلفه اصلی دارد: ۱. استخراج‌کننده ویژگی چند حالت ۲. ردیاب اخبار جعلی ۳. تفکیک‌کننده رویداد. برای استخراج ویژگی‌های چندحالت از شبکه‌های عصبی CNN استفاده می‌شود تا به طور خودکار ویژگی‌هایی از محتوای متن و تصاویر پست‌ها استخراج کند. استخراج کننده ویژگی چندوجهی استخراج ویژگی‌های متن و تصویر را برعهده دارد و با کمک ردیاب اخبار جعلی، نمایش‌های قابل تشخیص برای شناسایی اخبار جعلی را آموزش می‌بیند و به طور همزمان با حذف ویژگی‌های خاص رویداد، ویژگی‌های مشترک رویداد را حفظ می کندکه در آن استخراج کننده ویژگی، دوشاخه ویژگی متن و ویژگی تصویر دارد، برای استخراج ویژگی متنی از Text-CNN و برای استخراج ویژگی تصویر از شبکه VGG-۱۹ استفاده می‌شود، سپس ویژگی‌های استخراج شده را با ویژگی‌های موجود در تمایزگر رویداد مقایسه می‌کند، اگر ویژگی‌ها مشترک بود یعنی خبر جعلی است و خروجی با ردیاب اخبار جعلی نمایش داده می‌شود. دراین مرحله شبکه اقدام به حذف رویدادهای خاص می‌کند. ویژگی‌های تک حالته تنها شامل ویژگی‌های متنی می‌شود که از متن خبر استخراج شده و برای کشف اخبار جعلی استفاده می‌شود. در ویژگی‌های چند حالت که شامل متن و تصاویر است که برای تمامی ورودی‌هاست پس از یادگیری متن و نمایش ویژگی‌های مخفی متن و تصویر، آن‌ها با یکدیگرترکیب می‌شوند تا نمایش ویژگی چند حالت نهایی را شکل دهند که از متمایزکننده رویداد و تشخیص دهنده اخبار جعلی ساخته شده‌اند. تشخیص دهنده اخبار جعلی ویژگی‌های آموخته شده را به عنوان ورودی برای پیش‌بینی جعلی یا واقعی بودن پست‌ها در نظر می‌گیرد، متمایز کننده رویداد بر اساس این ویژگی‌ها، برچسب رویداد هر پست را مشخص می‌کند. در شکل ۳-۲ معماری EANN نشان داده می‌شود.



شکل ۳-۲: معماری شبکه EANN [۱۱]

در این شکل معماری شبکه عصبی EANN نشان داده‌شده است، همان‌طور که مشاهده می‌شود یک مجموعه داده متنی و یک مجموعه داده تصویر وجود دارد که شبکه رنگ آبی یک ویژگی متنی و رنگ نارنجی ویژگی تصویر است سپس

این ویژگی‌ها با یک‌لایه تماما متصل به هم متصل می‌شوند. این ویژگی‌ها به تمایز گر رویداد می‌روند که به رنگ سبز است در آنجا ویژگی‌های خاص یک رویداد حذف می‌شود و ویژگی‌های مشترک باقی می‌ماندن یا به تشخیص‌دهنده اخبار جعلی که به رنگ بنفش است داده می‌شود تا برچسب خبر مشخص شود. هدف اصلی این مدل یادگیری انتقال و تفکیک‌پذیری نمایش ویژگی برای اخبار جعلی است. همان‌طور که در معماری شبکه EANN مشاهده شد برای رسیدن به این هدف باید سه مؤلفه استخراج‌کننده ویژگی ،تشخیص‌دهنده اخبار جعلی و تفکیک‌کننده رویداد با یکدیگر ادغام شوند. استخراج ویژگی چندحالت شامل ویژگی متنی و تصویری است که برای انواع مختلف ورودی‌هاست. پس از یادگیری متن و نمایش ویژگی‌های پنهان متن و تصویر، آن‌ها با یکدیگر ترکیب می‌شوند تا نمایش ویژگی چند حالت هایی را شکل دهند که هر دو از تشخیص‌دهنده اخبار جعلی و تفکیک‌کننده رویداد در بالای استخراج‌کننده ویژگی چندحالت ساخته‌شده‌اند. تشخیص‌دهنده اخبار جعلی خروجی تمایزگر رویداد که شامل ویژگی‌های مشترک هست را به‌عنوان ورودی برای پیش‌بینی جعلی یا واقعی بودن اخبار در نظر می‌گیرد.

استخراج ویژگی متنی :

به‌منظور استخراج ویژگی‌های متنی، از شبکه‌های عصبی کانولوشن [1] به‌عنوان ماژول اصلی استخراج‌کننده ویژگی‌های متنی استفاده می‌شود. ثابت‌شده است که CNN در بسیاری از زمینه‌ها مانند بینایی رایانه‌ای و طبقه‌بندی متن مؤثر است. همان‌طور که در معماری EANN مشاهده شد، یک مدل اصلاح‌شده ،CNN یعنی Text CNN در استخراج‌کننده ویژگی متنی وجود داردبرای استخراج ویژگی‌های متنی، با استفاده از تعبیه کلمات هر کلمه در متن به‌صورت یک بردار نشان داده می‌شود و درنهایت یک بردار ویژگی متنی تشکیل می‌شود. برای هر بردار ویژگی از max pooling برای گرفتن بیشترین مقدار استفاده می‌شود تا مهم‌ترین اطلاعات استخراج شود، ویژگی‌های متنی بعد از max pooling با R_T نمایش داده می‌شود.

استخراج ویژگی تصویری :

تصاویری که در اخبار وجود دارد به‌عنوان ورودی استخراج‌کننده ویژگی تصویر هستند که با v نشان داده می‌شود، برای استخراج ویژگی‌های تصویر از شبکه آموزش‌دیده VGG۱۹ استفاده می‌شود. در طول فرایند آموزش با استخراج‌کننده ویژگی متنی پارامترهای از پیش آموزش‌دیده‌ی شبکه عصبی VGG۱۹ برای جلوگیری از بیش‌برازش ثابت نگه‌داشته می‌شود. ویژگی تصویر ناشی از آموزش شبکه با VGG۱۹ نمایش داده می‌شود. پس از استخراج ویژگی تصویر با استفاده از max pool-ing کاهش بعد انجام می‌شود و سپس از یک‌لایه تماما متصل [2] استفاده می‌شود و ویژگی‌های متن و تصویر به هم متصل می‌شوند و با R_F نمایش داده می‌شود.در استخراج ویژگی‌های تصویر پایین بودن کیفیت تصاویر و یا عدم تطابق متن و تصویر به کشف اخبار جعلی کمک می‌کند، برای مثال در اخبار جعلی بیشتر از اشیا حیوانات و.. استفاده می‌شود. ردیاب اخبار جعلی :

در این بخش، ردیاب اخبار جعلی معرفی می‌شود که ویژگی‌های چند حالت که شامل متن و تصویر است را به‌عنوان ورودی می‌گیرد سپس در یک‌لایه تماما متصل با استفاده از تابع soft max نوع خبر را مشخص می‌کند.

تمایز گر رویداد : تمایز گر رویداد اقدام به حذف ویژگی‌های خاص یک رویداد می‌کند و ویژگی‌های مشترک را نگه می‌دارد.

[1] CNN [2] Fully Connected

یک شبکه عصبی است که از دولایه کاملاً متصل با توابع فعالیت مربوط تشکیل‌شده است. هدف آن طبقه‌بندی صحیح برچسب خبر است.

در ادامه در رابطه با نحوه ی استخراج ویژگی های متن و تصویر مطالبی ارائه خواهد شد.

- استخراج ویژگی متنی:

ورودی بردار ویژگی متنی، متن پست است و خروجی ویژگی‌هایی است که از حالت متنی آموزش داده می‌شود. استخراج‌کننده ویژگی متنی را می‌توان به دو مرحله تقسیم کرد:

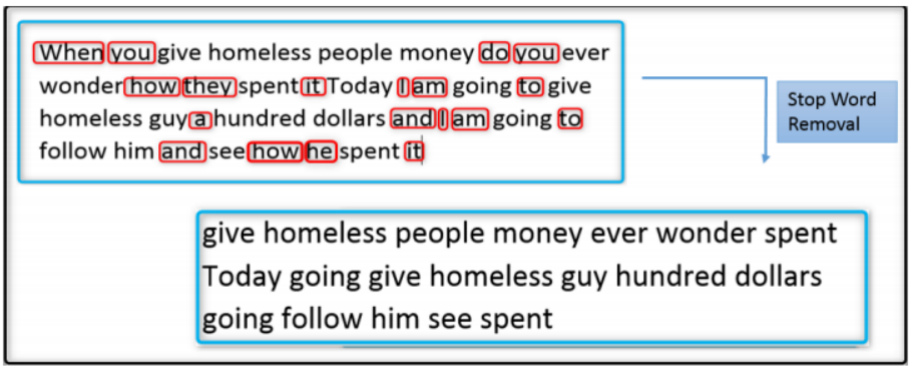
۱. فرایند پیش‌پردازش [۱]

۲. فرایند احساسات [۲] .

۱. فرایند پیش‌پردازش : برای اعمال الگوریتم های یادگیری ماشین بر روی‌داده‌های متنی نیاز به‌پیش پردازش خاصی است. تکنیک‌های متعددی برای تبدیل داده‌های متنی به یک‌شکل مدل‌سازی آماده وجود دارد. مراحل پیش‌پردازش که در ادامه موردبحث قرار می‌گیرد مربوط به عناوین و مقالات خبری است.

- حذف کلمات توقف : مرحله پیش‌پردازش با حذف کلمات توقف

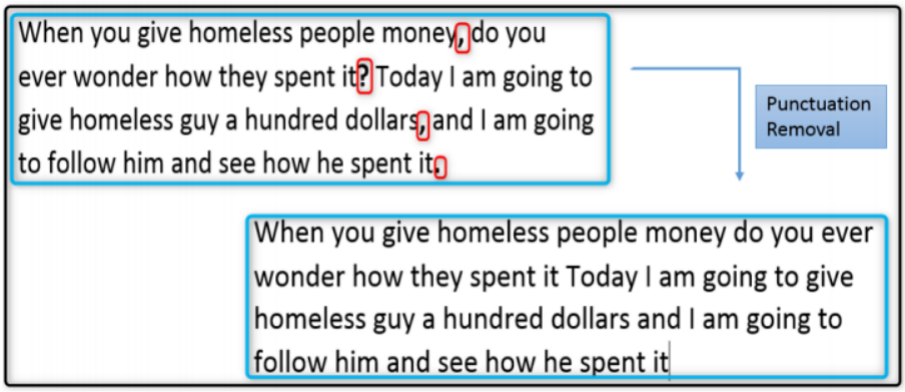
[۳] از داده‌های متنی شروع می‌شود. این کلمات به این دلیل که در متن به‌وفور یافت می‌شوند و هم این‌که فایده خاصی ندارند حذف می‌شوند. برای مثال حروف ربط مانند و، یا، اما، و حروف اضافه مانند از، در، با، و غیره از جملات حذف می‌شود. کلمات توقف بااینکه از اهمیت کمی برخوردار هستند اما بخش مهمی از زمان پردازش را به خود اختصاص می‌دهند و حذف این کلمات به‌عنوان بخشی از پیش‌پردازش داده‌ها اولین گام کلیدی در پردازش زبان طبیعی است.



شکل ۳-۳: نمونه ای از حذف کلمات توقف[۵۰]

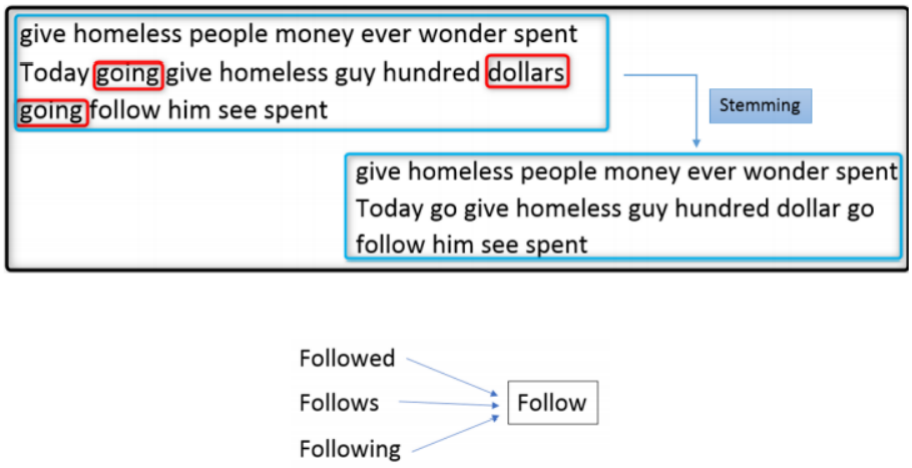
[1]Preprocessing step [2]Sentiment process [3]Stop Word

- حذف علائم نگارشی [1] : در این بخش علائم نگارشی اضافی حذف می‌شوند زیرا این علائم مانند کاما، نقطه، علامت سؤال و ..ارزش چندانی برای درک جمله ندارند.



شکل ۳-۴: نمونه ای از حذف علائم نگارشی[۵۰]

- کوتاه کردن [2] : تکنیکی است که در آن پیشوند و پسوندها از یک کلمه حذف می‌شوند.



شکل ۳-۵: نمونه ای از حذف علائم نگارشی[۵۰]

۲. فرایند احساسات

برای استخراج ویژگی‌های آموزنده و معنی دار از محتوای متنی، از تجزیه و تحلیل احساسات استفاده می‌شود. تجزیه و تحلیل احساسات شامل اظهار نظرات کاربران است که نظرات مثبت، منفی و خنثی دارند و یا شامل ارزیابی وضعیت نویسنده است. برای مثال هیجان زده یا عصبی یا پرخاشگر و.. ممکن است باشد. در زمینه تحلیل

[1]Punctuation [2]Stemming

احساسات، از احساسات متنی برای سیاست‌های منفی، مثبت و خنثی یک پیام متنی استفاده می‌شود. طبقه‌بندی را می توان با استفاده از رویکرد مبتنی بر واژه‌نامه انجام داد. مراحل تحلیل احساسات در الگوریتم زیر خلاصه شده است. احساسات در توصیف نوع خبر بسیار مؤثر است وهمچنین باعث افزایش دقت نیز می‌شود.

Inputs : Text File τ , the sentiment lexicon x
Output: S = P, N_g, N and Strength S, where P: Positive, N_g:Negative,N:Neutral.
Initialization: $SumPos$ and $SumNeg = 0$, where
$SumPos$: collects the polarity of positive tokens $t_{i-_{smt}}$ in τ
$SumNeg$: collects the polarity of negative tokens $t_{i-_{smt}}$ in τ
begin
  foreach $t_i \in \tau$ **do**
    Search for t_i in x
    if $t_i \in Pos - list$ **then**
      SumPos $\leftarrow$ $SumPos$+t_{i-smt}
    end
    else if $t_i \in Neg - list$ **then**
      SumNeg $\leftarrow$ $SumNeg$+t_{i-smt}
    end
  end
  if $SumPos > |SumNeg|$ **then**
    $S_{mt} = P$
    S =SumPos/(SumPos+SumNeg)
  end
  else if $SumPos < |SumNeg|$ **then**
    $S_{mt} = N_g$
    S =SumNeg/(SumPos+SumNeg)
  end
  else
    $S_{mt} = N$
    S =SumPos/(SumPos+SumNeg)
  end
end

شکل ۳-۶: فرایند احساسات برای متن[۵۰]

- استخراج کننده ویژگی تصویری:

تصاویر پیوست شده در محتوای اخبار ورودی استخراج‌کننده ویژگی تصویری است. به‌منظور استخراج کارآمد ویژگی‌های تصویری، ابتدا تصویر پیش‌پردازش می‌شود سپس تصاویر تقسیم‌بندی می‌شود.

- مراحل پیش‌پردازش

۱. اندازه همه تصاویر را به ۲۰۰*۲۰۰ تغییر داده می‌شود

۲.تصویر به مقیاس خاکستری [1] تبدیل می‌شود

۳.برای تبدیل تصویر سطح خاکستری به دودویی [2] از یک مقدار آستانه استفاده می‌شود.

آستانه گذاری با تبدیل تمام پیکسل‌های زیر آستانه به صفر و تمام پیکسل‌های بالای آستانه به یک که تصاویر دودویی خاکستری‌رنگ ایجاد می‌کند.

شکل ۳-۷: نمونه ای از پیش پردازش تصاویر[۵۰]

- فرایند تقسیم‌بندی: تقسیم‌بندی تصاویر روشی است که یک تصویر دیجیتالی را به چند بخش تقسیم می‌کند (مجموعه‌های پیکسل، که اغلب به‌عنوان اشیاء تصویر شناخته می‌شوند). هدف تقسیم‌بندی ساده‌سازی و یا تغییر نمایش تصویر به چیزی است که برای ارزیابی مناسب‌تر و ساده‌تر است.

[1] gray [2] binarization

فصل ۴

نتایج

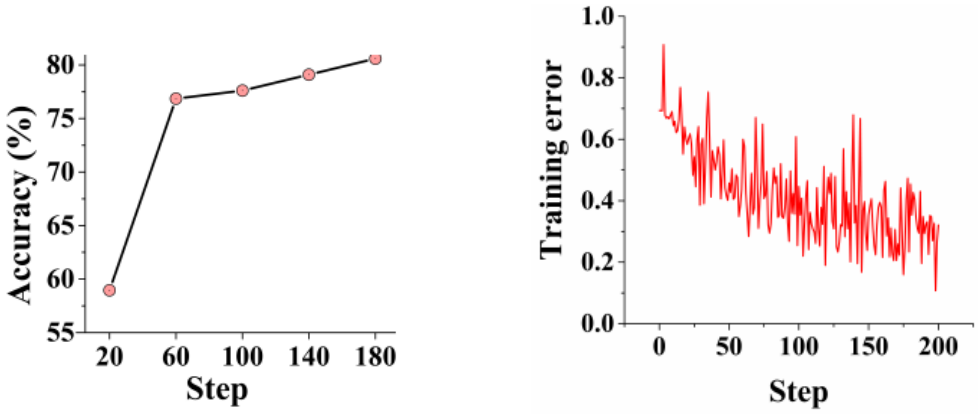
در این بخش نتایج به دست آمده از دو روش HDSF و EANN ارائه شده است و در ادامه نتایج مقایسه این دو روش روی مجموعه داده مشترک ارائه خواهد شد.

۴-۱ نتایج اجرای ساختار گفتمان سلسله مراتبی

ابتدا مجموعه داده مورد نیاز در این روش را معرفی نموده و بررسی های لازم انجام می‌شود.

۴-۱-۱ مجموعه داده :

در این تحقیق از پنج مجموعه داده اخبار جعلی موجود استفاده می‌شود، دو مجموعه داده اول توسط (شو و همکاران، ۲۰۱۷) [۵۱] جمع‌آوری‌شده و شامل مقالات آنلاین است که درستی آن‌ها توسط متخصصان BuzzFeed و PolitiFact تأیید شده است. برای دو مجموعه داده بعدی، از دو مجموعه داده اخبار جعلی آنلاین موجود در kaggle.com استفاده می‌شود. درنهایت مجموعه داده ساخته‌شده توسط McIntire استفاده می‌شود. ازآنجایی‌که چارچوب پیشنهادی HDSF یک چارچوب هدف کلی برای بررسی ساختار سطح گفتمان اسناد جعلی / واقعی بر اساس محتوای متنی ست، به منبع خاصی از داده محدود نمی‌شود و بنابراین همه مجموعه داده‌ها باهم ترکیب می‌شود. در کل، ۳۳۶۰ سند جعلی و ۳۳۶۰ سند واقعی وجود دارد.



(آ) خطای مجموعه آموزش

(ب) دقت روی مجموعه توسعه

شکل ۴-۱: بررسی HDSF [۱۰]

۴-۱-۲ تنظیمات ازمایش :

ابتدا پیش پردازش داده ها با حذف حروف اضافه،حذف اعداد، حرف حروف غیر انگلیسی انجام می‌شود. از بین این مجموعه داده ۱۳۴ سند به‌عنوان مجموعه توسعه، ۱۳۴ سند به‌عنوان مجموعه آزمایش انتخاب می‌شود. ۶۴۵۲ سند باقیمانده برای آموزش استفاده می‌شود. از تابع فعالیت RELU و همچنین تابع بهینه ساز ADAM در این تحقیق استفاده می‌شود. تعدا لایه های مخفی در شبکه، BLSTM برابر ۱۰۰و اندازه دسته‌ها در هر مرحله ۴۰ است. (bach size=40) ، نرخ یادگیری نیز ۰.۱ است.

۴-۱-۳ نتایج بدست امده از اجرای HDSF

شکل ۴-۱ (آ) خطای آموزش را هنگام بهینه‌سازی مدل نشان می‌دهد. همان‌طور که از این شکل مشاهده می‌شود، با ادامه روند آموزش، خطا در حال کاهش است. علاوه بر این، شکل(ب)دقت در مجموعه‌ای را که در طول آموزش وجود دارد نشان می‌دهد و با ادامه آموزش به‌صورت یکنواخت در حال افزایش است. ازاین‌رو، بر اساس این ارقام، می‌توان اطمینان حاصل کرد که این روش بهینه است و می‌توان اسناد اخبار جعلی را به‌درستی طبقه‌بندی کرد.

در جدول ؟؟ نتایج روش HDSF آورده شده است و همانطور که مشاهده می کنید این روش از دقت ۸۲/۱۹ درصد برخوردار است.

Method	Accuracy
HDSF	۸۲.۱۹

جدول ۴-۱: دقت روش HDSF

۴-۲ نتایج اجرای شبکه عصبی تمایزگر رویداد

در این بخش، ابتدا دو مجموعه داده بزرگ رسانه‌های اجتماعی مورداستفاده در آزمایش‌ها معرفی می‌شود، سپس عملکرد مدل با مدل متنی و تصویری و مدل EANN- بررسی می شود و نتایج ارائه می‌شود.

۴-۲-۱ مجموعه داده :

برای ارزیابی عملکرد مدل از دو مجموعه داده‌ی توییتر و ویبو استفاده‌شده است که در دسترس عموم قرار دارد و شامل محتوای متن و تصویر است.

Twitter: مجموعه داده توییتر به‌عنوان بخشی از mediaeval برای تائید مطالب چندرسانه‌ای منتشر شد. هدف این کار شناسایی مطالب چندرسانه‌ای جعلی بود، این مجموعه داده از تعدادی توییت تشکیل‌شده است و هر توییت حاوی مطالب متنی،تصویر،فیلم و.. است. شامل ۱۷۰۰۰ توییت است که حاوی رویدادهای مختلف است. به دو قسمت مجموعه آزمایشی و مجموعه توسعه تقسیم می‌شود. مجموعه توسعه حاوی ۹۰۰۰ توییت خبری جعلی و ۶۰۰۰ توییت خبری واقعی است و مجموعه آزمایشی نیز حاوی ۲۰۰۰توییت است. چون تمرکز روی محتوای متن و تصویر است از توییت های حاوی ویدئو صرف‌نظر می‌شود.

Method	Twitter	Weibo
# of fake News	7898	4749
# of real News	6026	4779
# of images	514	9528

شکل ۴-۲: بررسی مجموعه داده[۱۱]

Weibo: مجموعه داده Weibo برای تشخیص اخبار جعلی استفاده می‌شود. در این مجموعه داده، اخبار واقعی از منابع خبری معتبر چین، مانند خبرگزاری شینهوا جمع‌آوری‌شده است. این سیستم به‌عنوان منبع معتبری برای جمع‌آوری شایعات محسوب می‌شود. ابتدا تصاویر تکراری و باکیفیت پایین حذف می‌شود تا کیفیت کل مجموعه داده حفظ شود.

۴-۲-۲ نتایج به دست آمده از اجرای EANN

در این بخش مدلEANN را با چند مدل دیگر که در ادامه به توضیح آن پراخته می شود مقایسه کرده و نتایج ارائه می‌شود.

۴-۲-۳ مدل -EANN

مدل EANN شامل استخراج‌کننده ویژگی چند حالتِ، آشکارساز اخبار جعلی و تفکیک‌کننده رویداد است. یک مدل پیشنهادی، به نام −EANN طراحی می‌شود که متمایزگر رویداد را شامل نمی‌شود. جدول زیر نتایج تجربی مدل EANN و -EANN را در دو مجموعه داده نشان می‌دهد. طبق مشاهدات عملکرد کلی EANN پیشنهادی ازنظر صحت، دقت و رتبه F۱ بسیار بهتر از خطوط پایه است. در مدل EANN از تابع فعالیت RELU و همچنین تابع بهینه ساز ADAM و همچنین تابع هزینه cross-entropy استفاده می‌شود

۴-۲-۴ مدل متنی

در این مدل استخراج ویژگی های تصویری حذف می‌شود و فقط از ویژگی های متنی استفاده می کند و برای استخراج ویژگی های متنی R_T از CNN استفاده می‌شود.

۴-۲-۵ مدل تصویری

در این مدل استخراج ویژگی های متنی حذف می شود و فقط ویژگی های تصویری استخراج می‌شوند. در این مدل از شبکه VGG-۱۹ از پیش آموزش‌دیده و یک‌لایه کاملاً متصل برای استخراج ویژگی های تصویری R_V استفاده می‌شود. ویژگی های تصویری R_V به یک‌لایه کاملاً متصل برای پیش‌بینی نوع برچسب خبر متصل می‌شود. جدول ؟؟ عملکرد دو مدلEANN و HDSF را بر روی دو مجموعه داده توییتر و ویبو نمایش می‌دهد.

در ادامه به برسی و مقایسه این دو روش بر روی مجموعه داده مشترک پرداخنه و نتایج در قالب جدول ارائه می‌شود.

جدول ۴-۳ نتایج مقایسه دو روش HDSF و EANN با ۲۰۰ تکرار است، همانطور که مشاهده می‌شود مدل HDSF بر روی مجموعه داده توییتر و ویبو دارای دقت ۵۰ درصد است و مدل EANN بر روی مجموعه داده توییتر دارای دقت ۶۳.۲ و بر روی مجموعه داده ویبو دارای دقت ۶۸.۷ است،که نشان دهنده عملکرد بهتر مدل EANN است.

جدول ۴-۲: مقایسه عملکرد روش های مختلف

F۱	Recall	Precision	Accuracy	Method	Dataset
۰.۵۶۸	۰.۵۴۱	۰.۵۹۸	۰.۵۳۲	Text	Twitter
۰.۵۹۳	۰.۵۱۸	۰.۶۹۵	۰.۵۹۶	vis	
۰.۶۱۷	۰.۴۹۸	۰.۸۱۰	۰.۶۴۸	EANN-	
۰.۷۱۹	۰.۶۳۸	۰.۸۲۲	۰.۷۱۵	EANN	
۰.۷۴۸	۰.۶۸۳	۰.۸۲۷	۰.۷۶۳	Text	Weibo
۰.۶۴۵	۰.۶۷۷	۰.۶۱۵	۰.۶۱۵	vis	
۰.۸۰۰	۰.۷۹۵	۰.۸۰۶	۰.۷۹۵	EANN-	
۰.۸۲۹	۰.۸۱۲	۰.۸۴۷	۰.۸۲۷	EANN	

جدول ۴-۳: مقایسه عملکرد روش های مختلف

Accuracy	Data set	Metod
۵۰	Twitter	HDSF
۵۰	Weibo	
۶۳.۲	Twitter	
۶۸.۷	Weibo	EANN

فهرست منابع

[1] https://www.frontiersin.org/articles/10.3389/fgene.2019.00214/full. https://www.frontiersin.org/articles/10.3389/fgene.2019.00214/full. *arXiv preprint arXiv:1903.07389*, 2019.

[2] https://www.andreaperlato.com/aipost/the-activation function/. https://www.andreaperlato.com/aipost/the-activation-function/. *arXiv preprint arXiv:1903.07389*, 2019.

[3] https://www.matlabdl.com/backpropagation-algorithm-in matlab/. https://www.matlabdl.com/backpropagation-algorithm-in-matlab/. *arXiv preprint arXiv:1903.07389*, 2019.

[4] http://mehrdadsalimi.blog.ir/1396/02/25/Understanding LSTM-Networks. http://mehrdadsalimi.blog.ir/1396/02/25/understanding-lstm-networks. *arXiv preprint arXiv:1903.07389*, 2019.

[5] https://www.analyticssteps.com/blogs/common-architectures-convolution-neural networks. https://www.analyticssteps.com/blogs/common-architectures-convolution-neural-networks. *arXiv preprint arXiv:1903.07389*, 2019.

[6] https://www.soalina.com/ https://www.soalina.com/ *arXiv preprint arXiv:1903.07389*, 2019.

[7] https://www.i2tutorials.com/explain-concepts-of-pooling-in-convolutional-neural network/. https://www.i2tutorials.com/explain-concepts-of-pooling-in-convolutional-neural-network/. *arXiv preprint arXiv:1903.07389*, 2019.

[8] https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg. https://upload.wikimedia.org/wikipedia/commons/2/26/precisionrecall.svg. *arXiv preprint arXiv:1903.07389*, 2019.

[9] Qi, Peng, Cao, Juan, Yang, Tianyun, Guo, Junbo, and Li, Jintao. Exploiting multi-domain visual information for fake news detection. in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 518–527. IEEE, 2019.

[10] Karimi, Hamid and Tang, Jiliang. Learning hierarchical discourse-level structure for fake news detection. *arXiv preprint arXiv:1903.07389*, 2019.

[11] Wang, Yaqing, Ma, Fenglong, Jin, Zhiwei, Yuan, Ye, Xun, Guangxu, Jha, Kishlay, Su, Lu, and Gao, Jing. Eann: Event adversarial neural networks for multi-modal fake news detection. in *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, pp. 849–857, 2018.

[12] Goldani, Mohammad Hadi, Momtazi, Saeedeh, and Safabakhsh, Reza. Detecting fake news with capsule neural networks. *arXiv preprint arXiv:2002.01030*, 2020.

[13] Ruchansky, Natali, Seo, Sungyong, and Liu, Yan. Csi: A hybrid deep model for fake news detection. in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806, 2017.

[14] Shu, Kai, Sliva, Amy, Wang, Suhang, Tang, Jiliang, and Liu, Huan. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[15] Yang, Yang, Zheng, Lei, Zhang, Jiawei, Cui, Qingcai, Li, Zhoujun, and TI-CNN, Philip S Yu. Convolutional neural networks for fake news detection. *arXiv preprint arXiv:1806.00749*, 2(6), 2018.

[16] Ross, Lee, Ward, Andrew, et al. Naive realism in everyday life: Implications for social conflict and misunderstanding. *Values and knowledge*, 103:135, 1996.

[17] Nickerson, Raymond S. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, 2(2):175–220, 1998.

[18] Shu, Kai, Mahudeswaran, Deepak, Wang, Suhang, Lee, Dongwon, and Liu, Huan. Fakenewsnet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[19] Singh, Vivek, Dasgupta, Rupanjal, Sonagra, Darshan, Raman, Karthik, and Ghosh, Isha. Automated fake news detection using linguistic analysis and machine learning. in *International conference on social computing, behavioral-cultural modeling, & prediction and behavior representation in modeling and simulation (SBP-BRiMS)*, pp. 1–3, 2017.

[20] Pratiwi, Inggrid Yanuar Risca, Asmara, Rosa Andrie, and Rahutomo, Faisal. Study of hoax news detection using naïve bayes classifier in indonesian language. in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, pp. 73–78. IEEE, 2017.

[21] Kesarwani, Ankit, Chauhan, Sudakar Singh, and Nair, Anil Ramachandran. Fake news detection on social media using k-nearest neighbor classifier. in *2020 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pp. 1–4. IEEE, 2020.

[22] Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[23] Huang, Gao, Chen, Danlu, Li, Tianhong, Wu, Felix, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Multi-scale dense convolutional networks for efficient prediction. *arXiv preprint arXiv:1703.09844*, 2, 2017.

[24] Zhang, Yin, Jin, Rong, and Zhou, Zhi-Hua. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[25] Ramos, Juan. Using tf-idf to determine word relevance in document queries.

[26] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[۲۷] منهاج، محمد باقر. هوش محاسباتی - جلد اول: مبانی شبکه‌های عصبی. دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)، ۱۳۹۳.

[28] Vasilev, Ivan, Slater, Daniel, Spacagna, Gianmario, Roelants, Peter, and Zocca, Valentino. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[29] Chollet, François. *Deep Learning with Python*. Manning, November 2017.

[30] Sibi, P, Jones, S Allwyn, and Siddarth, P. Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, 47(3):1264–1268, 2013.

[31] Nielsen, Michael A. *Neural networks and deep learning*, vol. 2018. Determination press San Francisco, CA, 2015.

[32] Medsker, Larry R and Jain, LC. Recurrent neural networks. *Design and Applications*, 5, 2001.

[33] Olah, Christopher. Understanding lstm networks. 2015.

[34] Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. in *ICML*, 2011.

[35] Olah, Christopher. Understanding lstm networks. 2015.

[36] Gu, Jiuxiang, Wang, Zhenhua, Kuen, Jason, Ma, Lianyang, Shahroudy, Amir, Shuai, Bing, Liu, Ting, Wang, Xingxing, Wang, Gang, Cai, Jianfei, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[37] Yin, Wenpeng, Schütze, Hinrich, Xiang, Bing, and Zhou, Bowen. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

[38] Yang, Zichao, Yang, Diyi, Dyer, Chris, He, Xiaodong, Smola, Alex, and Hovy, Eduard. Hierarchical attention networks for document classification. in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

[39] Long, Yunfei. Fake news detection through multi-perspective speaker profiles. Association for Computational Linguistics, 2017.

[40] Qian, Feng, Gong, Chengyue, Sharma, Karishma, and Liu, Yan. Neural user response generator: Fake news detection with collective user intelligence. in *IJCAI*, vol. 18, pp. 3834–3840, 2018.

[41] Shu, Kai, Cui, Limeng, Wang, Suhang, Lee, Dongwon, and Liu, Huan. defend: Explainable fake news detection. in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 395–405, 2019.

[42] Monti, Federico, Frasca, Fabrizio, Eynard, Davide, Mannion, Damon, and Bronstein, Michael M. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[43] Khattar, Dhruv, Goud, Jaipal Singh, Gupta, Manish, and Varma, Vasudeva. Mvae: Multimodal variational autoencoder for fake news detection. in *The world wide web conference*, pp. 2915–2921, 2019.

[44] Kaliyar, Rohit Kumar, Goswami, Anurag, Narang, Pratik, and Sinha, Soumendu. Fndnet–a deep convolutional neural network for fake news detection. *Cognitive Systems Research*, 61:32–44, 2020.

[45] Han, Yi, Karunasekera, Shanika, and Leckie, Christopher. Graph neural networks with continual learning for fake news detection from social media. *arXiv preprint arXiv:2007.03316*, 2020.

[46] Lu, Yi-Ju and Li, Cheng-Te. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media. *arXiv preprint arXiv:2004.11648*, 2020.

[47] Paka, William Scott, Bansal, Rachit, Kaushik, Abhay, Sengupta, Shubhashis, and Chakraborty, Tanmoy. Cross-sean: A cross-stitch semi-supervised neural attention model for covid-19 fake news detection. *Applied Soft Computing*, 107:107393, 2021.

[48] Zhang, Xueyao, Cao, Juan, Li, Xirong, Sheng, Qiang, Zhong, Lei, and Shu, Kai. Mining dual emotion for fake news detection. in *Proceedings of the Web Conference 2021*, pp. 3465–3476, 2021.

[49] Kaliyar, Rohit Kumar, Goswami, Anurag, and Narang, Pratik. Deepfake: improving fake news detection using tensor decomposition-based deep neural network. *The Journal of Supercomputing*, 77(2):1015–1037, 2021.

[50] Shah, Priyanshi and Kobti, Ziad. Multimodal fake news detection using a cultural algorithm with situational and normative knowledge. in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–7. IEEE, 2020.

[51] Shu, Kai, Wang, Suhang, and Liu, Huan. Beyond news contents: The role of social context for fake news detection. in *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 312–320, 2019.

پیوست آ

نحوه کار و کد شبکه عصبی تمایزگر رویداد برای تشخیص اخبار جعلی

برنامه آ-۱: کد اجرایی شبکه عصبی تمایزگر رویداد

```
EANN.py
--------------------
import numpy as np
import argparse
import time, os
# import random
from docutils.utils.roman import toRoman


import src.process_data_weibo2 as process_data
import copy
import pickle as pickle
from random import sample
import torchvision
from sklearn.model_selection import train_test_split
import torch
from torch.optim.lr_scheduler import StepLR, MultiStepLR, ExponentialLR
import torch.nn as nn
from torch.autograd import Variable, Function
```

```
from torch.utils.data import Dataset, DataLoader
import torch.nn.functional as F
from torch.nn.utils.rnn import pack_padded_sequence

import torchvision.datasets as dsets
import torchvision.transforms as transforms

#from logger import Logger

from sklearn import metrics
from sklearn.preprocessing import label_binarize
import scipy.io as sio

class Rumor_Data(Dataset):
    def __init__(self, dataset):
        self.text = torch.from_numpy(np.array(dataset['post_text']))
        self.image = list(dataset['image'])
        #self.social_context = torch.from_numpy(np.array(dataset['
            social_feature']))
        self.mask = torch.from_numpy(np.array(dataset['mask']))
        self.label = torch.from_numpy(np.array(dataset['label']))
        self.event_label = torch.from_numpy(np.array(dataset['
            event_label']))
        print('TEXT: %d, Image: %d, labe: %d, Event: %d'
              % (len(self.text), len(self.image), len(self.label), len
                  (self.event_label)))

    def __len__(self):
        return len(self.label)

    def __getitem__(self, idx):
        return (self.text[idx], self.image[idx], self.mask[idx]), self.
            label[idx], self.event_label[idx]
```

```
class ReverseLayerF(Function):

    @staticmethod
    def forward(self, x):
        # self.lambd = args.lambd
        return x.view_as(x)

    @staticmethod
    def backward(self, grad_output):
        return (grad_output * -args.lambd)

def grad_reverse(x):
    # return ReverseLayerF()(x)
    return ReverseLayerF.apply(x)



# Neural Network Model (1 hidden layer)
class CNN_Fusion(nn.Module):
    def __init__(self, args, W):
        super(CNN_Fusion, self).__init__()
        self.args = args

        self.event_num = args.event_num

        vocab_size = args.vocab_size
        emb_dim = args.embed_dim

        C = args.class_num
        self.hidden_size = args.hidden_dim
        self.lstm_size = args.embed_dim
```

```
self.social_size = 19

# TEXT RNN
self.embed = nn.Embedding(vocab_size, emb_dim)
self.embed.weight = nn.Parameter(torch.from_numpy(W))
self.lstm = nn.LSTM(self.lstm_size, self.lstm_size)
self.text_fc = nn.Linear(self.lstm_size, self.hidden_size)
self.text_encoder = nn.Linear(emb_dim, self.hidden_size)

### TEXT CNN
channel_in = 1
filter_num = 20
window_size = [1, 2, 3, 4]
self.convs = nn.ModuleList([nn.Conv2d(channel_in, filter_num, (
    K, emb_dim)) for K in window_size])
self.fc1 = nn.Linear(len(window_size) * filter_num, self.
    hidden_size)

self.dropout = nn.Dropout(args.dropout)

#IMAGE
#hidden_size = args.hidden_dim
vgg_19 = torchvision.models.vgg19(pretrained=True)
for param in vgg_19.parameters():
    param.requires_grad = False
# visual model
num_ftrs = vgg_19.classifier._modules['6'].out_features
self.vgg = vgg_19
self.image_fc1 = nn.Linear(num_ftrs,  self.hidden_size)
#self.image_fc2 = nn.Linear(512, self.hidden_size)
self.image_adv = nn.Linear(self.hidden_size,  int(self.
    hidden_size))
self.image_encoder = nn.Linear(self.hidden_size, self.
    hidden_size)
```

```
###social context
self.social = nn.Linear(self.social_size, self.hidden_size)

##ATTENTION
self.attention_layer = nn.Linear(self.hidden_size, emb_dim)

## Class  Classifier
self.class_classifier = nn.Sequential()
self.class_classifier.add_module('c_fc1', nn.Linear(2 * self.
    hidden_size, 2))
#self.class_classifier.add_module('c_bn1', nn.BatchNorm2d(100))
#self.class_classifier.add_module('c_relu1', nn.ReLU(True))
#self.class_classifier.add_module('c_drop1', nn.Dropout2d())
#self.class_classifier.add_module('c_fc2', nn.Linear(self.
    hidden_size, 2))
#self.class_classifier.add_module('c_bn2', nn.BatchNorm2d(self.
    hidden_size))
#self.class_classifier.add_module('c_relu2', nn.ReLU(True))
#self.class_classifier.add_module('c_fc3', nn.Linear(100, 10))
self.class_classifier.add_module('c_softmax', nn.Softmax(dim=1)
    )

###Event Classifier
self.domain_classifier = nn.Sequential()
self.domain_classifier.add_module('d_fc1', nn.Linear(2 * self.
    hidden_size, self.hidden_size))
#self.domain_classifier.add_module('d_bn1', nn.BatchNorm2d(self
    .hidden_size))
self.domain_classifier.add_module('d_relu1', nn.LeakyReLU(True)
    )
self.domain_classifier.add_module('d_fc2', nn.Linear(self.
    hidden_size, self.event_num))
```

```
        self.domain_classifier.add_module('d_softmax', nn.Softmax(dim
            =1))

    def init_hidden(self, batch_size):
        # Before we've done anything, we dont have any hidden state.
        # Refer to the Pytorch documentation to see exactly
        # why they have this dimensionality.
        # The axes semantics are (num_layers, minibatch_size,
            hidden_dim)
        return (to_var(torch.zeros(1, batch_size, self.lstm_size)),
                to_var(torch.zeros(1, batch_size, self.lstm_size)))

    def conv_and_pool(self, x, conv):
        x = F.relu(conv(x)).squeeze(3)  # (sample number,hidden_dim,
            length)
        #x = F.avg_pool1d(x, x.size(2)).squeeze(2)
        x = F.max_pool1d(x, x.size(2)).squeeze(2)

        return x

    def forward(self, text, image,  mask):
        ### IMAGE #####
        image = self.vgg(image) #[N, 512]
        image = F.leaky_relu(self.image_fc1(image))

        ##########CNN##################
        text = text.to(torch.int64)
        text = self.embed(text)
        text = text * mask.unsqueeze(2).expand_as(text)
        text = text.unsqueeze(1)
        text = [F.leaky_relu(conv(text)).squeeze(3) for conv in self.
            convs]  # [(N,hidden_dim,W), ...]*len(window_size)
        #text = [F.avg_pool1d(i, i.size(2)).squeeze(2) for i in text]
            # [(N,hidden_dim), ...]*len(window_size)
```

```
        text = [F.max_pool1d(i, i.size(2)).squeeze(2) for i in text]
        text = torch.cat(text, 1)
        text = F.leaky_relu(self.fc1(text))
        text_image = torch.cat((text, image), 1)

        ### Fake or real
        class_output = self.class_classifier(text_image)
        ## Domain (which Event )
        reverse_feature = grad_reverse(text_image)
        domain_output = self.domain_classifier(reverse_feature)

        # ### Multimodal
        # text_reverse_feature = grad_reverse(text)
        # image_reverse_feature = grad_reverse(image)
        # text_output = self.modal_classifier(text_reverse_feature)
        # image_output = self.modal_classifier(image_reverse_feature
        return class_output, domain_output

def to_var(x):
    if torch.cuda.is_available():
        x = x.cuda()
    return Variable(x)



def to_np(x):
    return x.data.cpu().numpy()

def select(train, selec_indices):
    temp = []
    for i in range(len(train)):
        print("length is "+str(len(train[i])))
        print(i)
        #print(train[i])
        ele = list(train[i])
```

```
        temp.append([ele[i] for i in selec_indices])
    return temp

def make_weights_for_balanced_classes(event, nclasses = 15):
    count = [0] * nclasses
    for item in event:
        count[item] += 1
    weight_per_class = [0.] * nclasses
    N = float(sum(count))
    for i in range(nclasses):
        weight_per_class[i] = N/float(count[i])
    weight = [0] * len(event)
    for idx, val in enumerate(event):
        weight[idx] = weight_per_class[val]
    return weight

def split_train_validation(train, percent):
    whole_len = len(train[0])

    train_indices = (sample(range(whole_len), int(whole_len * percent))
       )
    train_data = select(train, train_indices)
    print("train data size is "+ str(len(train[3])))
    # print()

    validation = select(train, np.delete(range(len(train[0])),
       train_indices))
    print("validation size is "+ str(len(validation[3])))
    print("train and validation data set has been splited")

    return train_data, validation


def main(args):
```

```
print('loading data')
train, validation, test, W = load_data(args)
# with open('weibo_df.pkl', 'rb') as f:
#     train, validation, test, W, word_index = pickle.load(f)

train_dataset = Rumor_Data(train)
validate_dataset = Rumor_Data(validation)
test_dataset = Rumor_Data(test)
# Data Loader (Input Pipeline)
train_loader = DataLoader(dataset=train_dataset,
                          batch_size=args.batch_size,
                          shuffle=True)
validate_loader = DataLoader(dataset = validate_dataset,
                             batch_size=args.batch_size,
                             shuffle=False)
test_loader = DataLoader(dataset=test_dataset,
                         batch_size=args.batch_size,
                         shuffle=False)

print('building model')
model = CNN_Fusion(args, W)

if torch.cuda.is_available():
    print("CUDA")
    model.cuda()

# Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(filter(lambda p: p.requires_grad, list
    (model.parameters())),
                             lr= args.learning_rate)
#optimizer = torch.optim.RMSprop(filter(lambda p: p.requires_grad,
    list(model.parameters())),
                             #lr=args.learning_rate)
```

```
#scheduler = StepLR(optimizer, step_size= 10, gamma= 1)



iter_per_epoch = len(train_loader)
print("loader size " + str(len(train_loader)))
best_validate_acc = 0.000
best_test_acc = 0.000
best_loss = 100
best_validate_dir = ''
best_list = [0,0]

print('training model')
adversarial = True
# Train the Model
for epoch in range(args.num_epochs):

    p = float(epoch) / 100
    #lambd = 2. / (1. + np.exp(-10. * p)) - 1
    lr = 0.001 / (1. + 10 * p) ** 0.75

    optimizer.lr = lr
    #rgs.lambd = lambd
    start_time = time.time()
    cost_vector = []
    class_cost_vector = []
    domain_cost_vector = []
    acc_vector = []
    valid_acc_vector = []
    test_acc_vector = []
    vali_cost_vector = []
    test_cost_vector = []

    for i, (train_data, train_labels, event_labels) in enumerate(
        train_loader):
```

```
train_text, train_image,  train_mask, train_labels,
    event_labels = \
     to_var(train_data[0]), to_var(train_data[1]), to_var(
         train_data[2]), \
     to_var(train_labels), to_var(event_labels)

# Forward + Backward + Optimize
optimizer.zero_grad()
train_labels = train_labels.to(torch.int64)
event_labels = event_labels.to(torch.int64)
class_outputs, domain_outputs = model(train_text,
    train_image, train_mask)
## Fake or Real loss
class_loss = criterion(class_outputs, train_labels)
# Event Loss
domain_loss = criterion(domain_outputs, event_labels)
loss = class_loss + domain_loss
loss.backward()
optimizer.step()
_, argmax = torch.max(class_outputs, 1)

cross_entropy = True

if True:
     accuracy = (train_labels == argmax.squeeze()).float().
         mean()
else:
     _, labels = torch.max(train_labels, 1)
     accuracy = (labels.squeeze() == argmax.squeeze()).float
         ().mean()

# class_cost_vector.append(class_loss.data[0])
class_cost_vector.append(class_loss.item())
domain_cost_vector.append(domain_loss.item())
```

```
        cost_vector.append(loss.item())
        acc_vector.append(accuracy.item())
        # if i == 0:
        #     train_score = to_np(class_outputs.squeeze())
        #     train_pred = to_np(argmax.squeeze())
        #     train_true = to_np(train_labels.squeeze())
        # else:
        #     class_score = np.concatenate((train_score, to_np(
            class_outputs.squeeze())), axis=0)
        #     train_pred = np.concatenate((train_pred, to_np(argmax
            .squeeze())), axis=0)
        #     train_true = np.concatenate((train_true, to_np(
            train_labels.squeeze())), axis=0)



    model.eval()
    validate_acc_vector_temp = []
    for i, (validate_data, validate_labels, event_labels) in
        enumerate(validate_loader):
        validate_text, validate_image,  validate_mask,
            validate_labels, event_labels = \
            to_var(validate_data[0]), to_var(validate_data[1]),
                to_var(validate_data[2]), \
            to_var(validate_labels), to_var(event_labels)

        validate_labels = validate_labels.to(torch.int64)
        validate_outputs, domain_outputs = model(validate_text,
            validate_image, validate_mask)
        _, validate_argmax = torch.max(validate_outputs, 1)
        vali_loss = criterion(validate_outputs, validate_labels)
        #domain_loss = criterion(domain_outputs, event_labels)
            #_, labels = torch.max(validate_labels, 1)
```

```
    validate_accuracy = (validate_labels == validate_argmax.
        squeeze()).float().mean()
    vali_cost_vector.append(vali_loss.item())
        #validate_accuracy = (validate_labels ==
            validate_argmax.squeeze()).float().mean()
    validate_acc_vector_temp.append(validate_accuracy.item())
validate_acc = np.mean(validate_acc_vector_temp)
valid_acc_vector.append(validate_acc)
model.train()
print ('Epoch [%d/%d],  Loss: %.4f, Class Loss: %.4f, domain
    loss: %.4f, Train_Acc: %.4f,  Validate_Acc: %.4f.'
        % (
        epoch + 1, args.num_epochs,  np.mean(cost_vector), np.
            mean(class_cost_vector),  np.mean(
            domain_cost_vector),
             np.mean(acc_vector),   validate_acc))

if validate_acc > best_validate_acc:
    best_validate_acc = validate_acc
    if not os.path.exists(args.output_file):
        os.mkdir(args.output_file)

    best_validate_dir = args.output_file + str(epoch + 1) + '.
        pkl'
    torch.save(model.state_dict(), best_validate_dir)

duration = time.time() - start_time
# print ('Epoch: %d, Mean_Cost: %.4f, Duration: %.4f,
    Mean_Train_Acc: %.4f, Mean_Test_Acc: %.4f'
# % (epoch + 1, np.mean(cost_vector), duration, np.mean(
    acc_vector), np.mean(test_acc_vector)))
# best_validate_dir = args.output_file + 'weibo_GPU2_out.' +
    str(52) + '.pkl'
```

```
# Test the Model
print('testing model')
model = CNN_Fusion(args, W)
model.load_state_dict(torch.load(best_validate_dir))
#    print(torch.cuda.is_available())
if torch.cuda.is_available():
    model.cuda()
model.eval()
test_score = []
test_pred = []
test_true = []
for i, (test_data, test_labels, event_labels) in enumerate(
   test_loader):
    test_text, test_image, test_mask, test_labels = to_var(
        test_data[0]), to_var(test_data[1]), to_var(test_data[2]),
           to_var(test_labels)
    test_outputs, domain_outputs= model(test_text, test_image,
       test_mask)
    _, test_argmax = torch.max(test_outputs, 1)
    if i == 0:
        test_score = to_np(test_outputs.squeeze())
        test_pred = to_np(test_argmax.squeeze())
        test_true = to_np(test_labels.squeeze())
    else:
        test_score = np.concatenate((test_score, to_np(test_outputs
           .squeeze())), axis=0)
        test_pred = np.concatenate((test_pred, to_np(test_argmax.
           squeeze())), axis=0)
        test_true = np.concatenate((test_true, to_np(test_labels.
           squeeze())), axis=0)

test_accuracy = metrics.accuracy_score(test_true, test_pred)
```

```
    test_f1 = metrics.f1_score(test_true, test_pred, average='macro')
    test_precision = metrics.precision_score(test_true, test_pred,
        average='macro')
    test_recall = metrics.recall_score(test_true, test_pred, average='
        macro')
    test_score_convert = [x[1] for x in test_score]
    test_aucroc = metrics.roc_auc_score(test_true, test_score_convert,
        average='macro')

    test_confusion_matrix = metrics.confusion_matrix(test_true,
        test_pred)

    print("Classification Acc: %.4f, AUC-ROC: %.4f"
          % (test_accuracy, test_aucroc))
    print("Classification report:\n%s\n"
          % (metrics.classification_report(test_true, test_pred)))
    print("Classification confusion matrix:\n%s\n"
          % (test_confusion_matrix))



def parse_arguments(parser):
    parser.add_argument('training_file', type=str, metavar='<
        training_file>', help='')
    #parser.add_argument('validation_file', type=str, metavar='<
        validation_file>', help='')
    parser.add_argument('testing_file', type=str, metavar='<
        testing_file>', help='')
    parser.add_argument('output_file', type=str, metavar='<output_file>
        ', help='')

    parse.add_argument('--static', type=bool, default=True, help='')
    parser.add_argument('--sequence_length', type=int, default=28, help
        ='')
```

```
    parser.add_argument('--class_num', type=int, default=2, help='')
    parser.add_argument('--hidden_dim', type=int, default = 32, help=''
        )
    parser.add_argument('--embed_dim', type=int, default=32, help='')
    parser.add_argument('--vocab_size', type=int, default=300, help='')
    parser.add_argument('--dropout', type=int, default=0.5, help='')
    parser.add_argument('--filter_num', type=int, default=5, help='')
    parser.add_argument('--lambd', type=int, default= 1, help='')
    parser.add_argument('--text_only', type=bool, default= False, help=
        '')

    #    parser.add_argument('--sequence_length', type = int, default =
         28, help = '')
    #    parser.add_argument('--input_size', type = int, default = 28,
        help = '')
    #    parser.add_argument('--hidden_size', type = int, default =
        128, help = '')
    #    parser.add_argument('--num_layers', type = int, default = 2,
        help = '')
    #    parser.add_argument('--num_classes', type = int, default = 10,
         help = '')
    parser.add_argument('--d_iter', type=int, default=3, help='')
    parser.add_argument('--batch_size', type=int, default=20, help='')
    parser.add_argument('--num_epochs', type=int, default=100, help='')
    parser.add_argument('--learning_rate', type=float, default=0.001,
        help='')
    parser.add_argument('--event_num', type=int, default=10, help='')

    #    args = parser.parse_args()
    return parser


def get_top_post(output, label, test_id, top_n = 500):
    filter_output = []
```

```
    filter_id = []
    #print(test_id)
    #print(output)
    for i, l in enumerate(label):
        #print(np.argmax(output[i]))
        if np.argmax(output[i]) == l and int(l) == 1 :
            filter_output.append(output[i][1])
            filter_id.append(test_id[i])

    filter_output = np.array(filter_output)

    top_n_indice = filter_output.argsort()[-top_n:][::-1]

    top_n_id = np.array(filter_id)[top_n_indice]
    top_n_id_dict = {}
    for i in top_n_id:
        top_n_id_dict[i] = True

    pickle.dump(top_n_id_dict, open("../Data/weibo/top_n_id.pickle", "
       wb"))

    return top_n_id


def word2vec(post, word_id_map, W):
    word_embedding = []
    mask = []
    #length = []

    for sentence in post:
        sen_embedding = []
        seq_len = len(sentence) -1
        mask_seq = np.zeros(args.sequence_len, dtype = np.float32)
        mask_seq[:len(sentence)] = 1.0
```

```
        for i, word in enumerate(sentence):
            sen_embedding.append(word_id_map[word])

        while len(sen_embedding) < args.sequence_len:
            sen_embedding.append(0)



        word_embedding.append(copy.deepcopy(sen_embedding))
        mask.append(copy.deepcopy(mask_seq))
        #length.append(seq_len)
    return word_embedding, mask


def tokenize_posts(df, word_id_map):
    n = len(df)
    tokenized_list = []
    mask_list = []
    for i in range(n):
        sentences = df.iloc[i]['sentences']
        tokenized_sentences, mask = word2vec(sentences, word_id_map,
            None)
        tokenized_list.append(tokenized_sentences)
        mask_list.append(mask)

    df['tokenized_sentences'] = tokenized_list
    df['mask'] = mask_list
    return df

def load_data(args):
    train, validate, test, dataFrame_train, dataFrame_val,
        dataFrame_test = process_data.get_data(args.text_only)
    #print(train[4][0])
    word_vector_path = '../Data/weibo/word_embedding.pickle'
    f = open(word_vector_path, 'rb')
```

```
weight = pickle.load(f)  # W, W2, word_idx_map, vocab
W, W2, word_idx_map, vocab, max_len = weight[0], weight[1], weight
    [2], weight[3], weight[4]
args.vocab_size = len(vocab)
args.sequence_len = max_len
print("translate data to embedding")

word_embedding, mask = word2vec(validate['post_text'], word_idx_map
    , W)
validate['post_text'] = word_embedding
validate['mask'] = mask



print("translate test data to embedding")
word_embedding, mask = word2vec(test['post_text'], word_idx_map, W)
test['post_text'] = word_embedding
test['mask']=mask
#test[-2]= transform(test[-2])
word_embedding, mask = word2vec(train['post_text'], word_idx_map, W
    )
train['post_text'] = word_embedding
train['mask'] = mask
print("sequence length " + str(args.sequence_length))
print("Train Data Size is "+str(len(train['post_text'])))
print("Finished loading data ")

dataFrame_train = tokenize_posts(dataFrame_train, word_idx_map)
dataFrame_test = tokenize_posts(dataFrame_test, word_idx_map)
dataFrame_val = tokenize_posts(dataFrame_val, word_idx_map)

with open('weibo_df.pkl', 'wb') as f:
    pickle.dump([dataFrame_train, dataFrame_test, dataFrame_val, W,
        word_idx_map], f)
```

```
    return train, validate, test, W

def transform(event):
    matrix = np.zeros([len(event), max(event) + 1])
    #print("Translate  shape is " + str(matrix))
    for i, l in enumerate(event):
        matrix[i, l] = 1.00
    return matrix



if __name__ == '__main__':
    parse = argparse.ArgumentParser()
    parser = parse_arguments(parse)
    train = ''
    test = ''
    output = '../Data/weibo/RESULT/'
    args = parser.parse_args([train, test, output])

    main(args)
```

---

process_data_weibo2.py

```
# encoding=utf-8
import pickle as pickle
import random
from random import *
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import os
from collections import defaultdict
import sys, re
```

```
import pandas as pd
from PIL import Image
import math
from types import *
from gensim.models import Word2Vec
import jieba
from sklearn.cluster import AgglomerativeClustering
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
import os.path
from gensim.models import Word2Vec

# def stopwordslist(filepath = '/content/gdrive/My Drive/Data/weibo/
    stop_words.txt'):
def stopwordslist(filepath = '../Data/weibo/stop_words.txt'):
    # stopwords = {}
    # for line in open(filepath, 'r').readlines():
    #     # line = unicode(line, "utf-8").strip()
    #     line = str(line).strip()
    #     stopwords[line] = 1
    stopwords = [line.strip() for line in open(filepath, 'r', encoding=
        'utf-8').readlines()]
    return stopwords

def clean_str_sst(string):
    """
    Tokenization/string cleaning for the SST dataset
    """
    string = re.sub(u" [ ""--:,.|-_/nbsp+&;   @())#  0]", "", string)
    return string.strip().lower()



# import sys
```

```
# reload(sys)
# sys.setdefaultencoding("utf-8")
#
def read_image():
    image_list = {}
    file_list = ['../Data/weibo/nonrumor_images/', '../Data/weibo/
       rumor_images/']
    # file_list = ['/content/gdrive/My Drive/Data/weibo/nonrumor_images
       /', '/content/gdrive/My Drive/Data/weibo/rumor_images/']
    for path in file_list:
        data_transforms = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
               0.225])
            ])

        for i, filename in enumerate(os.listdir(path)):  # assuming gif

            # print(filename)
            try:
                im = Image.open(path + filename).convert('RGB')
                im = data_transforms(im)
                #im = 1
                image_list[filename.split('/')[-1].split(".")[0].lower
                   ()] = im
            except:
                print(filename)
    print("image length " + str(len(image_list)))
    #print("image names are " + str(image_list.keys()))
    return image_list

def write_txt(data):
```

```
    f = open("../Data/weibo/top_n_data.txt", 'wb')
    # f = open("/content/gdrive/My Drive/Data/weibo/top_n_data.txt", '
        wb')
    for line in data:
        for l in line:
            f.write(l+"\n")
        f.write("\n")
        f.write("\n")
    f.close()
text_dict = {}
def write_data(flag, image, text_only):

    def read_post(flag):
        stop_words = stopwordslist()
        # pre_path = "/content/gdrive/My Drive/Data/weibo/tweets/"
        pre_path = "../Data/weibo/tweets/"
        file_list = [pre_path + "test_nonrumor.txt", pre_path + "
            test_rumor.txt", \
                        pre_path + "train_nonrumor.txt", pre_path + "
                            train_rumor.txt"]
        # if flag == "train":
        #     id = pickle.load(open("/content/gdrive/My Drive/Data/
            weibo/train_id.pickle", 'rb'))
        # elif flag == "validate":
        #     id = pickle.load(open("/content/gdrive/My Drive/Data/
            weibo/validate_id.pickle", 'rb'))
        # elif flag == "test":
        #     id = pickle.load(open("/content/gdrive/My Drive/Data/
            weibo/test_id.pickle", 'rb'))
        if flag == "train":
            id = pickle.load(open("../Data/weibo/train_id.pickle", 'rb'
                ))
        elif flag == "validate":
```

```
    id = pickle.load(open("../Data/weibo/validate_id.pickle", '
        rb'))
elif flag == "test":
    id = pickle.load(open("../Data/weibo/test_id.pickle", 'rb')
        )


post_content = []
labels = []
image_ids = []
twitter_ids = []
data = []
column = ['post_id', 'image_id', 'original_post', 'post_text',
    'label', 'event_label', 'sentences']
key = -1
map_id = {}
top_data = []
for k, f in enumerate(file_list):
    # f = open(f, 'rb')
    f = open(f, 'r', encoding='utf-8')
    if (k + 1) % 2 == 1:
        label = 0  ### real is 0
    else:
        label = 1  ####fake is 1

    twitter_id = 0
    line_data = []
    top_line_data = []

    for i, l in enumerate(f.readlines()):
        # key += 1
        # if int(key /3) in index:
        # print(key/3)
        # continue
        if (i + 1) % 3 == 1:
```

```
        line_data = []
        twitter_id = l.split('|')[0]
        line_data.append(twitter_id)

    if (i + 1) % 3 == 2:

        line_data.append(l.lower())

    if (i + 1) % 3 == 0:
        # l = clean_str_sst(unicode(l, "utf-8"))
        sentences = clean_sentences(l.split(' '),
            stop_words)
        l = clean_str_sst(str(l))
        seg_list = jieba.cut_for_search(l)
        new_seg_list = []
        for word in seg_list:
            if word not in stop_words:
                new_seg_list.append(word)

        clean_l = " ".join(new_seg_list)
        if len(clean_l) > 10 and line_data[0] in id:
            line_data.append(l)
            line_data.append(clean_l)
            line_data.append(label)
            post_content.append(sentences)
            event = int(id[line_data[0]])
            if event not in map_id:
                map_id[event] = len(map_id)
                event = map_id[event]
            else:
                event = map_id[event]

            line_data.append(event)
            line_data.append(sentences)
```

```
                        data.append(line_data)

            f.close()
            # print(data)
            #     return post_content




        data_df = pd.DataFrame(np.array(data), columns=column)
        write_txt(top_data)

        return post_content, data_df

    post_content, post = read_post(flag)
    print("Original post length is " + str(len(post_content)))
    print("Original data frame is " + str(post.shape))


    def find_most(db):
        maxcount = max(len(v) for v in db.values())
        return [k for k, v in db.items() if len(v) == maxcount]

    def select(train, selec_indices):
        temp = []
        for i in range(len(train)):
            ele = list(train[i])
            temp.append([ele[i] for i in selec_indices])
            #   temp.append(np.array(train[i])[selec_indices])
        return temp

#     def balance_event(data, event_list):
#         id = find_most(event_list)[0]
#         remove_indice = random.sample(range(min(event_list[id]), \
```

```
#                                                   max(event_list[id])), int
    (len(event_list[id]) * 0.9))
#         select_indices = np.delete(range(len(data[0])), remove_indice
    )
#         return select(data, select_indices)

    def paired(text_only = False):
        ordered_image = []
        ordered_text = []
        ordered_post = []
        ordered_event= []
        label = []
        post_id = []
        image_id_list = []
        #image = []

        image_id = ""
        for i, id in enumerate(post['post_id']):
            for image_id in post.iloc[i]['image_id'].split('|'):
                image_id = image_id.split("/")[-1].split(".")[0]
                if image_id in image:
                    break

            if text_only or image_id in image:
                if not text_only:
                    image_name = image_id
                    image_id_list.append(image_name)
                    ordered_image.append(image[image_name])
                ordered_text.append(post.iloc[i]['original_post'])
                ordered_post.append(post.iloc[i]['post_text'])
                ordered_event.append(post.iloc[i]['event_label'])
                post_id.append(id)
```

```
        label.append(post.iloc[i]['label'])

label = np.array(label, dtype=np.int)
ordered_event = np.array(ordered_event, dtype=np.int)
post['main_image_name'] = image_id_list


print("Label number is " + str(len(label)))
print("Rummor number is " + str(sum(label)))
print("Non rummor is " + str(len(label) - sum(label)))




#
if flag == "test":
    y = np.zeros(len(ordered_post))
else:
    y = []


data = {"post_text": np.array(ordered_post),
        "original_post": np.array(ordered_text),
        "image": ordered_image, "social_feature": [],
        "label": np.array(label), \
        "event_label": ordered_event, "post_id":np.array(
            post_id),
        "image_id":image_id_list}
#print(data['image'][0])


print("data size is " + str(len(data["post_text"])))

return data, post
```

```
    paired_data, post = paired(text_only)

    print("paired post length is "+str(len(paired_data["post_text"])))
    print("paried data has " + str(len(paired_data)) + " dimension")
    return paired_data, post


def clean_sentences(sentences, stop_words):
    clean_sentences = []
    for s in sentences:
        s = clean_str_sst(str(s))
        seg_list = jieba.cut_for_search(s)
        new_seg_list = []
        for word in seg_list:
            if word not in stop_words:
                new_seg_list.append(word)

        clean_s = " ".join(new_seg_list)
        clean_sentences.append(clean_s)

    return clean_sentences

def load_data(train, validate, test):
    vocab = defaultdict(float)
    all_text = list(train['post_text']) + list(validate['post_text'])+
       list(test['post_text'])
    for sentence in all_text:
        for word in sentence:
            vocab[word] += 1
    return vocab, all_text
```

```
def build_data_cv(data_folder, cv=10, clean_string=True):
    """
    Loads data and split into 10 folds.
    """
    revs = []
    pos_file = data_folder[0]
    neg_file = data_folder[1]
    vocab = defaultdict(float)
    with open(pos_file, "rb") as f:
        for line in f:
            rev = []
            rev.append(line.strip())
            if clean_string:
                orig_rev = clean_str(" ".join(rev))
            else:
                orig_rev = " ".join(rev).lower()
            words = set(orig_rev.split())
            for word in words:
                vocab[word] += 1
            datum = {"y": 1,
                     "text": orig_rev,
                     "num_words": len(orig_rev.split()),
                     "split": np.random.randint(0, cv)}
            revs.append(datum)
    with open(neg_file, "rb") as f:
        for line in f:
            rev = []
            rev.append(line.strip())
            if clean_string:
                orig_rev = clean_str(" ".join(rev))
            else:
                orig_rev = " ".join(rev).lower()
            words = set(orig_rev.split())
```

```
            for word in words:
                vocab[word] += 1
            datum = {"y": 0,
                     "text": orig_rev,
                     "num_words": len(orig_rev.split()),
                     "split": np.random.randint(0, cv)}
            revs.append(datum)
    return revs, vocab


def get_W(word_vecs, k=32):
    """
    Get word matrix. W[i] is the vector for word indexed by i
    """
    # vocab_size = len(word_vecs)
    word_idx_map = dict()
    W = np.zeros(shape=(len(word_vecs) + 1, k), dtype='float32')
    W[0] = np.zeros(k, dtype='float32')
    i = 1
    for word in word_vecs:
        W[i] = word_vecs[word]
        word_idx_map[word] = i
        i += 1
    return W, word_idx_map


def load_bin_vec(fname, vocab):
    """
    Loads 300x1 word vecs from Google (Mikolov) word2vec
    """
    word_vecs = {}
    with open(fname, "rb") as f:
        header = f.readline()
        vocab_size, layer1_size = map(int, header.split())
```

```
        binary_len = np.dtype('float32').itemsize * layer1_size
        for line in xrange(vocab_size):
            word = []
            while True:
                ch = f.read(1)
                if ch == ' ':
                    word = ''.join(word)
                    break
                if ch != '\n':
                    word.append(ch)
            if word in vocab:
                word_vecs[word] = np.fromstring(f.read(binary_len),
                    dtype='float32')
            else:
                f.read(binary_len)
    return word_vecs


def add_unknown_words(word_vecs, vocab, min_df=1, k=32):
    """
    For words that occur in at least min_df documents, create a
        separate word vector.
    0.25 is chosen so the unknown vectors have (approximately) same
        variance as pre-trained ones
    """
    for word in vocab:
        if word not in word_vecs and vocab[word] >= min_df:
            word_vecs[word] = np.random.uniform(-0.25, 0.25, k)



def get_data(text_only):
    #text_only = False
```

```
if text_only:
    print("Text only")
    image_list = []
else:
    print("Text and image")
    image_list = read_image()

train_data, dataFrame_train = write_data("train", image_list,
    text_only)
valiate_data, dataFrame_val = write_data("validate", image_list,
    text_only)
test_data, dataFrame_test = write_data("test", image_list,
    text_only)

print("loading data...")
# w2v_file = '/content/gdrive/My Drive/Data/GoogleNews-vectors-
    negative300.bin'
vocab, all_text = load_data(train_data, valiate_data, test_data)
# print(str(len(all_text)))

print("number of sentences: " + str(len(all_text)))
print("vocab size: " + str(len(vocab)))
max_l = len(max(all_text, key=len))
print("max sentence length: " + str(max_l))

#
#
# word_embedding_path = "/content/gdrive/My Drive/Data/weibo/w2v.
    pickle"
word_embedding_path = "../Data/weibo/w2v.pickle"

# w2v = pickle.load(open(word_embedding_path, 'rb'))
w2v = pickle.load(open(word_embedding_path, 'rb'), encoding='latin
    -1')
```

```
    # print(temp)
    print("word2vec loaded!")
    print("num words already in word2vec: " + str(len(w2v)))
    print("word2vec loaded!")
    print("num words already in word2vec: " + str(len(w2v)))
    add_unknown_words(w2v, vocab)
    W, word_idx_map = get_W(w2v)
    # # rand_vecs = {}
    # # add_unknown_words(rand_vecs, vocab)
    W2 = rand_vecs = {}
    w_file = open("../Data/weibo/word_embedding.pickle", "wb")
    # w_file = open("/content/gdrive/My Drive/Data/weibo/word_embedding
        .pickle", "wb")
    pickle.dump([W, W2, word_idx_map, vocab, max_l], w_file)
    w_file.close()
    return train_data, valiate_data, test_data, dataFrame_train,
        dataFrame_val, dataFrame_test
```

پیوست ب

نحوه کار و کد ساختار سلسله مراتبی سطح گفتمان برای تشخیص اخبار جعلی

برنامه ب-۱: کد اجرایی ساختار سلسله مراتبی سطح گفتمان

```
model.py
--------------------
import numpy as np
import torch
import torch.nn as nn
import config
import utils
args = config.args
class DependencyBLSTM(nn.Module):
    def __init__(self, num_words, max_sen_length, max_doc_sent_length):
        super(DependencyBLSTM, self).__init__()
        self.max_sen_length = max_sen_length
        self.max_doc_sent_length = max_doc_sent_length
        self.dropout = nn.Dropout(p=args.dropout)
        self.word_embedding = nn.Embedding(num_embeddings=num_words,
            embedding_dim=args.word_dim)
        self.Softmax = nn.Softmax(dim=0)
```

```
############################## Sentence level Functions
    ########################################

self.forwardLSTM_sent = nn.LSTM(num_layers=1, input_size=args.
    word_dim,
                                dropout=args.dropout,
                                hidden_size=int(args.
                                   blstm_hidden_unit_dim),
                                batch_first=True)
self.backwardLSTM_sent = nn.LSTM(num_layers=1, input_size=args.
    word_dim,
                                 dropout=args.dropout,
                                 hidden_size=args.
                                    blstm_hidden_unit_dim,
                                 batch_first=True)
self.sentence_encoder = nn.Sequential(nn.Linear(args.word_dim,
    args.blstm_hidden_unit_dim),
                                      nn.LeakyReLU(), nn.
                                          Dropout(p=args.
                                          dropout))
############################## Doc level Functions
    ########################################


self.parent_encoder_doc = nn.Sequential(
    nn.Linear(args.blstm_hidden_unit_dim, int(args.
        blstm_hidden_unit_dim)),
    nn.LeakyReLU(), nn.Dropout(p=args.dropout))
self.child_encoder_doc = nn.Sequential(
    nn.Linear(args.blstm_hidden_unit_dim, int(args.
        blstm_hidden_unit_dim)),
    nn.LeakyReLU(), nn.Dropout(p=args.dropout))
```

```
        self.root_score_encoder_doc = nn.Linear(args.
            blstm_hidden_unit_dim, 1)
        self.root_embed_doc = \
            utils.wrap_with_variable(torch.FloatTensor(np.zeros(shape=(
                args.blstm_hidden_unit_dim))),
            gpu=args.gpu,
            requires_grad=True)
        self.r_embeds_doc = nn.Sequential(
            nn.Linear(3 * args.blstm_hidden_unit_dim,
                      int(args.blstm_hidden_unit_dim)),
            nn.LeakyReLU(), nn.Dropout(p=args.dropout))

        self.final_binary_classifier = nn.Linear(int(args.
            blstm_hidden_unit_dim), 2)

    def create_sentence_batches(self, docs):
        all_doc_batches = []
        all_doc_batches_inverse = []
        all_doc_seq_lengths = []
        for doc in docs:
            doc_sent_embed, doc_sent_embed_inverse = [], []
            seq_lengths = []
            for sent_word_indices in doc['word_indices']:
                j = utils.wrap_with_variable(torch.LongTensor(np.array(
                    sent_word_indices).astype(int)), gpu=args.gpu,
                    requires_grad=False)

                word_embed = self.word_embedding(j)
                word_embed = self.dropout(word_embed)
                X = torch.zeros(self.max_sen_length, args.word_dim)
                X[0:len(sent_word_indices)] = word_embed.data
                X = utils.wrap_with_variable(X, gpu=args.gpu,
                    requires_grad=True)
                doc_sent_embed.append(X)
```

```
            idx = [i for i in range(word_embed.data.size(0) - 1,
                -1, -1)]
            if args.gpu > -1:
                idx = torch.LongTensor(idx).cuda(args.gpu)
            else:
                idx = torch.LongTensor(idx)
        X_inverse = torch.zeros(self.max_sen_length, args.word_dim)
        X_inverse[0:len(sent_word_indices)] = word_embed.data.
            index_select(0, idx)
        X_inverse = utils.wrap_with_variable(X_inverse, gpu=args.gpu
            , requires_grad=True)
            doc_sent_embed_inverse.append(X_inverse)

            seq_lengths.append(len(sent_word_indices))

        doc_sent_embed = torch.stack(doc_sent_embed)
        doc_sent_embed_inverse = torch.stack(doc_sent_embed_inverse
            )
        all_doc_batches.append(doc_sent_embed)
        all_doc_batches_inverse.append(doc_sent_embed_inverse)
        all_doc_seq_lengths.append(seq_lengths)
    return all_doc_batches, all_doc_batches_inverse,
        all_doc_seq_lengths

def get_sentence_encodings(self, all_doc_batches,
    all_doc_batches_inverse, all_doc_seq_lengths):
    all_doc_sentence_encodings = []
    for doc_batch, doc_batch_inverse, doc_seq_length in zip(
        all_doc_batches, all_doc_batches_inverse,
        all_doc_seq_lengths):
        doc_sentence_encodings = []

        fwrd_outputs, _ = self.forwardLSTM_sent(doc_batch)
```

```
            bwrd_outputs, _ = self.backwardLSTM_sent(doc_batch_inverse)
            for sent_forward, sent_backward, l in zip(doc_batch,
                doc_batch_inverse, doc_seq_length):
                idx = [i for i in range(l - 1, -1, -1)]
                if args.gpu > -1:
                    idx = torch.LongTensor(idx).cuda(args.gpu)
                else:
                    idx = torch.LongTensor(idx)
                bwrd_outputs_inverse = utils.wrap_with_variable(
                    sent_backward.data.index_select(0, idx), gpu=args.
                    gpu,
                 requires_grad=True)

                h = self.sentence_encoder(0.5 * (sent_forward[l - 1] +
                    bwrd_outputs_inverse[l-1]))
                doc_sentence_encodings.append(h)
            doc_sentence_encodings = torch.stack(doc_sentence_encodings
                )
            all_doc_sentence_encodings.append(doc_sentence_encodings)
        return all_doc_sentence_encodings


    def forward(self, docs):
 all_doc_batches, all_doc_batches_inverse,
all_doc_seq_lengths = self.create_sentence_batches(docs)
all_doc_sentence_encodings = self.get_sentence_encodings(
    all_doc_batches,
all_doc_batches_invers
                    all_doc_seq_lengths)
        all_doc_doc_dependency_tree_info = []
        all_final_features = []
        for sentence_encodings in all_doc_sentence_encodings:
            fri = []
            Aij = []
            for i in range(len(sentence_encodings)):
```

```
        fri.append(self.root_score_encoder_doc(
            sentence_encodings[i]))
        for j in range(len(sentence_encodings)):
            if i == j:
             Aij.append(utils.wrap_with_variable(torch.tensor
                (-9999999.000), gpu=args.gpu, requires_grad=True))
                    continue

            x = torch.dot(self.parent_encoder_doc(
                sentence_encodings[i]),
                            self.child_encoder_doc(
                                sentence_encodings[j]))
            Aij.append(x)
    Aij = torch.stack(Aij)
    Aij = Aij.view(len(sentence_encodings), len(
        sentence_encodings))
    Aij = self.Softmax(Aij)
    fri = torch.stack(fri)
    fri = self.Softmax(fri)
    ri = []
    for i in range(len(sentence_encodings)):
        tmp = []
        tmp3 = []
        for k in range(len(sentence_encodings)):
            tmp.append(torch.mul(Aij[k, i], sentence_encodings[
                k]))
            tmp3.append(torch.mul(Aij[i, k], sentence_encodings
                [i]))
        tmp3 = torch.stack(tmp3)
        tmp3 = torch.sum(tmp3, 0)
        tmp = torch.stack(tmp)
        tmp = torch.sum(tmp, 0)
        tmp2 = torch.mul(fri[i], self.root_embed_doc)
```

```
                ri.append(self.r_embeds_doc(torch.cat((
                    sentence_encodings[i], tmp + tmp2, tmp3))))
            ri = torch.stack(ri)
            final_feature = torch.mean(ri,0)
            all_final_features.append(final_feature)
            all_doc_doc_dependency_tree_info.append([Aij.data, fri.data
                ])

        all_final_features = torch.stack(all_final_features)
        output = self.final_binary_classifier(all_final_features)
        return output, all_doc_doc_dependency_tree_info
```

---

train.py

```
import os
import random
import numpy as np

import torch

from sklearn.metrics import accuracy_score
# confusion_matrix
# classification_report
from sklearn.metrics import confusion_matrix, classification_report
from torch import nn

from torch.autograd import Variable

from torch.optim.lr_scheduler import StepLR

#config :
import config
import utils
```

```
# DependencyBLSTM : Long Short Term Memory

#http://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1

from model import DependencyBLSTM

args = config.args

output_dir = 'Models/' + args.sim_name + '/'

utils.creat_word_embedding()

def run():
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    save_path = 'Models/' + args.sim_name + '/model.pt'

    with open(output_dir + 'config', 'w') as config_file:
        argss = (str(args).split('(')[1].split(')')[0].split(','))
        for a in argss:
            config_file.write("{}\n".format(a))
    if os.path.exists(save_path):
        model = torch.load(save_path)
        model_loaded = True
        print('Great!!! Pre-Trained Model Loaded !!!')
    else:
        model_loaded = False
        print('No pre-trained model ')
        model = DependencyBLSTM(num_words=utils.get_num_words(),
            max_sen_length=97, max_doc_sent_length=326)

    if not os.path.exists(output_dir + 'train_performance_log.csv'):
```

```
    train_performance_log = open(output_dir + '
        train_performance_log.csv', 'w')
    train_performance_log.write('Step,Loss\n')
else:
    train_performance_log = open(output_dir + '
        train_performance_log.csv', 'a')

if not os.path.exists(output_dir + 'eval_performance_log.txt'):
    eval_performance_log = open(output_dir + 'eval_performance_log.
        txt', 'w')
else:
    eval_performance_log = open(output_dir + 'eval_performance_log.
        txt', 'a')

if args.gpu > -1:
    model.cuda(device=int(args.gpu))

if not model_loaded:
    if args.fill_embedding:
        embed = utils.get_word_embeddings(source='google')
        if args.gpu > -1:
            model.word_embedding.weight.data.set_(torch.FloatTensor
                ((embed)).cuda(int(args.gpu)))
        else:
            with torch.no_grad():
                model.word_embedding.weight.set_(torch.FloatTensor((
                    embed)))
    else:
        if args.gpu > -1:
            model.word_embedding.weight.data.set_(
                torch.FloatTensor((np.zeros((utils.get_num_words()
                    + 1, args.word_dim)).astype()))).cuda(
                    int(args.gpu)))
        else:
```

```
            model.word_embedding.weight.data.set_(
                torch.FloatTensor((np.zeros((utils.get_num_words()
                    + 1, args.word_dim))).astype(float)))

if args.train_embeddings == False:
    model.word_embedding.weight.requires_grad = False

params = [p for p in model.parameters() if p.requires_grad]
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(params=params, lr=args.lr,
   weight_decay=args.l2_coeff)
model.zero_grad()
scheduler = StepLR(optimizer, step_size=50, gamma=0.9)
print('Loading sets...')
dev_set = utils.get_split_data(split='dev')
train_set = utils.get_split_data(split='train')
train_set =train_set[0:int(len(train_set)/2)]
print('Train and dev sets loaded')

def train():
    prev_accuracy = 0
    model.train()
    for step in range(args.step_num + 1):
        random.shuffle(train_set)
        docs = train_set[0:args.batch_size]
        labels = Variable(torch.LongTensor([d['label'] for d in
            docs]))
        doc_encodings, _ = model(docs)
        optimizer.zero_grad()
        outputs = doc_encodings.cpu()
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        scheduler.step()
```

```
        print("Step {} Loss {}".format(step, loss.item()))
        train_performance_log.write("{},{}\n".format(step, loss.
            item()))
        if step % 20 == 0 and step:
            accuracy = evaluation(step)
            if accuracy >= prev_accuracy:
                torch.save(model, save_path)
                print("Best model saved in {} Accuracy {}".format(
                    save_path, accuracy))
                prev_accuracy = accuracy

def evaluation(step):
    print('Start evaluation ...')
    model.eval()
    eval_labels = [d['label'] for d in dev_set]
    labels = Variable(torch.LongTensor(eval_labels))
    doc_encodings, _ = model(dev_set)
    outputs = doc_encodings.cpu()
    loss = criterion(outputs, labels).item()
    _, predictions = torch.max(outputs.data, 1)
    predictions = predictions.numpy()

    accuracy = accuracy_score(y_true=np.array(eval_labels), y_pred=
        predictions)
    report = classification_report(y_true=np.array(eval_labels),
        y_pred=predictions, target_names=['Real', 'Fake'])
    conf_matrix = confusion_matrix(y_true=np.array(eval_labels),
        y_pred=predictions)
    eval_performance_log.write("Step {}, Loss {} Accuracy {} \n".
        format(step, loss, accuracy))
    eval_performance_log.write("{}\n".format(report))
    eval_performance_log.write("{}\n".format(conf_matrix))
    eval_performance_log.write("{}\n".format('=' * 50))
```

```
        print('************* Evaluation ***************')
        print("Step {}, Loss {}  Accuracy {} ".format(step, loss,
            accuracy))
        print(report)
        print(conf_matrix)
        print('****************************************')
        return accuracy

    train()

run()
```

---

test.py

```
import os
import pickle
import numpy as np
import torch
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from torch import nn
from torch.autograd import Variable
import config
import utils
from model import DependencyBLSTM

args = config.args
output_dir ='Models/' + args.sim_name + '/'
utils.creat_word_embedding()

def test():
    if not os.path.exists(output_dir):
```

```
    os.makedirs(output_dir)
save_path = 'Models/' + args.sim_name + '/model.pt'

if os.path.exists(save_path):
    model = torch.load(save_path)
    print('Great!!! Pre-Trained Model Loaded !!!')
else:
    print('No pre-trained model ')
    model = DependencyBLSTM(num_words=utils.get_num_words(),
        max_sen_length=97, max_doc_sent_length=326)

if not os.path.exists(output_dir + 'test_performance_log.txt'):
    test_performance_log = open(output_dir + 'test_performance_log.
        txt', 'w')
else:
    test_performance_log = open(output_dir + 'test_performance_log.
        txt', 'a')

if args.gpu > -1:
    model.cuda(device=int(args.gpu))
if args.fill_embedding :
    embed = utils.get_word_embeddings(source='google')
    if args.gpu > -1:
        with torch.no_grad():
            model.word_embedding.weight.set_(torch.FloatTensor((
                embed)).cuda(int(args.gpu)))
    else:
        with torch.no_grad():
            model.word_embedding.weight.set_(torch.FloatTensor((
                embed)))
else:
    if args.gpu > -1:
        model.word_embedding.weight.data.set_(
```

```
            torch.FloatTensor((np.zeros((utils.get_num_words() + 1,
                args.word_dim)).astype())).cuda(int(args.gpu)))
    else:
        model.word_embedding.weight.data.set_(
            torch.FloatTensor((np.zeros((utils.get_num_words() + 1,
                args.word_dim))).astype(float)))

if args.train_embeddings == False:
    model.word_embedding.weight.requires_grad = False

criterion = nn.CrossEntropyLoss()
test_set = utils.get_split_data(split='test')
print('Start Test ...')
model.eval()
test_labels = [d['label'] for d in test_set]
labels = Variable(torch.LongTensor(test_labels))

lengths = [len(doc['word_indices']) for doc in test_set]
doc_encodings, all_doc_doc_dependency_tree_info = model(test_set)

outputs = doc_encodings.cpu()
loss = criterion(outputs, labels).item()
_, predictions = torch.max(outputs.data, 1)
predictions = predictions.numpy()

with open(output_dir + "matrix.pkl", 'wb') as f:
    pickle.dump([lengths, all_doc_doc_dependency_tree_info,
        test_labels], f)

accuracy = accuracy_score(y_true=np.array(test_labels), y_pred=
    predictions)
report = classification_report(y_true=np.array(test_labels), y_pred
    =predictions, target_names=['Real', 'Fake'])
```

```
    conf_matrix = confusion_matrix(y_true=np.array(test_labels), y_pred
        =predictions)
    test_performance_log.write(" Loss {} Accuracy {} \n".format(loss,
        accuracy))
    test_performance_log.write("{}\n".format(report))
    test_performance_log.write("{}\n".format(conf_matrix))
    test_performance_log.write("{}\n".format('=' * 50))

    print('************* Test ***************')
    print("Loss {}  Accuracy {} ".format(loss, accuracy))
    print(report)
    print(conf_matrix)
    print('****************************************')
    return accuracy


test()

fake_doc_stat, real_doc_stat = utils.dependecy_tree_stat(dir=output_dir
    )
#print(fake_doc_stat)
#print('*' * 100)
#print(real_doc_stat)
#print('*' * 100)
print("Avg. Number of Leaf Nodes: Fake {} Real {}".format(np.mean(
    fake_doc_stat[:, 2]), np.mean(real_doc_stat[:, 2])))
print("Avg. Preorder Difference: Fake {} Real {}".format(np.mean(
    fake_doc_stat[:, 4]), np.mean(real_doc_stat[:, 4])))
print("Avg. Parent-Child Distance: Fake {} Real {}".format(np.mean(
    fake_doc_stat[:, 3]), np.mean(real_doc_stat[:, 3])))
```

---

utils.py

```
import pickle
import numpy as np
import pandas as pd
from torch.autograd import Variable
import config
import gensim.models.keyedvectors as word2vec
import os
import urllib.request
args = config.args



###################### Tree
    ##################################################
class myTree(object):
    def __init__(self, name='Node', children=None, data=None, parent=
        None):
        self.parent = parent
        self.name = name
        self.index = -1
        self.children = []
        self.data = data
        self.characters = []
        self.parent_relation_index = -1
        if children is not None:
            for child in children:
                self.add_child(child)

    def __repr__(self):
        return self.name

    def add_child(self, node):
        assert isinstance(node, myTree)
        self.children.append(node)
```

```
    def __str__(self):
        if len(self.children) == 0:
            x = 'NONE'
        else:
            x = [a.name for a in self.children]
        return "{} Children:{}".format(self.name, x)

    def __getitem__(self, item):
        return self.name


def get_leaves(node):
    leaves = []
    if len(node.children) == 0:
        leaves.append(node)
    else:
        for child in node.children:
            leaves.extend(get_leaves(child))
    return leaves


def get_preorder(tree, X):
    X.append((tree.name))
    if len(tree.children) == 0:
        return
    for t in tree.children:
        X.append(get_preorder(t, X))


###################### Utility Functions
   ################################################
args.project_dir='/content/drive/MyDrive/HDSF/'
def wrap_with_variable(tensor, gpu, requires_grad=True):
    if gpu > -1:
```

```
        return Variable(tensor.cuda(gpu), requires_grad=requires_grad)
    else:
        return Variable(tensor, requires_grad=requires_grad)


def get_word_embeddings(source='google'):
    with open('Data/word_emb_' + source + '.pkl', 'rb') as f:
        embed = pickle.load(f)
    return embed


def creat_word_embedding():
    if os.path.exists('Data/word_emb_google.pkl'):
        return
    if not os.path.exists('GoogleNews-vectors-negative300.bin'):
        print("Please download https://drive.google.com/file/d/0
            B7XkCwpI5KDYNlNUTTlSS21pQmM/edit and place it in".format(
            args.project_dir))
        exit(-1)
    print('Creating word embeddings ...')
    model = word2vec.KeyedVectors.load_word2vec_format('/content/drive/
        MyDrive/HDSF/GoogleNews-vectors-negative300.bin',binary=True)

    all_words = pd.read_csv("Data/words.csv", names=['Index', 'Word', '
        Freq'])
    word_embeddings = np.zeros(shape=(len(all_words) + 1, args.word_dim
        ))
    non_exist_words = []
    for w in all_words.values:
        if (w[1] in model):
            word_embeddings[w[0]] = model[w[1]]
        else:
            word_embeddings[w[0]] = np.random.rand(1, args.word_dim)[0]
            non_exist_words.append(w[1])
```

```
    word_embeddings[len(all_words)] = np.random.rand(1, args.word_dim)
        [0]
    with open(args.project_dir + 'Data/word_emb_google.pkl', 'wb') as
        pkl:
         pickle.dump(word_embeddings, pkl)
    print('Word embedding file created and save in {}'.format(args.
        project_dir + 'Data/word_emb_google.pkl'))
    #print(non_exist_words, len(non_exist_words))


def get_split_data(split='train'):
    documents = []
    with open('Splits/' + split + '.split.csv', 'r') as f:
        records = f.read().splitlines()

    for record in records:
        label = record.split(',')[-1]
        if label == 'Fake':
            l = 1
        else:
            l = 0
        with open( args.project_dir+'Data/' + record.split(',')[1], 'r'
            ) as f:
            word_indices = f.read().splitlines()
            word_indices = [w.split(',') for w in word_indices]
            documents.append({"word_indices": word_indices, 'label': l
                })
    return documents


def get_num_words():
    all_words = pd.read_csv("Data/words.csv", names=['Index', 'Word', '
        Freq'])
    return len(all_words)
```

```
def construct_dependecy_tree(matrixp, rootp):
    trees = []
    root_index = np.argmax(rootp)
    root = myTree(name=str(root_index))
    trees.append(root)
    current_nodes = []
    current_nodes.append(root_index)
    matrixp[:, root_index] = np.array([-1 for _ in range(len(rootp))])
    flag = True
    child_parent_diff = 0
    while flag:
        currmax = -99
        father = -1
        child = -1
        for c in current_nodes:
            m = np.argmax(matrixp[c, :])
            if matrixp[c, m] > currmax:
                currmax = matrixp[c, m]
                father = c
                child = m

        father_index_in_the_list = -1
        for i, t in enumerate(trees):
            if t.name == str(father):
                father_index_in_the_list = i
                break
        node = myTree(name=str(child))
        child_parent_diff += np.abs(int(child) - int(trees[
            father_index_in_the_list].name))
        node.parent = trees[father_index_in_the_list]
        trees[father_index_in_the_list].add_child(node)
        trees[father_index_in_the_list].data = currmax
```

```
        trees.append(node)
        current_nodes.append(child)
        matrixp[:, child] = np.array([-1 for _ in range(len(rootp))])
        if len(current_nodes) == len(rootp):
            flag = False
    return trees[0], child_parent_diff


def dependecy_tree_stat(dir):
    with open(dir + 'matrix.pkl', 'rb') as f:
        x = pickle.load(f)
        lengths, all_doc_doc_dependency_tree_info, labels = x[0], x[1],
            x[2]
        pijs = [a[0].numpy() for a in all_doc_doc_dependency_tree_info]
        piroots = [a[1].numpy() for a in
           all_doc_doc_dependency_tree_info]
    fake_doc_stat = []
    real_doc_stat = []
    for index, (matrixp, rootp, label) in enumerate(zip(pijs, piroots,
       labels)):
        tree, child_parent_diff = construct_dependecy_tree(matrixp,
           rootp)
        X = []
        get_preorder(tree, X)
        Y = []
        for ss in X:
            if ss is not None:
                Y.append(int(ss))
        preorder_diff = 0
        for i, n in enumerate(Y):
            preorder_diff += np.abs(i + 1 - n)
        leaves = get_leaves(tree)
        if label == 1:
```

```
            fake_doc_stat.append([label, len(matrixp), len(leaves) / (
                np.log10(len(matrixp))),
                                  child_parent_diff / (np.log10(len(
                                      matrixp))),
                                  preorder_diff / (np.log10(len(matrixp
                                      ))))])
        else:
            real_doc_stat.append([label, len(matrixp), len(leaves) / (
                np.log10(len(matrixp))),
                                  child_parent_diff / (np.log10(len(
                                      matrixp))),
                                  preorder_diff / (np.log10(len(matrixp
                                      ))))])

    fake_doc_stat = np.array(fake_doc_stat)
    real_doc_stat = np.array(real_doc_stat)
    return fake_doc_stat, real_doc_stat

#creat_word_embedding()
```

واژه‌نامه فارسی به انگلیسی

اخبار جعلی . . . . . Fake News

ارزیابی . . . . . Valuation

متن کاوی . . . . . Text Mining

یادگیری عمیق . . . . . Deep Learning

یادگیری ماشین . . . . . Machine learning

داده های آزمایشی . . . . . Test Data

تابع فعالیت . . . . . Activation Function

تابع هزینه . . . . . Cost Function

شبکه عصبی پیچشی . . . . . Convolutional neural network

شبکه عصبی بازگشتی . . . . . Recurrent neural network

تعبیه کلمات . . . . . Word Embedding

حافظه طولانی کوتاه مدت . . . . . Long Short Term Memory

شبکه عصبی تمایزگر رویداد . . . . . Event Adversarial Neural Network

ساختار سلسله مراتبی . . . . . Hierarchical Structure

شبکه توجه . . . . . Attention network

قابل توضیح . . . . . explainable

پس انتشار خطا . . . . . Back propagation

دروازه . . . . . Gate

علائم نگارشی . . . . . Punctuation

کلمات توقف . . . . . Stop Word

پیش پردازش . . . . . Preprocessing

کوتاه کردن . . . . . Stemming

فرایند احساسات . . . . . Sentiment process

واژه‌نامه انگلیسی به فارسی

اخبار جعلی . . . . . Fake news

نرخ یادگیر . . . . . Learning Rate

کلمات تعبیه شده . . . . . word embding

شبکه های عصبی بازگشتی . . . . . Recurrent Neural Network

داده های پرت . . . . . Drop out

حافظه طولانی کوتاه مدت . . . . . LSTM

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Rostami	Name:Zahra	Student No.:9713185093
Supervisor: Dr. Mina Masoudifar		
Advisor: Dr. Alireza Ghodsi		
Faculty of Mathematics and Computer Science		
Program: Applied Mathematics Field:Operational Research		

Title of thesis: Writing projects, theses and dissertations using HSU-Thesis Class

Keywords:Writing Thesis, Template, LaTeX, XePersian

Abstract: Graduate students who will be submitting a dissertation or research master's thesis should familiarize themselves with the Graduate School's formatting requirements and deadlines for submission. This thesis document class is used to produce either a master's or a doctoral thesis that meets the formatting requirments of the Graduate School without any knowledge about formatting requirements. This thesis studies on writing projects, theses and dissertations using HSU-Thesis Class.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

A Thesis Submitted in Partial Fulfilment of the Requirement for the Degree of Master of Science in Applied Mathematics

Writing projects, theses and dissertations using HSU-Thesis Class

Supervisor:
Dr. Mina Masoudifar

Advisor:
Dr. Alireza Ghodsi

By:
Zahra Rostami

August 2016