|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **بازیابی ویدیو براساس توصیف متنی صحنه‌ی ورودی**   |  |  |  | | --- | --- | --- | | ردیف | نام و نام‌خانوادگی | شماره‌ی دانشجویی | | 1 | سایین اعلا | 99400023 | | 2 | فرناز خوش دوست آزاد | ۹۹۵۲۱۲۵۳ |      |  | | --- | | موضوع پروژه: این پروژه به موضوع "ویدیو ریتریول" یا بازیابی ویدیو بر اساس توضیحات متنی (Captions) می‌پردازد. در این پروژه، هدف این است که با استفاده از توضیحات متنی داده‌شده (مانند کپشن‌های تصاویر و ویدیوها)، مشابه‌ترین ویدیوها به یک عبارت جستجو شده را پیدا کنیم. این مدل به طور خاص برای ویدیوهایی که همراه با توضیحات متنی هستند طراحی شده و از تکنیک‌های مدل‌های زبانی و تصویری (مانند CLIP) برای استخراج ویژگی‌های متنی و تصویری استفاده می‌کند.  مجموعه داده: در این پروژه از مجموعه داده‌ی Flickr8k و mini-cocoاستفاده شده است که شامل 8000 تصویر و 40,000 توضیح (کپشن) است. این مجموعه داده برای مدل‌های پردازش زبان طبیعی و دید کامپیوتری بسیار معروف است و اغلب در پروژه‌های مرتبط با بازیابی اطلاعات تصویر و ویدیو به کار می‌رود. این مجموعه داده به عنوان نقطه شروع برای آموزش مدل‌ها و تست آن‌ها بر روی تصاویر و ویدیوهای مختلف استفاده می‌شود.  در این نوتبوک از ابزارهای پردازش متن را برای ترجمه و پردازش داده‌های مربوط به تصاویر استفاده می‌کنیم. هدف اصلی این برنامه، پردازش و ترجمه توضیحات مرتبط با تصاویر و آماده‌سازی یک مجموعه داده بهینه است. پیش‌نیازها و وابستگی‌ها نصب کتابخانه‌های مود نیاز کد شامل استفاده از کتابخانه‌های مختلفی برای پردازش زبان طبیعی و ترجمه است. در ابتدا، کتابخانه‌های مورد نیاز نصب می‌شوند. این کتابخانه‌ها شامل موارد زیر هستند:  * nltk: برای پردازش زبان طبیعی * googletrans: برای ترجمه متن  وارد کردن کتابخانه‌ها توضیح کتابخانه‌ها  * os: برای کار با فایل‌ها و دایرکتوری‌ها. * json: برای پردازش فایل‌های JSON. * googletrans: برای ترجمه متون. * sentence\_transformers: برای ایجاد بردارهای جمله‌ای و محاسبه شباهت‌ها بین جملات.  تعریف مسیرهای فایل input\_file = './Dataset/flickr8k/captions.txt'  output\_file = './Dataset/flickr8k/caption\_fa.txt'  این متغیرها مسیر فایل‌های ورودی و خروجی را مشخص می‌کنند:   * input\_file: مسیر فایل ورودی که شامل توضیحات تصاویر است. * output\_file: مسیر فایل خروجی که توضیحات ترجمه‌شده در آن ذخیره می‌شوند. |      |  | | --- | | دلیل استفاده از پیش‌پردازش: پیش‌پردازش داده‌ها در پروژه‌های یادگیری ماشین اهمیت بالایی دارد چرا که می‌تواند دقت و کارایی مدل‌ها را به طور قابل توجهی بهبود دهد. در این پروژه، داده‌های متنی (کپشن‌ها) و تصویری (تصاویر و ویدیوها) هر دو نیاز به پیش‌پردازش دارند تا ویژگی‌های قابل استفاده برای مدل‌ها استخراج شوند. پیش‌پردازش صحیح می‌تواند منجر به عملکرد بهتر مدل‌های زبانی و تصویری شود.  روش‌های پیش‌پردازش:  ترجمه و نرمال‌سازی متون: برای پیش‌پردازش داده‌های متنی، از ترجمه خودکار استفاده شد تا تمام توضیحات به زبان فارسی تبدیل شوند. این کار به ما اجازه می‌دهد که از مدل‌های زبانی فارسی برای پردازش متون استفاده کنیم.  استخراج ویژگی‌های تصویری: تصاویر نیاز به تغییر اندازه دارند تا با ورودی‌های مدل‌های تصویری سازگار شوند. در این پروژه، تصاویر به اندازه 224x224 تغییر اندازه داده شدند.  سازگاری با مدل‌های CLIP: مدل‌های CLIP برای پردازش همزمان ویژگی‌های متنی و تصویری طراحی شده‌اند، بنابراین نیاز به تنظیم ورودی‌های خاص برای این مدل‌ها داریم که به صورت خودکار از طریق پیش‌پردازش انجام شد.  روش‌های استفاده شده:  برای داده‌های متنی، از ترجمه Google Translate برای ترجمه کپشن‌ها به زبان فارسی استفاده کردیم.  برای داده‌های تصویری، از تغییر اندازه تصاویر و نرمال‌سازی استفاده کردیم. |      |  | | --- | | مدل‌های موجود: برای این نوع از پروژه‌های بازیابی ویدیو بر اساس توضیحات متنی، مدل‌های مختلفی وجود دارند، از جمله:  CLIP (Contrastive Language-Image Pretraining): این مدل به طور خاص برای ارتباط میان متن و تصویر طراحی شده است.ViLT: یک مدل مبتنی بر Transformer برای پردازش همزمان متن و تصویر.VisualBERT: مدل مشابه دیگری برای پردازش متن و تصویر است.مدل انتخابی: ما از مدل CLIP (مخصوص زبان فارسی) استفاده کرده‌ایم. مدل CLIP برای پردازش همزمان تصاویر و متون به کار می‌رود و توانایی خوبی در درک ارتباط معنایی میان تصاویر و متون دارد. از آنجا که CLIP به خوبی برای این کار آموزش دیده و در پروژه‌های مشابه عملکرد عالی نشان داده است، این مدل به عنوان مدل اصلی انتخاب شد.  دلایل انتخاب CLIP: قابلیت پردازش همزمان تصاویر و متون. عملکرد بسیار خوب در بسیاری از مسائل بازیابی اطلاعات. استفاده از مدل‌های پیش‌آموزش دیده (مثل CLIP Vision برای تصاویر و CLIP Text برای متون).دسترسی به مدل‌های فارسی‌سازی شده مانند "SajjadAyoubi/clip-fa-vision" که توانایی پردازش داده‌های فارسی را دارد.  بارگذاری مدل پردازش زبان طبیعی:  model = SentenceTransformer('distiluse-base-multilingual-cased-v1')  مدل distiluse-base-multilingual-cased-v1 از sentence-transformers برای پردازش متن چندزبانه استفاده می‌شود که پیش از آن به سراغ bleu رفتیم، اما چون این مدل جدیدتر بود از آن استفاده کردیم.  تابع convert\_json\_to\_textکارکرد این تابع: این تابع فایل JSON شامل توضیحات تصاویر را به یک فایل متنی ساده تبدیل می‌کند. در هر خط از خروجی، نام تصویر و توضیح مربوط به آن ذخیره می‌شود.پارامترها:  json\_file: مسیر فایل JSON ورودی.output\_file: مسیر فایل متنی خروجی.تابع Preparing\_minicocoکارکرد این تابع:این تابع نام فایل‌های تصویری را در مجموعه داده MiniCOCO تغییر می‌دهد:  نام‌هایی که با COCO\_train2014\_ شروع می‌شوند، بخش ابتدایی حذف شده و صفرهای اضافه از اول نام حذف می‌شوند.  این کد برای پردازش و آماده‌سازی مجموعه داده‌های تصویری و توضیحات مربوط به آن‌ها طراحی شده است. این فرآیند شامل تبدیل داده‌های JSON به متن، تغییر نام فایل‌ها، و استفاده از مدل‌های یادگیری عمیق برای پردازش زبان است. این مستند می‌تواند به توسعه‌دهندگان کمک کند تا به درک بهتری از عملکرد کد برسند و تغییرات لازم را در آن اعمال کنند.  train\_fintune\_Clipfa::مستندات پروژه: آموزش و تنظیم دقیق CLIP در این پروژه، مدل CLIP برای پردازش تصاویر و توضیحات متنی مرتبط آموزش داده می‌شود. این برنامه شامل مراحل دانلود داده، پردازش تصاویر، آماده‌سازی داده‌ها و تنظیم دقیق مدل است تا عملکرد بهینه‌ای در استخراج ویژگی‌های تصویری و متنی داشته باشد.پیش‌نیازها و وابستگی‌ها نصب کتابخانه‌های مورد نیاز  !pip install -q git+https://github.com/sajjjadayobi/clipfa.git  کتابخانه clipfa که روی GitHub قرار دارد، برای پردازش مدل CLIP مورد نیاز است و از طریق این دستور نصب می‌شود.پس از وارد کردن کتابخانه‌ها که هر کدام از کاربردها در زیر نوشته شده است مراحل زیر را طی می کنیم.کاربرد کتابخانه‌ها:  torch: برای پیاده‌سازی مدل‌های یادگیری عمیق.  zipfile: جهت استخراج فایل‌های فشرده.  kagglehub: برای دریافت مجموعه داده‌ها از Kaggle.  numpy و pandas: جهت پردازش داده‌های عددی و جدولی.  PIL: برای کار با تصاویر.  tqdm: نمایش پیشرفت پردازش داده‌ها.  torch.nn: شامل توابع از دست دادن و لایه‌های عصبی.  sklearn.model\_selection: برای تقسیم داده‌ها به بخش‌های آموزشی و ارزیابی.  transformers: برای استفاده از مدل‌های پیش‌آموزش‌دیده مانند CLIP و Roberta. دریافت مجموعه داده Flickr8k  !kaggle datasets download -d aladdinpersson/flickr8kimagescaptions  with zipfile.ZipFile("flickr8kimagescaptions.zip", "r") as zip\_ref:  zip\_ref.extractall("flickr8k")  این بخش، مجموعه داده Flickr8k را از Kaggle دانلود کرده و آن را استخراج می‌کند. این مجموعه شامل تصاویر و توضیحات متنی مرتبط برای آموزش مدل است.  پردازش داده‌ها  csv\_file = "/kaggle/input/dataset/Captions.csv"  image\_dir = "/kaggle/input/flickr8kimagescaptions/flickr8k/images"  df = pd.read\_csv(csv\_file, delimiter=",", names=["Filename", "Caption"], skiprows=1)  df['Filename'] = df['Filename'].astype(str).str.strip()  df['Caption'] = df['Caption'].astype(str).str.strip()  این قطعه کد داده‌های مربوط به توضیحات تصاویر را از فایل CSV خوانده و پردازش می‌کند تا آماده استفاده در مدل CLIP شود.تعریف مجموعه داده برای PyTorchسپس کلاسی به اسم Flickr8kDataset تعریف کردیم که عملکرد آن در زیر آورده شده است.عملکرد این کلاس: داده‌های تصویری و متنی را خوانده و پردازش می‌کند. تصاویر را به بردارهای ویژگی تبدیل می‌کند.متن توضیحات را توکنایز کرده و آماده استفاده در مدل می‌کند.آماده‌سازی مدل و آموزش:  vision\_model = CLIPVisionModel.from\_pretrained("openai/clip-vit-base-patch32")  language\_model = RobertaModel.from\_pretrained("roberta-base")  tokenizer = AutoTokenizer.from\_pretrained("roberta-base")  feature\_extractor = CLIPFeatureExtractor.from\_pretrained("openai/clip-vit-base-patch32")  مدل‌های مورد استفاده:  CLIPVisionModel: برای پردازش تصاویر و استخراج ویژگی‌های آن‌ها.  RobertaModel: جهت پردازش متون و تبدیل آن‌ها به نمایش عددی.  AutoTokenizer: برای توکنایز کردن توضیحات متنی.  CLIPFeatureExtractor: برای پردازش و استخراج ویژگی‌های تصویری از ورودی‌های تصویری.  Clip\_retrieval\_2: پس از نصب کتابخانه های مورد نیاز مراحل زیر را طی می کنیم.مستندات پروژه: بازیابی و پردازش ویدیو  دانلود و استخراج مجموعه داده ویدیوdef download\_and\_extract\_videos():  print("Downloading video dataset...")  url = 'https://drive.google.com/uc?id=1GYdaAsxRCqBI-N8KP1fIYOGmS\_M3hxBW'  output = 'videos.zip'  gdown.download(url, output, quiet=False)  print("Extracting video dataset...")  with zipfile.ZipFile(output, 'r') as zip\_ref:  zip\_ref.extractall("videos")  print("Dataset is ready.")  این تابع مجموعه داده ویدیویی را از Google Drive دانلود کرده و آن را استخراج می‌کند.پردازش فریم‌های ویدیودر تابعextract\_frames به موارد زیر پرداختیم.عملکرد این تابع:باز کردن ویدیو با cv2.VideoCapture.استخراج فریم‌ها بر اساس نرخ فریم داده‌شده (frame\_rate).تبدیل فریم‌ها به تصویر RGB و ذخیره آن‌ها. تعریف مجموعه داده ویدیویی برای مدل CLIP در این قسمت نیز VideoDataset از این تابع استفاده کردیم.ویژگی‌های این کلاس:بارگذاری ویدیوها و استخراج فریم‌ها. استفاده از ویژگی‌های مدل CLIP برای پردازش تصاویر و ایجاد نمایش برداری.آماده‌سازی مدل و یادگیری برا این کار از مدل‌های زیر استفاده‌ می کنیم:  CLIPVisionModel: پردازش و استخراج ویژگی‌های تصویری.  AutoModel و AutoTokenizer: پردازش متون.  transforms.Compose: تغییر اندازه، تبدیل به Tensor و نرمال‌سازی داده‌ها. |      |  | | --- | | مرحله اول - آماده‌سازی داده‌ها:   1. ابتدا، داده‌ها از مجموعه Flickr8k خوانده شدند و پس از ترجمه به زبان فارسی، در قالب فایل متنی ذخیره شدند. 2. داده‌های تصویری به اندازه 224x224 تغییر اندازه داده شدند تا با ورودی‌های مدل CLIP سازگار باشند. 3. پس از آماده‌سازی داده‌ها، آن‌ها به فرمت CSV ذخیره شدند تا بتوان به راحتی به آن‌ها دسترسی داشت.   مرحله دوم - انتخاب و بارگذاری مدل:   1. مدل CLIP با استفاده از وزن‌های پیش‌آموزش دیده برای زبان فارسی بارگذاری شد. 2. مدل‌های Vision و Text از CLIP برای پردازش تصاویر و متون به طور همزمان انتخاب شدند.   مرحله سوم - آموزش و فاین‌تیونینگ مدل:   1. مدل‌های تصویری و متنی برای بهینه‌سازی عملکرد در بازیابی ویدیو بر اساس توضیحات متنی فاین‌تیون شدند. 2. از Cosine Embedding Loss برای آموزش استفاده شد تا ویژگی‌های استخراج شده از تصاویر و متون به هم نزدیک شوند. 3. پس از آموزش، مدل‌های فاین‌تیون شده ذخیره شدند تا در بخش‌های بعدی از آن‌ها استفاده شود.   مرحله چهارم - استخراج ویژگی‌های ویدیوها و جستجوی مشابه‌ترین ویدیوها:   1. ویژگی‌های ویدیوها با استخراج فریم‌ها از ویدیوها و استفاده از مدل CLIP استخراج شدند. 2. برای هر ویدیو، ویژگی‌های متنی (کپشن‌ها) استخراج شده و شباهت آن‌ها با ویژگی‌های ویدیوها محاسبه شد. 3. شباهت‌ها بر اساس Cosine Similarity محاسبه و ویدیوهای مشابه با هر جستجو ارائه شدند. |      |  | | --- | | معیارهای ارزیابی:  دقت مدل: برای ارزیابی دقت مدل، شباهت‌ها بین ویدیوها و توضیحات متنی محاسبه شد.  Cosine Similarity: برای اندازه‌گیری میزان شباهت بین ویژگی‌های استخراج شده از ویدیو و متن استفاده شد.  ارزیابی کیفی: ویدیوهایی که بیشترین شباهت را به توضیحات متنی داشتند، به عنوان خروجی مدل ارائه شدند.  نتایج مدل: برای دو نمونه آزمایشی، مدل توانست ویدیوهای مرتبط با توضیحات متنی را به دقت شبیه‌سازی کند. در یک مورد، مدلی که در تشخیص ویدیوهایی با توضیحات پیچیده‌تر دچار مشکل شده بود، اما در توضیحات ساده و مستقیم، دقت بالایی داشت. |      |  | | --- | | برای پروژه که Api و gui داشته باشد از فریم ورک Django استفاده شد که شمال back end و front-end باشد.  که برای فرانت فایل index.html است و همچنین api مخصوص بک برای پیدا کردن عکس مربوطه image\_search است |      |  | | --- | | https://www.kaggle.com/datasets/aladdinpersson/flickr8kimagescaptions  https://www.kaggle.com/datasets/nagasai524/mini-coco2014-dataset-for-image-captioning  <https://huggingface.co/datasets/AhmedSSabir/Textual-Image-Caption-Dataset>  <https://github.com/sajjjadayobi/CLIPfa>  https://storage.googleapis.com/openimages/web/index.html | |