دانشکده مهندسی کامپیوتر

تمرین درس

پردازش زبان طبیعی

استاد درس: دکتر مینایی

سجاد رمضانی 96471298
یاسمین مدنی 97532265

نیم‌سال دوم
سال تحصیلی ۰۱-۰۰

فاز ۱

۱.۱ مقدمات

در این پروژه ما قصد داریم بتوانیم اخبار را بر اساس عنوان آنها دسته بندی کنیم موضوع داده تیتر خبرهای موجود در سایت های خبری و دسته بندی آن است. که ما به دو زبان فارسی و انگلیسی دیتا جمع آوری کرده ایم از این جهت که اگر بخواهیم مدل خوبی را پیشنهاد دهیم باید به گونه ای باشد که قابلیت تعمیم پذیری به چند زبان را داشته باشد از آنجا که داده ها به صورت تیتر خبر هستند پس زبان آنها رسمی بوده و محاوره ای نیست.

۲.۱ داده

داده های گرده آوری شده در دو زبان فارسی و انگلیسی و همینطور به دو صورت دیتا های اسکریپ شده از وب سایت های خبری و داده های استاندارد برای این تسک بوده. در قسمت داده های اسکریپ شده از ابزار Scrapy برای جمع آوری داده ها استفاده شده و چهاراسپایدر برای وب سایت های متفاوت نوشته شده است. که به صورت خلاصه در زیر بیان شده است

- HuffingPostSpider برای استخراج داده های سایت huffpost که شامل اخبار از دسته های گوناگون می باشد.
- Techcrunch Scrapper: اسپایدر برای استخراج داده های سایت تک کرانچ که شامل اخبار مرتبط با تکنولوژی است
- Nytimes Rss Spider: اسپایدر پارس کننده اخبار سایت New York times

- اسپایدر سایت خبر انلاین: برای یکی از مهم ترین وب سایت های خبری ایران هم یک اسپایدر نوشته ایم

جدا از موارد بیان شده از دیتا ست های آماده برای این تسک نیز استفاده شده است که به شرح زیر است

- News Category Dataset که دیتاستی از همان سایتhuffpost که بخشی از دیتا را از آن به دست آورده ایم می باشد برای گرفتن دیتاست می توانید به اینجا مراجعه فرمایید
- BBC news Data: داده های اخبار سایت BBC برای گرفتن دیتاست می توانید به اینجا مراجعه فرمایید
- دیتا ست دیگر داده های اخبار افتصادی است که درکگل منتشر شده است که می توان در اینجا مشاهده کنید

۱.۲.۱ جمع آوری دیتا

در این قسمت توضیح کوتاهی درمورد نحوه جمع آوری دیتا و اسپایدر های نوشته شده می پردازیم

همانطور که گفته شد با استفاده از فریم ورک Scrapy به راحتی می توان اسپایدر برای وب سایت های متفاوت نوشت و سپس با نوشتن پایپلاین مناسب این دیتا ها را در فرمت مناسب نگه داری کرد. که در اینجا یک نمونه از اسپایدر ها را بررسی می کنیم با استفاده از این فریم ورک ابتدا یک کلاس می نویسیم که با ارث بری از کلاس های موجود تنها کافی است چند متد آن را پیاده سازی کنیم برای این کار به مثال گردآوری داده از وب سایت خبر انلاین توجه کنید

ابتدا لینک و فرمت پارامتر های ان در متغیر urls آورده شده است که در ان دو پارامتر tp که در واقع ایدی موضوع است و pi که شماره صفحه است را مشاهده می کند که برای شروع ۱۰۰۰ تا از ایدی ها برای موضوعات متفاوت داده شده است. سپس هر صفحه ایی که درخواست داده می شود پس از گرفتن html ان به تابع parse داده می شود تا قسمت های مورد نیاز از آن استخراج شود که با نگاه کردن به فرمت صفحه و استخراج css selector های مناسب این موارد استخراج می شود و تا صفحه مشخص شده در page limit از اخبار این دسته بندی آورده می شود.

و در اخر هم یک ابجکت از نوع Title Item برگردانده می شود

برای نوشتن این موارد هم یک pipeline ارایه شده است که در فرمت جیسون دیتا ها را بنویسید که کد آن هم در زیر ارایه شده است. برای مشاهده دقیق تر این موارد به ریپومراجعه شود

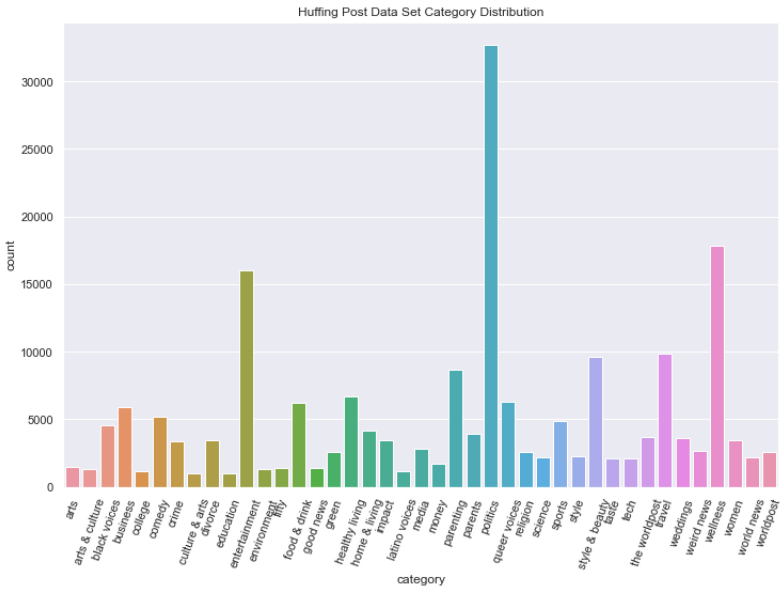
```python
class KhabarOnlineScraper(scrapy.Spider):
    name = "khabaronline"
    page_limit = 200
    def start_requests(self):
        urls = [
            f"https://www.khabaronline.ir/archive?pi=1&tp={i}"
            for i in range(10000)
        ]
        for url in urls:

            yield scrapy.Request(url=url, callback=self.parse)
    def parse(self, response):
        for item in response.css('section[id="box202"]').css("ul")[0].css("li"):
            title = item.css("h3").css("a::text")[0].get()
            category = item.css("p").css("span").css("a::text").get()
            yield TitleItem(title=title, category=category)
        try:
            if int(re.search("pi=(\d+)", response.url)
            .groups(1)) < self.page_limit:
                yield scrapy.Request(
                    re.sub(
                        "pi=(\d+)",
                        lambda exp: "pi={}".format(int(exp.groups()[0]) + 1),
                        response.url,
                    ),
                    self.parse,
                )
        except:
            pass
```

```python
class JsonWriterPipeline:

    def open_spider(self , spider):
        self.file = open(ITEM_OUTPUT_PATH , 'w',encoding='utf-8')

    def close_spider(self , spider):
        self.file.close()

    def process_item(self, item, spider):
        line = json.dumps(ItemAdapter(item).asdict(),ensure_ascii=False) + '\n'
        self.file.write(line)
        return item
```

۲.۲.۱ دیتاست های آماده

news-category-dataset

این مجموعه داده شامل حدود ۲۰۰ هزار عنوان خبری از سال ۲۰۱۲ تا ۲۰۱۸ است که از HuffPost به دست آمده است. هر عنوان خبری یک دسته بندی مربوطه دارد که دسته بندی ها را در شکل زیر مشاهده می کنید.



Dataset: BBC

شامل ۲۲۲۵ داده از وب‌سایت خبری بی‌بی‌سی در پنج حوزه موضوعی از سال ۲۰۰۴ تا ۲۰۰۵ است. دارای ۵ کلاس از نوع (کسب و کار، سرگرمی، سیاست، ورزش، فناوری) می باشد

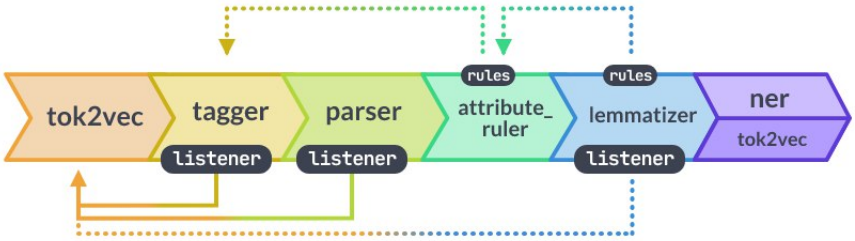
Category	Heading	Article
Business	510	510
Entertainment	386	386
Politics	417	417
Sport	511	511
Tech	401	401

US Financial News Articles

داده های مربوط به اخبار اقتصادی جمع آوری شده از سایت های Bloomberg.com, CNBC.com, reuters.com, wsj.com, fortune.com

۳.۱ پیش پردازش

در این مرحله نیاز داریم تا داده های جمع آوری شده را تمیز و پیش پردازش کنیم که روش های گوناگون و تسک های مختلفی از جمله ریشه گیری کلمات، حذف استاپ وردها و ... دارد. در اینجا از پایپ لاین آموزش داده شده ی en_core_web_sm استفاده می کنیم این پایپ لاین ابتدا متن را نشانه‌گذاری می‌کند تا یک شی Doc تولید کند. سپس Doc در چندین مرحله مختلف پردازش می‌شود. خط لوله معمولاً شامل یک tagger یک lemmatizer، یک parserو یک entity recognizer است. که هر یک از این بخش ها متن پردازش شده را به بخش بعدی می فرستد. برای مشاهده داک مدل به اینجا مراجعه کنید.



در ادامه در تابع زیر باقی عملیات لازم جهت پاکسازی متن را انجام می دهیم. که این شامل حذف استاپ وردها،اعداد،فاصله هاو..می شود.

```
def clean_doc(d):
    doc = []
    for t in d:
        if not any([t.is_stop, t.is_digit,
        not t.is_alpha, t.is_punct, t.is_space,
        t.lemma_ == '-PRON-']):
            doc.append(t.lemma_)
    return ' '.join(doc)


def preprocess(articles):
    iter_articles = (article for article in articles)
    clean_articles = []
    for i, doc in
    enumerate(nlp.pipe(iter_articles, batch_size=100,n_process=8), 1):
        if i % 1000 == 0:
            print(f"{i / len(articles):2%.}", end=" ", flush=True)
        clean_articles.append(clean_doc(doc))
    return clean_articles
```

برای داشتن دید بهتر تعدادی مثال از تیتر اخبار پیش و پس از پیش پردازش در تصاویر زیر به نمایش گذاشته شده است.

داده پیش و پس از پردازش

raw: Captive Medic's Bodycam Shows Firsthand Horror Of Mariupol
cleaned: Captive Medic Bodycam show Firsthand Horror Mariupol

داده پیش و پس از پردازش

raw: Russia Is Firing Its Senior Commanders. What Does That Mean For Ukraine War?
cleaned: Russia fire senior commander mean Ukraine War

داده پیش و پس از پردازش

raw: LinkedIn Settles With U.S. Over Alleged Pay Discrimination
cleaned: LinkedIn Settles Alleged Pay discrimination

داده پیش و پس از پردازش

raw: Starbucks Workers Have Unionized More Than 50 Stores In The U.S.
cleaned: Starbucks Workers unionize Stores

یک مثال از شرایطی که می تواند در تصمیم گیری مدل در آینده تاثیر بگذارد

raw: LinkedIn Settles With U.S. Over Alleged Pay Discrimination
cleaned: LinkedIn Settles Alleged Pay discrimination

۴.۱ آمار مربوط به دیتاهای جمع آوری شده

از آنجا که دیتاهای مختلفی جمع آوری کرده ایم در بخش زیر تعدادی نمودار مربوط به هر یک از این دیتاستهارا به نمایش گذاشته ایم اما از آنجا که انتظار می رود داده های جمع شده از سایت huffpostو دیتاست آن پاسخ بهتری در فازهای بعدی برای ما فراهم کند آمار خواسته شده در داک پروژه را در مورد این دیتاست نوشته ایم. برای بدست آوردن این اماره ها از توابع موجود در nltk استفاده شده است.

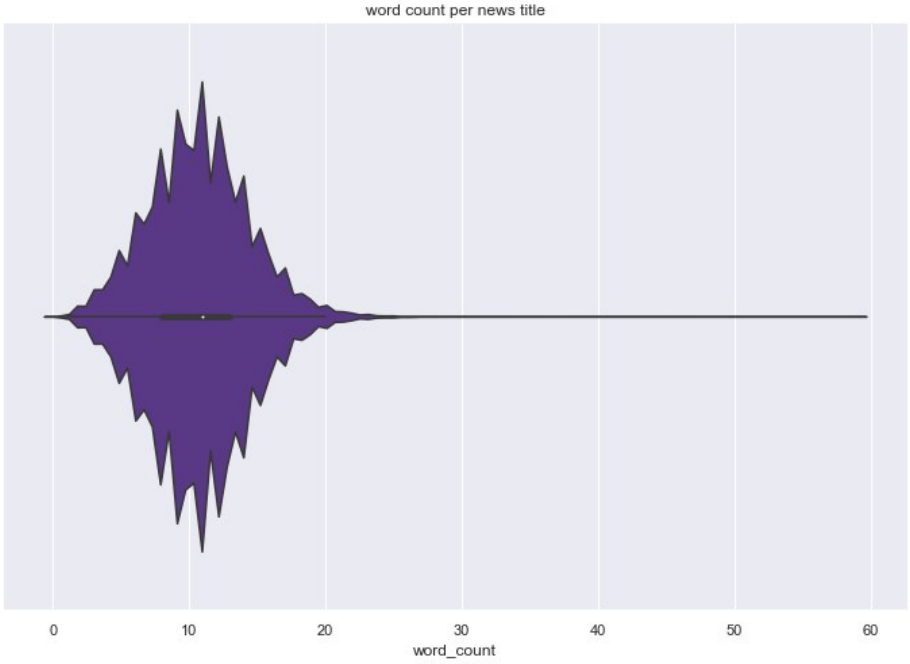
برای تفکیک جملات از sent_tokenize استفاده شده است. که در صورت پیش فرض با استفاده از punctuation ها می تواند جملات را تفکیک کند برای مطالعه بیشتر به اینجا مراجعه کنید

برای تفکیک کلمات از word_tokenize استفاده شده است. که در صورت پیش فرض با استفاده از TreebankWordTokenizer استفاده می کند.

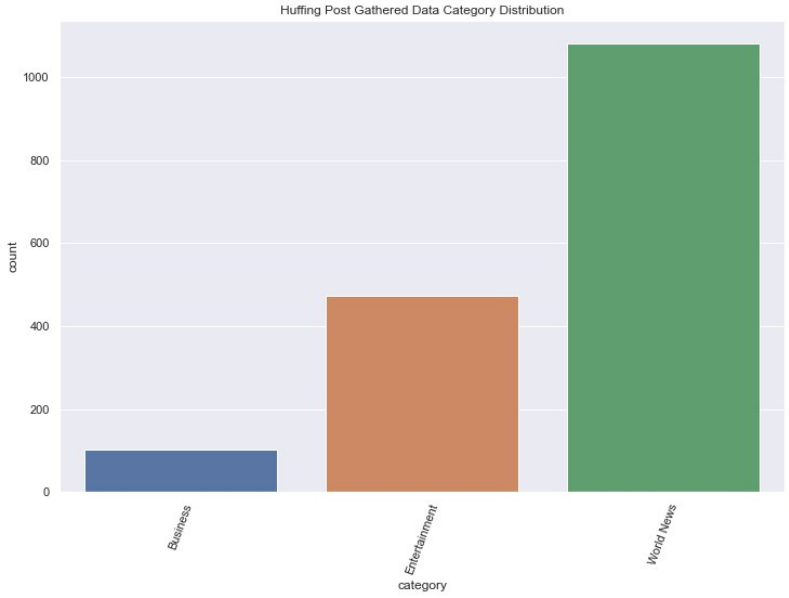
huffpost ۱.۴.۱

- تعداد واحد داده : ۲۰۱۰۵۹ عنوان خبر
- تعدا جملات : ۲۰۹۵۵۷
- تعداد کلمات ۲۱۴۶۸۹۱
- تعداد کلمات منحصر به فرد: ۶۴۷۳۲(بعد تمیز کردن)
- تعداد کلمات منحصر به فرد: ۷۴۵۰۵(قبل تمیز کردن)

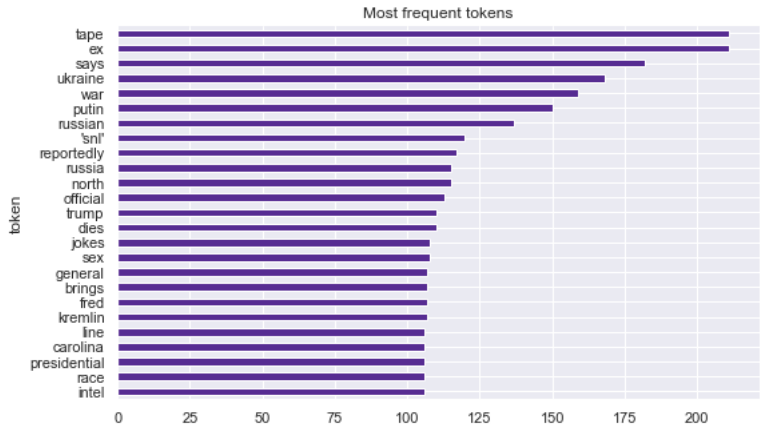
تعداد توکنهای موجود برحسب هدلاین

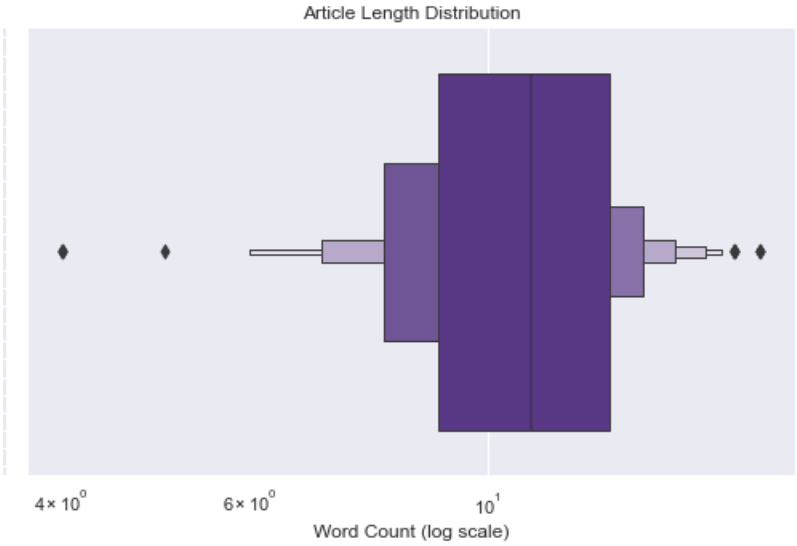


برچسب داده ها و تعداد آنها در داده های کرال شده از سایت

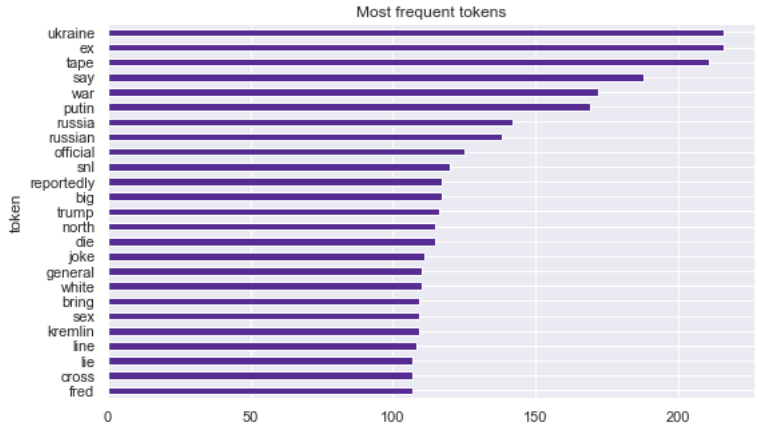


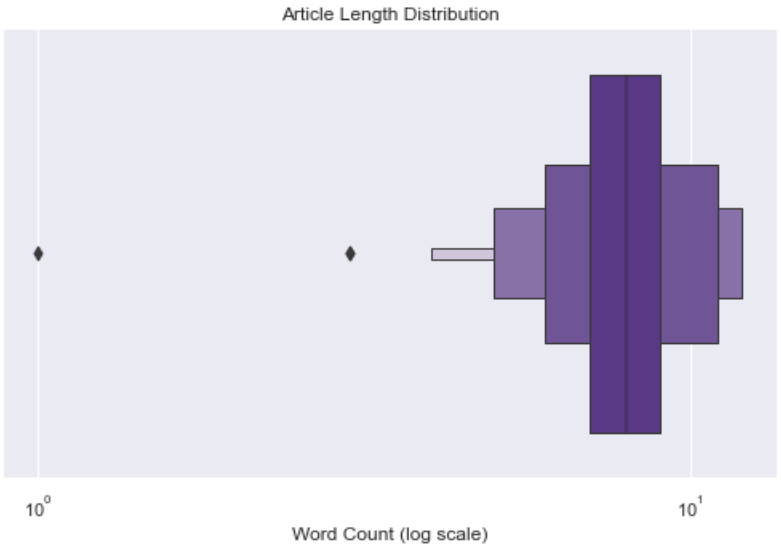
در شکل زیر نمودارهای توکن های پرتکرار و طول جملات داده های جمع آوری شده را مشاهده می کنیم.



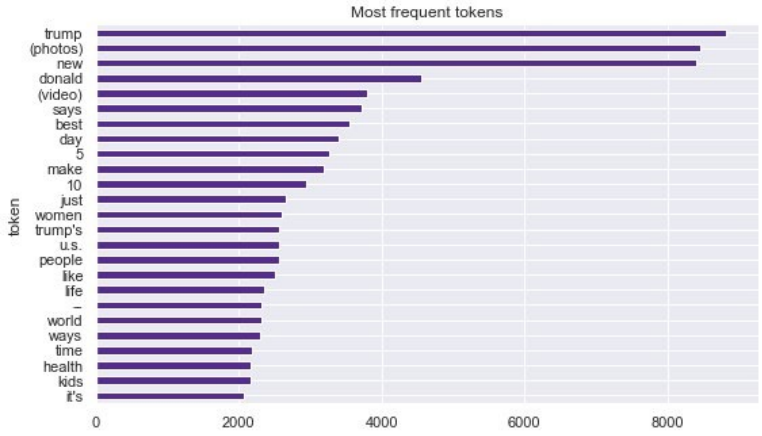


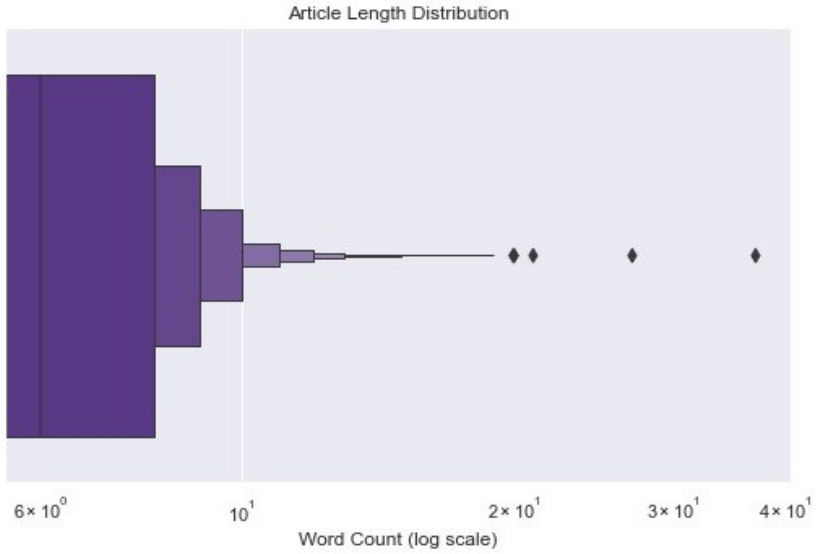
همچنین در شکل زیر نمودارهای مربوطه را پس از عملیات پیش پردازش می توان مشاهده کرد

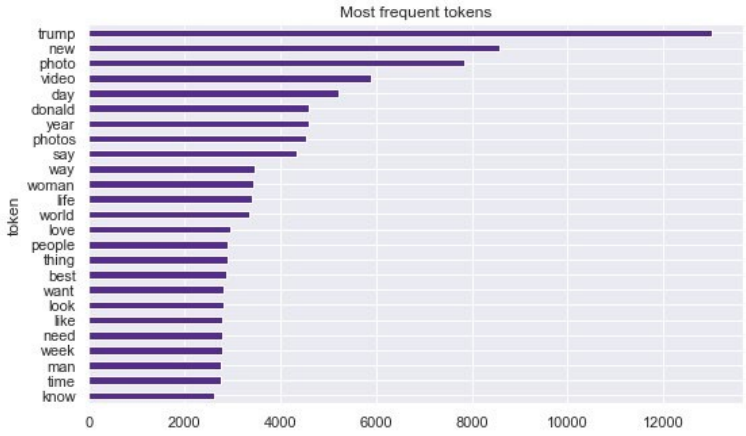


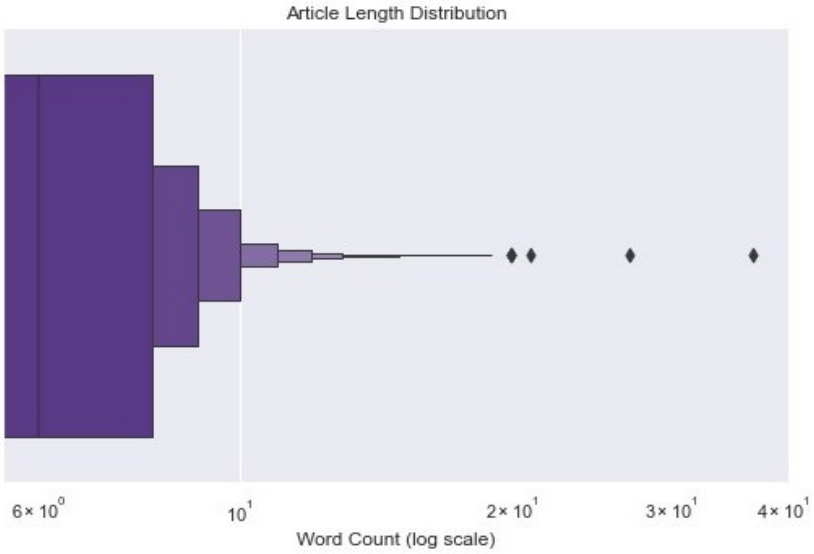


به علاوه بخشی از داده های ما از طریق دیتاست توضیح داده شده پیش از این تامین شده از این رو این عملیات پیش پردازش را روی داده های موجود در دیتاست نیز اجرا می کنیم همان طور که در نمودارهای زیر مشاهده می شود داده های موجود در دیتا بیس حاوی مقادیر اطلاعات از قبیل اعداد و ... اند که با انجام عملیات پیش پردازش حذف شده و اولویت در ترتیب توکن های پر تکرار تغییر می کند.


Most frequent tokens
token
trump
(photos)
new
donald
(video)
says
best
day
5
make
10
just
women
trump's
u.s.
people
like
life
–
world
ways
time
health
kids
it's
0
2000
4000
6000
8000

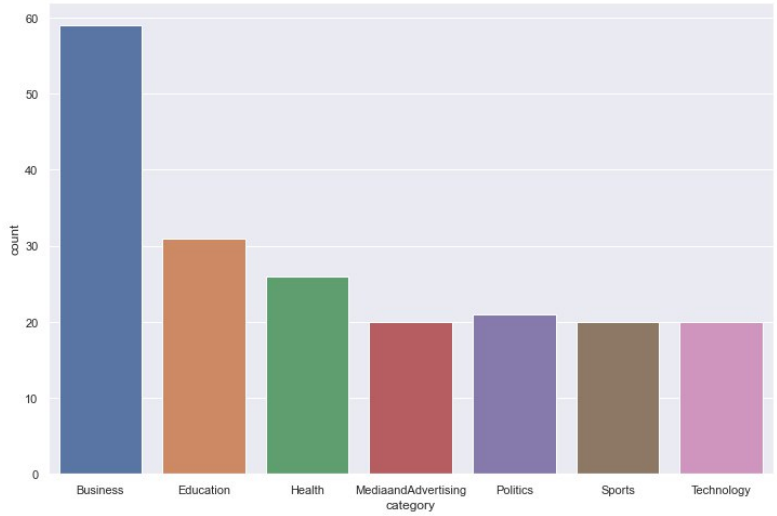

Article Length Distribution
6× 10^0
10^1
2× 10^1
3× 10^1
4× 10^1
Word Count (log scale)


Most frequent tokens
token
trump
new
photo
video
day
donald
year
photos
say
way
woman
life
world
love
people
thing
best
want
look
like
need
week
man
time
know
0
2000
4000
6000
8000
10000
12000


Article Length Distribution
6× 10^0
10^1
2× 10^1
3× 10^1
4× 10^1
Word Count (log scale)

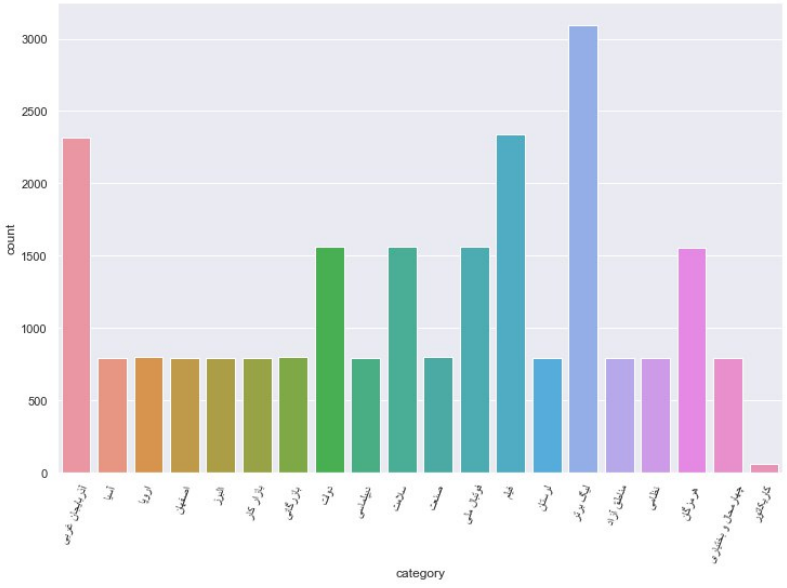
۲.۴.۱ nyt

برچسب داده ها و تعداد آنها در داده های کرال شده از سایت



۳.۴.۱ خبرآنلاین

برچسب داده ها و تعداد آنها در داده های کرال شده از سایت



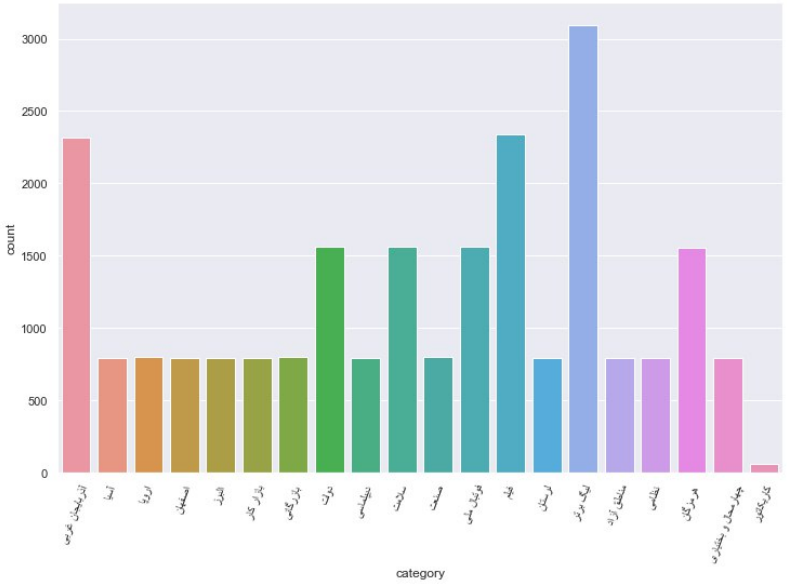
فاز ۲

۱.۲ مقدمه

۱.۱.۲ داده انتخاب شده

از آنجا که در فاز قبل داده های انگلیسی به صورت جز بررسی شده بودند و در داک فاز دوم پروژه آموزش روی مدل پارس برت مدنظر بود و داده فارسی نیز در فاز قبل جمع آوری شده بود یک بار دیگر به گزارش تعدادی از آمار دیتاست فارسی می پردازیم. و سپس فاز دوم را روی داده های فارسی ادامه خواهیم داد. داده های کرال شده با استفاده از اسپایدر های توضیح داده شده در فاز اول از سایت خبرآنلاین جمع آوری شده و برای لیبل آنها از دسته بندی خود سایت برای اخبار و استخراج آن از css سایت کمک گرفته ایم تا دقیق تر باشد.

برچسب داده ها و تعداد آنها در داده های کرال شده از سایت



۲.۱.۲ توضیحات

در اینجا لازم است که توضیح دهیم که داده های به دست آمده را به صورت کامل بالانس نکرده ایم چرا که این بالانس نبودن برای ما اهمیت دارد چرا که کلاس با داکیومنت بیشتر یعنی اخبار بیشتری را شامل می شود و این عدم توازن برای یادگیری بهتر برای ما ارزشمند است. اما داده ها تا حدود اولیه ای بالانس شده اند.

۲.۲ بخش اول:تولید جملات

۱.۲.۲ Language modeling

مدل‌سازی زبان استفاده از تکنیک‌های مختلف آماری و احتمالی برای تعیین احتمال وجود یک توالی معین از کلمات در یک جمله است. مدل‌های زبان بدنه داده‌های متنی را تجزیه و تحلیل می‌کنند تا مبنایی برای پیش‌بینی‌های کلمه‌شان فراهم کنند.

مدل سازی زبان در اپلیکیشن های مدرن NLP بسیار مهم است چراکه که ماشین ها می توانند اطلاعات کیفی را درک کنند. هر نوع مدل زبانی، به نوعی، اطلاعات کیفی را به اطلاعات کمی تبدیل می کند. این به افراد اجازه می‌دهد تا به میزان محدودی با ماشین‌ها ارتباط برقرار کنند.

۲.۲.۲ Masked language modelling

مدل‌سازی زبان ماسک‌شده را می‌توان شبیه به مدل‌سازی رمزگذاری خودکار در نظر گرفت که بر اساس ساختن نتایج از ورودی‌های نامرتب یا خراب شده کار می‌کند. masking اینگونه کار می کند که ما کلمات را از یک دنباله ورودی یا جملات پنهان می‌کنیم و مدل طراحی‌شده برای تکمیل جمله نیاز به پیش‌بینی کلمات پوشانده شده دارد. ما می‌توانیم این نوع روش مدل‌سازی را با فرآیند پر کردن جاهای خالی یک برگه امتحانی مقایسه کنیم. با دانستن روش عملکرد این روش درمی یابیم که ما باید از این مدل ها در جایی استفاده کنیم که لازم است بافت کلمات را پیش بینی کنیم. از آنجایی که کلمات می توانند در مکان های مختلف معانی متفاوتی داشته باشند، مدل نیاز به یادگیری بازنمایی عمیق و چندگانه کلمات دارد.

۳.۲.۲ BERT or ParsBERT

BERT از Transformer استفاده می کند، مکانیزمی که روابط متنی بین کلمات (یا کلمات فرعی) را در یک متن یاد می گیرد. Transformer در شکل اصلی خود شامل دو مکانیسم مجزا است یک رمزگذار که ورودی متن را می‌خواند و یک رمزگشا که یک پیش‌بینی برای کار ایجاد می‌کند. از آنجایی که هدف BERT تولید یک مدل زبان است، تنها مکانیزم رمزگذار ضروری است.

ParsBERT یک مدل زبان تک زبانه بر اساس معماری BERT گوگل است. این مدل بر روی مجموعه‌های بزرگ فارسی با سبک‌های نوشتاری مختلف از موضوعات متعدد (مانند علمی، رمان، اخبار) با بیش از ۹.۳ میلیون سند، ۷۳ میلیون جمله و ۳.۱ میلیارد کلمه از قبل آموزش داده شده است.

۴.۲.۲ Fine-tuning

در NLP به روند آموزش مجدد یک مدل زبان از پیش آموزش داده شده با استفاده از داده های سفارشی خود اشاره دارد. در نتیجه روند تنظیم دقیق، وزن مدل اصلی به‌روزرسانی می‌شود تا ویژگی‌های داده‌های دامنه و وظیفه‌ای که ما به آن علاقه‌مند هستیم را در نظر بگیرد.

۵.۲.۲ Hugging Face

Hugging Face یک جامعه و پلتفرم علم داده است که امکانات زیر را ارائه می دهد

- ابزارهایی که کاربران را قادر به ساخت، آموزش و استقرار مدل‌های ML بر اساس کدها و فناوری‌های منبع باز (OS) می‌سازد.

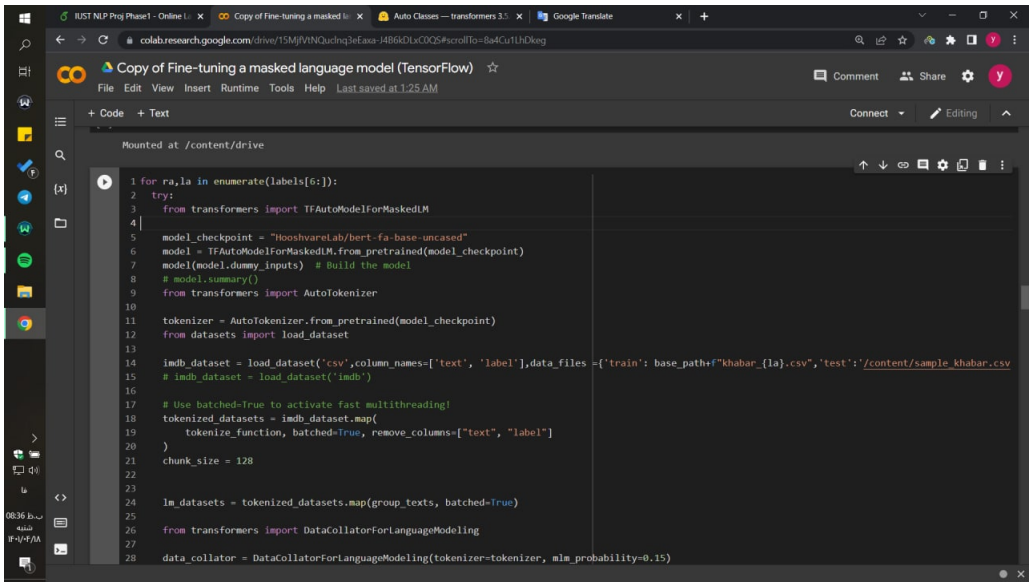
- مکانی که در آن جامعه گسترده ای از دانشمندان داده، محققان و مهندسان ML می توانند گرد هم آیند و ایده های خود را به اشتراک بگذارند، حمایت دریافت کنند و در پروژه های منبع باز مشارکت کنند.

برای بسیاری از برنامه‌های NLP که شامل مدل‌های ترانسفورماتور می‌شوند، می‌توانید به سادگی یک مدل از پیش آموزش دیده را از Hugging Face Hub بگیرید و آن را مستقیماً روی داده‌های خود برای کارتنظیم کنید. به شرطی که پیکره مورد استفاده برای پیش‌آموزش با پیکره مورد استفاده برای تنظیم دقیق تفاوت زیادی نداشته باشد، یادگیری انتقال معمولاً نتایج خوبی ایجاد می‌کند.

۶.۲.۲ توضیحات کد و نتایج

در این بخش به ازای هر کلاس جملات را در کنار هم قرار داده ایم با استفاده از یک احتمال تعدادی از این کلمات را ماسک کرده ایم.اگرچه کارهای پیش بردازشی دیگری چون تعیین توکن ورودی خروجی نیز باید صورت بگیرد در انتها این داده ها را به صورت چانک شده برای آموزش به مدل می دهیم. برای تولید جملات هم از همین روش ماسک کردن استفاده کرده ایم چرا که اگر قرار بود به صورت مستقل جمله تولید شود با تسک جداگانه NLGروبرو بودیم. با توجه به توضیحات قبلی در قدم اول مدل از پیش آموزش دیده را لود میکنیم.

در بسیاری از موارد، معماری مورد نظر ما را می‌توان از روی نام یا مسیر مدل از پیش‌آموزش دیده که به متد from pretrained می‌دهیم حدس زد. AutoClasses این کار را انجام دهند تا به طور خودکار مدل مربوطه را با توجه به نام/مسیر وزن ها/پیکربندی/واژگان از پیش آموزش دیده بازیابی کنیم. برای نمونه بخش ابتدایی سلول کد برای فاین تیون کردن داده ها برای لیبل های متفاوت در زیر نمایش داده شده است.



پس از آموزش با داده های خودمان نتایج تولید شده توسط مدل به صورت زیر برای تعدادی از دسته های داده ها نمایش داده شده اند.

```
Perplexity: 30.07
Epoch 1/5
208/208 [==============================] - 123s 493ms/step - loss: 3.0681 - val_loss: 2.5665
Epoch 2/5
208/208 [==============================] - 105s 505ms/step - loss: 2.4385 - val_loss: 2.2355
Epoch 3/5
208/208 [==============================] - 105s 506ms/step - loss: 2.1514 - val_loss: 2.1366
Epoch 4/5
208/208 [==============================] - 105s 507ms/step - loss: 1.9980 - val_loss: 1.9539
Epoch 5/5
208/208 [==============================] - 105s 505ms/step - loss: 1.8791 - val_loss: 1.9024
3/3 [==============================] - 0s 112ms/step - loss: 1.8869
Perplexity: 6.60
```

شکل ۱.۲: perplexity befor and after

```
فرآیند کاهش آمارهای کرونا در تهران زمانبر است <<<
فرآیند کاهش آمارهای کرونا در تهران تدریجی است <<<
فرآیند کاهش آمارهای کرونا در تهران چگونه است <<<
فرآیند کاهش آمارهای کرونا در تهران جدی است <<<
فرآیند کاهش آمارهای کرونا در تهران اغازشده است <<<
به سود رئیسی صحت دارد [MASK] انصراف
انصراف زاکانی به سود رئیسی صحت دارد <<<
انصراف مظاهری به سود رئیسی صحت دارد <<<
انصراف قالیباف به سود رئیسی صحت دارد <<<
انصراف استقلال به سود رئیسی صحت دارد <<<
انصراف رییسی به سود رئیسی صحت دارد <<<
«در ورزشگاه ها [MASK] جدیدترین خبرهای «حضور
«جدیدترین خبرهای «حضور زنان در ورزشگاه ها <<<
«جدیدترین خبرهای «حضور بانوان در ورزشگاه ها <<<
«جدیدترین خبرهای «حضور مردم در ورزشگاه ها <<<
«جدیدترین خبرهای «حضور ایران در ورزشگاه ها <<<
«جدیدترین خبرهای «حضور کرونا در ورزشگاه ها <<<
به شوهرتون دادید [MASK] دوست جون ها شماها چه جوری خبر
دوست جون ها شماها چه جوری خبر بدی به شوهرتون دادید <<<
دوست جون ها شماها چه جوری خبر خوبی به شوهرتون دادید <<<
دوست جون ها شماها چه جوری خبر بد به شوهرتون دادید <<<
دوست جون ها شماها چه جوری خبر خوشی به شوهرتون دادید <<<
دوست جون ها شماها چه جوری خبر خوب به شوهرتون دادید <<<
```

شکل ۲.۲: مثال برای دسته شماره ۱ یا اخبار سیاسی اجتماعی

```
1/1 [==============================] - 3s 3s/step - loss: 3.0332
Perplexity: 20.76
Epoch 1/5
20/20 [==============================] - 35s 497ms/step - loss: 3.3059 - val_loss: 3.3086
Epoch 2/5
20/20 [==============================] - 10s 492ms/step - loss: 3.1465 - val_loss: 3.3066
Epoch 3/5
20/20 [==============================] - 10s 498ms/step - loss: 3.0516 - val_loss: 2.3184
Epoch 4/5
20/20 [==============================] - 10s 504ms/step - loss: 2.8499 - val_loss: 2.8691
Epoch 5/5
20/20 [==============================] - 10s 509ms/step - loss: 2.8165 - val_loss: 3.2012
1/1 [==============================] - 0s 92ms/step - loss: 2.4141
Perplexity: 11.18
```

شکل ۳.۲: perplexity befor and after

```
را دارند [MASK] کدام ستاره ها بالاترین
کدام ستاره ها بالاترین امتیاز را دارند <<<
کدام ستاره ها بالاترین وزن را دارند <<<
کدام ستاره ها بالاترین درجه را دارند <<<
کدام ستاره ها بالاترین رتبه را دارند <<<
کدام ستاره ها بالاترین کیفیت را دارند <<<
```

شکل ۴.۲: مثال برای دسته شماره ۱۳ اخبارفیلم و سینما

برای دیدن نمونه های بیشتر و کد کامل میتوان به نوت بوک مراجعه کرد.

۳.۲ بخش دوم

۱.۳.۲ مقدمه

برای ورودی دادن داده ها به مدل های کلاسیک معرفی شده در بخش دو و سه جدا از پیش پردازش هایی از قبیل حداقل تعداد کاراکتر و .. یک CountVetorizer را فیت کرده ایم.. CountVetorizer به معنای تجزیه یک جمله یا هر متنی به کلمات با انجام کارهای پیش پردازش مانند تبدیل همه کلمات به حروف کوچک و در نتیجه حذف کاراکترهای خاص است. مدل‌های NLP نمی‌توانند داده‌های متنی را درک کنند، آنها فقط اعداد را می‌پذیرند، بنابراین این داده‌های متنی باید بردار شوند. و فیچرهای ورودی مدل ها tfidf می باشد

به علاوه هنگام گزارش عملکرد از آنجا که دقت می تواند همیشه ملاک خوبی برای گزارش کردن نباشد و نسبت به سایز کلاس های موثر ایراد هایی داشته باشد confusion matrix نیز نمایش داده شده است که با استفاده از آن می توان تمامی آماره های دیگر را بدست آورد.

۲.۳.۲ Multinomial naïve bayes

این الگوریتم یک رویکرد یادگیری است که در پردازش زبان طبیعی رایج است. این روش با استفاده از قضیه بیز، برچسب یک متن، مانند یک ایمیل یا تیتر خبر را حدس می زند. احتمال هر تگ را برای یک نمونه مشخص محاسبه می کند و تگ را با بیشترین شانس خروجی می دهد. وقتی پیش‌بینی‌کننده B در دسترس باشد، احتمال کلاس A را محاسبه می‌کنیم. این بر اساس فرمول زیر است:

$$P(A|B) = P(A) * P(B|A)/P(B)$$

مزایا و معایب

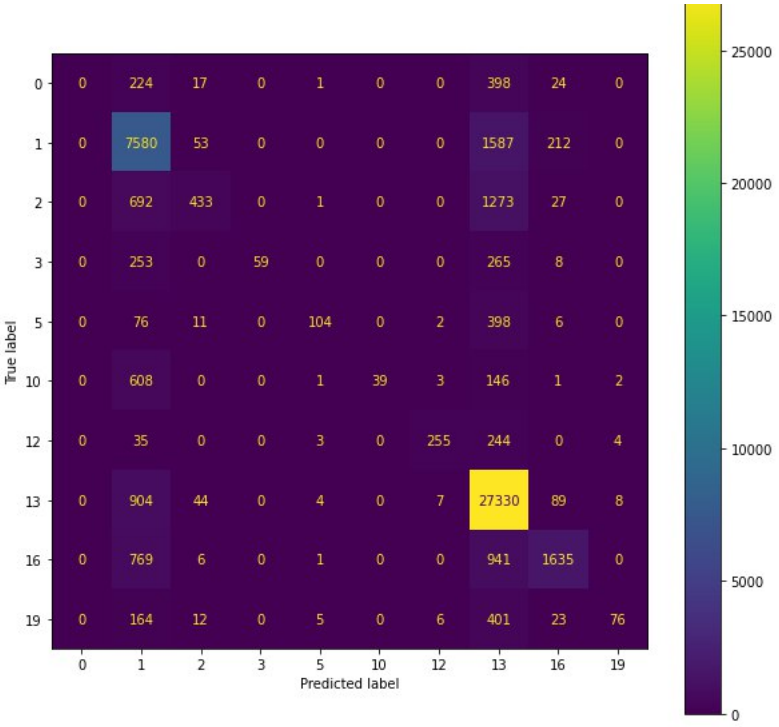
اجرای آن ساده است زیرا تنها کاری که باید انجام دهیم محاسبه احتمال است. با داده های پیوسته و گسسته کار می کند و می توان از آن برای پیش بینی ب کاربردهای بلادرنگ استفاده کرد. بسیار مقیاس پذیر است و می تواند مجموعه داده های عظیمی را به راحتی مدیریت کند.

کد و نتایج

ار آنجا که در این بخش به یک مدل ساده نیاز داشتیم از نایبو بیز استفاده کرده که قطعه کد مربوط به تعریف آن در شکل زیر آورده شده است

```
1 from sklearn.naive_bayes import MultinomialNB
2 clf = MultinomialNB().fit(X_train, y_train)
```

نتایج مربوط به آموزش این مدل در تصاویر زیر قابل مشاهده است



شکل ۵.۲: naive bayes confusion matrix

همانطور که مشاهده میشود این مدل به فیچرهایی که ماژوریتی داشته اند اهمیت بیشتری داده و فیچر های ماینور را نادیده گرفته است. همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

acc train : 0.7958605435011586

acc test : 0.7902043395828945

۴.۲ بخش سوم

۱.۴.۲ Logistic Regression

وقتی ویژگی هایی داریم که میان آنها کوریلیشن وجود دارد همبستگی وقتی اتفاق می افتد به طوری که وقتی برچسب ها تکرار می شوند، شانس بیشتری برای برجسته کردن ویژگی های تکراری در الگوریتم Naive Bayes وجود دارد. این در رگرسیون لجستیک اتفاق نمی افتد زیرا ویژگی های تکرار شونده تعداد دفعات کمتری شمرده می شوند و با تکرار جبران می شوند. Naive Bayes مستقیماً از ویژگی‌ها با هدف کمال بیشتر محاسبه می‌کند، اما اگر تعداد ویژگی‌ها بیشتر باشد، نتایج ضعیفی می‌دهد. کالیبراسیون ها را در نظر نمی گیرد و در صورت وجود وابستگی در ویژگی ها، آن را در نظر می گیرد و به ویژگی اضافه می کند و آنها را برجسته تر می کند اگر این ویژگی تأثیر منفی بگذارد، نتایج ضعیفی به همراه خواهد داشت. این مشکل در رگرسیون لجستیک نیست، زیرا کالیبراسیون ویژگی‌ها به موقع اتفاق می‌افتد، زمانی که ویژگی‌ها تعداد دفعات بیشتری اضافه می‌شوند و نتایج دقیق را ارائه می‌دهند.

خطا در Naive Bayes بیشتر است و اگر طبقه بندی بر روی مقدار کمی داده انجام شود و اگر ویژگی های وابسته ای وجود داشته باشد که در حین انجام محاسبات الگوریتمی نادیده گرفته شده اند، اشتباه فاحشی است. از این رو، Naive Bayes همیشه برای مشکلات طبقه بندی راه حلی نیست. این خطا در رگرسیون لجستیک کمتر است، جایی که می‌توانیم به راحتی پاسخ ویژگی‌های وابسته یا مستقل با داده‌های بزرگ را پیدا کنیم.

کد و نتایج

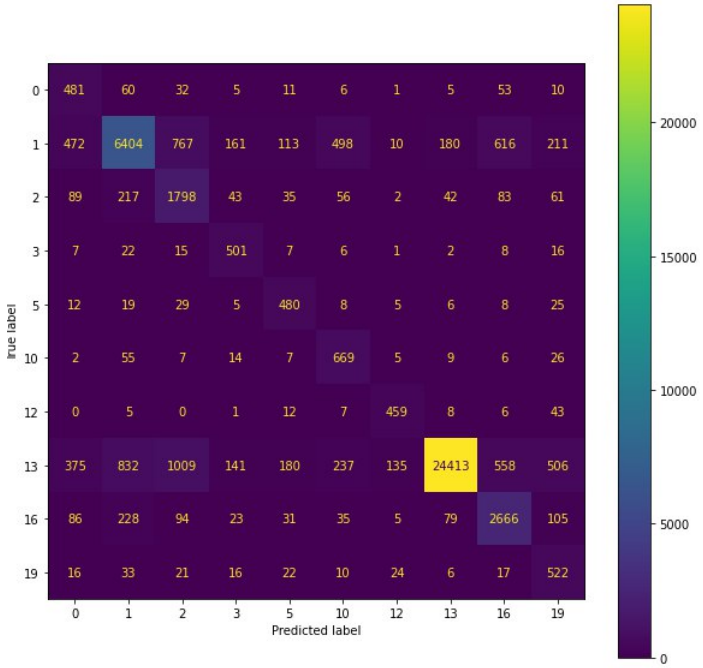
از این رو یکی از راه های بهبود تسک بخش دو می تواند استفاده از این روش باشد.

در تصویر زیر مدل تعریف شده است.

```
[ ]  1 from sklearn.linear_model import LogisticRegression
     2 clf = LogisticRegression(class_weight='balanced').fit(np.stack(X_train, axis=0), y_train)
```

شکل ۶.۲: logistic regression model

در تصویر زیر می توان نتایج به دست آمده را در قالب نمودار مشاهده کرد.



شکل ۷.۲: logistic regression confusion matrix

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

acc train : 0.8291605224352222

acc test: 0.8087844954708237

۲.۴.۲ Gradient Boosting

GBM یک تکنیک مجموعه ای است که در آن چندین درخت به صورت متوالی ساخته می شوند و هر درخت سعی می کند اشتباهات درخت قبلی را اصلاح کند. در کل تفاوت زیادی بین این روش و لاجستیک رگرشن وجود ندارد اما مدل GBM قادر است وابستگی های غیرخطی جزئی و تعامل بین متغیرها را به تصویر بکشد.

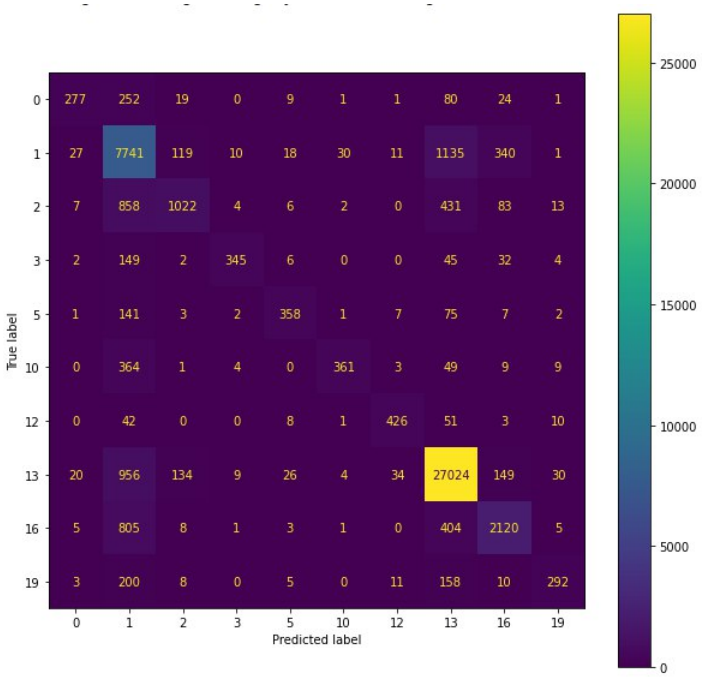
کد و نتایج

در تصویر زیر مدل تعریف شده است.

```
1 from sklearn.ensemble import GradientBoostingClassifier
2 clf = GradientBoostingClassifier().fit(X_train, y_train)
```

شکل ۸.۲: gradient boosting model

در تصویر زیر می توان نتایج به دست آمده را در قالب نمودار مشاهده کرد.



شکل ۹.۲: gradient boosting confusion matrix

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

acc train : 0.8523488519064673

test acc:0.8419212133979356

۳.۴.۲ Neural Net with Ngram

روش دیگر استفاده از شبکه های عمیق است. در این روش از n-gram های ۱ تا ۳ استفاده می کنیم و با شرط اینکه هرکلمه باید در ۱۰۰ داک دیده شود تا محاسبه شود از شمارش غلط های املایی و... جلوگیری می کنیم. بدیهی است که اسفاده از این ان گرام های بیشتر دامنه را بیشتر کرده و ترکیبات بیشتری را آموزش می بیند. سپس ورودی را به یک نورال نت ساده می دهیم که در تصاویر سایر ویژگی های آن مشاهده می شود.

کد و نتایج

در تصویر زیر مدل تعریف شده است.

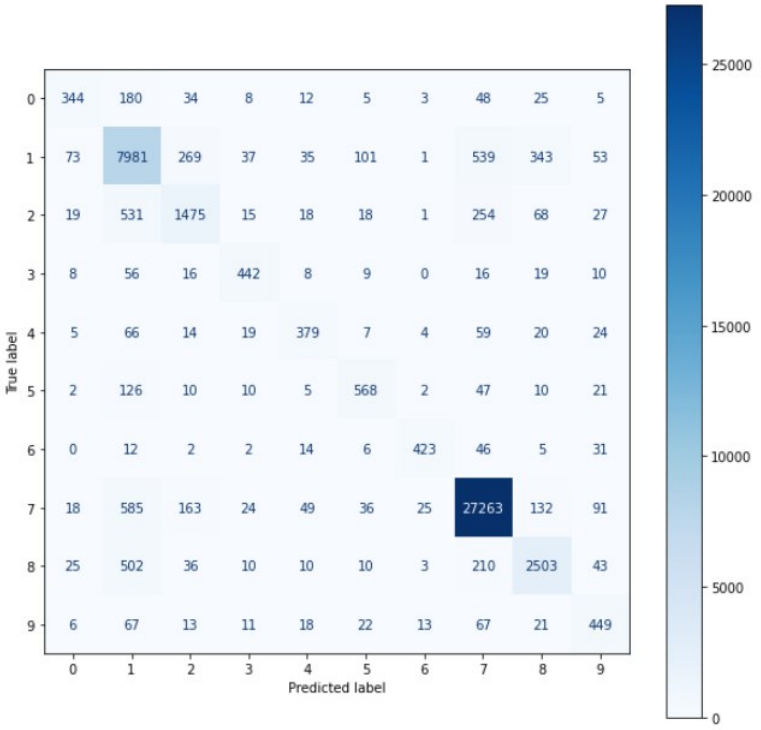
```
[ ]  model_clf = keras.Sequential(
        [
            layers.Dense(128, activation="relu", name="layer1"),
            layers.Dense(64, activation="relu", name="layer2"),
            layers.Dropout(0.2),
            layers.Dense(20,activation='softmax', name="layer3"),
        ]
    )
```

شکل ۱۰.۲: neural net with ngram model

```
model_clf.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
history = model_clf.fit(X_train, y_train,
                        epochs=5,
                        batch_size=128)

Epoch 1/5
```

شکل ۱۱.۲: neural net with ngram model

در تصویر زیر می توان نتایج به دست آمده را در قالب نمودار مشاهده کرد.



شکل ۱۲.۲: neural net with ngram confusion matrix

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

acc train :0.9666

test acc:0.8811249136924744

Simple RNN ۴.۴.۲

شبکه عصبی بازگشتی (RNN) نوعی شبکه عصبی مصنوعی است که از داده های متوالی یا داده های سری زمانی استفاده می کند. این الگوریتم‌های یادگیری عمیق معمولاً برای مشکلات ترتیبی یا زمانی، مانند ترجمه زبان، پردازش زبان طبیعی (nlp) ، تشخیص گفتار، و شرح تصاویر استفاده می‌شوند

شبکه های عصبی مکرر از داده های آموزشی برای یادگیری استفاده می کنند. آنها با "حافظه" خود متمایز

می شوند زیرا آنها اطلاعات را از ورودی های قبلی برای تأثیرگذاری بر ورودی و خروجی فعلی می گیرند. در حالی که شبکه‌های عصبی عمیق سنتی فرض می‌کنند که ورودی‌ها و خروجی‌ها مستقل از یکدیگر هستند، خروجی شبکه‌های عصبی بازگشتی به عناصر قبلی در توالی بستگی دارد.

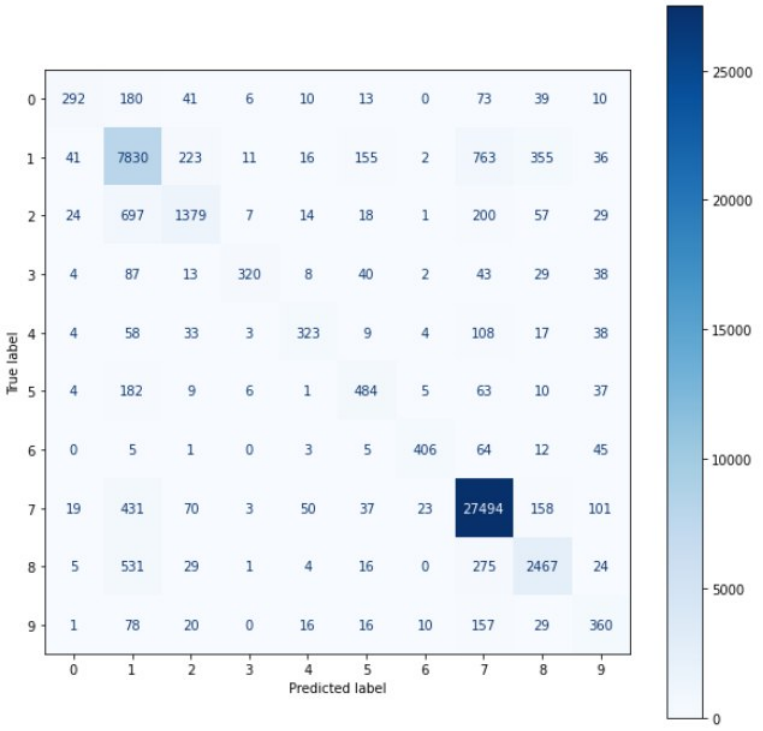
کد و نتایج

در تصویر زیر مدل تعریف شده است.

```
1 model = Sequential()
2 model.add(Embedding(max_features, 32))
3 model.add(layers.SimpleRNN(32))
4 model.add(layers.Dense(20, activation='softmax'))
```

شکل ۱۳.۲: simple RNN model

در تصویر زیر می توان نتایج به دست آمده را در قالب نمودار مشاهده کرد.



شکل ۱۴.۲: simple RNN confusion matrix

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

train acc: 0.8960

test acc : 0.8711817860603333

LSTM ۵.۴.۲

شبکه‌های حافظه کوتاه‌مدت بلندمدت – که معمولاً به آنها «LSTM» می‌گویند – نوع خاصی از RNN هستند که قادر به یادگیری وابستگی‌های بلندمدت هستند.

LSTM ها برای جلوگیری از مشکل وابستگی طولانی مدت طراحی شده اند. به خاطر سپردن اطلاعات برای مدت طولانی عملاً رفتار پیش فرض آنهاست، نه چیزی که برای یادگیری آن تلاش می کنند!

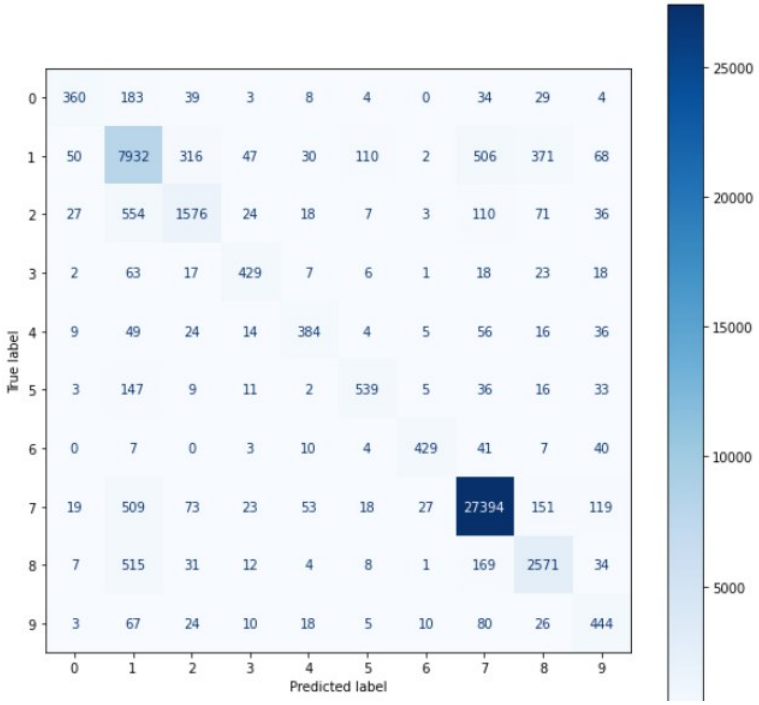
کد و نتایج

در تصویر زیر مدل تعریف شده است.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(layers.LSTM(32,dropout=0.2,return_sequences=True))
model.add(layers.LSTM(32,dropout=0.2,return_sequences=False))
model.add(layers.Dense(20, activation='softmax'))
```

شکل ۱۵.۲: LSTM model

در تصویر زیر می توان نتایج به دست آمده را در قالب نمودار مشاهده کرد.



شکل ۱۶.۲: LSTM confusion matrix

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است.

train acc: 0.9093

test acc:0.8859911561012268

۶.۴.۲ Transformer

مدل ترانسفورماتور NLP مکانیزم «اتنشن» را معرفی کرد که رابطه بین همه کلمات در جمله را در نظر می گیرد. وزن‌های دیفرانسیل ایجاد می‌کند که نشان می‌دهد کدام عناصر دیگر در جمله برای تفسیر یک کلمه مشکل مهم‌تر هستند. به این ترتیب عناصر مبهم می توانند به سرعت و کارآمد حل شوند.

کد و نتایج

نمونه ای از پیش پردازش لازم برای آماده سازی دیتا برای ترنسفورم در زیر قابل مشاهده است.

```
def cleanhtml(raw_html):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', raw_html)
    return cleantext


def cleaning(text):
    text = text.strip()

    cleaning regular #
    text = clean(text,
        fix_unicode=True,
        to_ascii=False,
        lower=True,
        no_line_breaks=True,
        no_urls=True,
        no_emails=True,
        no_phone_numbers=True,
        no_numbers=False,
        no_digits=False,
        no_currency_symbols=True,
        no_punct=False,
        replace_with_url="",
        replace_with_email="",
        replace_with_phone_number="",
        replace_with_number="",
        replace_with_digit="0",
        replace_with_currency_symbol="",
    )

    htmls cleaning #
    text = cleanhtml(text)

    normalizing #
```

```
    normalizer = hazm.Normalizer()
    text = normalizer.normalize(text)

    patterns wierd removing #
    wierd_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  emoticons #
        u"\U0001F300-\U0001F5FF"  pictographs & symbols #
        u"\U0001F680-\U0001F6FF"  symbols map & transport #
        u"\U0001F1E0-\U0001F1FF"  (iOS) flags #
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u'\U00010000-\U0010ffff'
        u"\u200d"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\u3030"
        u"\ufe0f"
        u"\u2069"
        u"\u2066"
        u"\u200c" #
        u"\u2068"
        u"\u2067"
        "]+", flags=re.UNICODE)

    text = wierd_pattern.sub(r'', text)

    hashtags ,spaces extra removing #
    text = re.sub("#", "", text)
    text = re.sub("\s+", " ", text)

    return text
```

در تصویر زیر روند آموزش را برای مدل میتوان مشاهده کرد.

شکل ۱۷.۲: Transformer model

همچنین دقت به دست آمده برای داده های آموزش و آزمایش به شرح زیر است

train acc: 0.892

test acc: 0.907

۵.۲ یافته ها و نتیجه‌گیری

از آنجا که تسک مربوط به تشخیص نوع خبر از دسته تسک هایی است که ذاتا داده های خیلی طولانی ندارد و جملات کوتاه اند احتمالا استفاده از مدل های خیلی عمیق ایده مناسبی نبوده و مدل های کلاسیک بهتر و به صرفه تر خواهند بود. به علاوه اثر پیش پردازش را نیز نمیتوان نادیده گرفت و هرچه این عمل بهتر صورت گیرد،نتایج بهتری خواهیم داشت.

model	train acc	test acc
Multinomial naïve bayes	0.7958	0.7902
Logistic Regression	0.8291	0.8087
Gradient Boosting	0.8523	0.8419
Neural Net with Ngram	0.9666	0.8811
Simple RNN	0.8960	0.8711
LSTM	0.9093	0.8859
Transformer	0.892	0.907