دانشگاه سمنان

دانشکده مهندسی برق و کامپیوتر

درجه تحصیلی: کارشناسی

سیستم‌های چندرسانه‌ای

گردآورندگان:

محدثه مطلبی/ پرنیان شاکریان/ نجمه یزدی/ فاطمه کدخدایی فرد

استاد:

دکتر صفاری

سال تحصیلی: خرداد ۱۴۰۳

AI summarizer ابزار یا سیستمی است که از هوش مصنوعی برای فشرده‌سازی قطعات بزرگ متن به خلاصه‌های کوتاه‌تر استفاده می‌کند و در عین حال ایده‌های اصلی و اطلاعات کلیدی در آن حفظ می‌شود. معمولا AI summarizer از تکنیک‌های پردازش زبان طبیعی (NLP) برای درک زمینه، معناشناسی و اهمیت متن برای ایجاد خلاصه‌های منسجم و مختصر استفاده می‌کند. دو نوع اصلی از تکنیک‌های خلاصه‌سازی مورد استفاده در این هوش مصنوعی وجود دارد:

۱. Extractive Summarization: این روش شامل انتخاب و استخراج مهمترین جملات یا عبارات به طور مستقیم از متن اصلی است. در این روش، خلاصه کننده نکات کلیدی را شناسایی و آن‌ها را جمع آوری می‌کند تا تشکیل یک متن خلاصه دهد. این روش جملات جدیدی تولید نمی‌کند، بلکه جملات موجود را در کنار هم قرار می‌دهد.
۲. Abstractive Summarization: برخلاف خلاصه سازی Extractive، این روش جملات جدیدی تولید کرده که ماهیت متن اصلی را به تصویر می‌کشد و شامل درک محتوا و بیان مجدد آن به شکل کوتاه‌تر است که اغلب منجر به خلاصه‌های طبیعی و منسجم‌تر می‌شود.

خلاصه‌کننده‌های هوش مصنوعی را می‌توان در کاربردهای مختلفی استفاده کرد، از جمله:

- جمع‌آوری اخبار: جمع‌بندی مقالات خبری برای ارائه مروری سریع.
- تحقیقات دانشگاهی: ایجاد چکیده برای مقالات پژوهشی.
- مدیریت محتوا: خلاصه کردن گزارش‌ها، ایمیل‌ها یا اسناد طولانی برای خواندن سریع.

طی این پروژه ما با استفاده از یکی از مدل‌های زبانی موجود سامانه‌ای رو طراحی خواهیم کرد که بتواند بر اساس یک کتاب به سوالاتی پرسیده شده پاسخ مناسبی دهد. در نتیجه برای رسیدن به این مرحله سامانه باید بتواند سه عمل مهم را انجام دهد. این مراحل شامل موارد زیر که پیکر بندی اصلی پروژه ما را تشکیل می‌دهد می‌باشد:

گام ۱: انتخاب یک مدل زبان

۱. microsoft/phi: این مدل برای پردازش زبان طراحی شده است و به دلیل آموزش در مجموعه داده های متنوع، برای انواع پرسش و پاسخ مناسب است. به همین دلیل ما microsoft/phi را برای تعادل بین عملکرد و منابع مورد نیاز انتخاب می‌کنم، زیرا برای انواع وظایف NLP از جمله پاسخگویی به سؤالات بهینه شده است.
۲. TinyLlama: این مدل کوچکتر است اما از نظر منابع محاسباتی کارآمدتر می‌باشد.
۳. Google gemma: این یک مدل نسبتاً بزرگ است که می‌تواند عملکرد بهتری ارائه دهد اما ممکن است به منابع بیشتری نیاز داشته باشد.

گام ۲: طراحی معماری سیستم

این سیستم دارای اجزای زیر خواهد بود:

۱. Text Ingestion: ماژولی برای ورودی و ذخیره داده‌های متنی.
۲. Question Processing: ماژولی برای رسیدگی به سؤالات کاربر و آماده سازی آن‌ها برای مدل از جمله حذف نویز، حذف اعداد از متن، حذف کلمات توقف، ریشه یابی، توکن سازی.
۳. Answer Retrieval: ماژولی برای استفاده از مدل زبانبه منظور یافتن و برگرداندن قسمت مربوطه از متن یا ایجاد پاسخ بر اساس متن.
۴. Summarization: ماژولی برای خلاصه کردن متن برای پردازش آسان تر.

گام ۳: پیاده سازی روش های خلاصه سازی و پاسخگویی به سؤال

ما از پایتون و کتابخانه‌های مناسب (مانند ترانسفورماتورهای Hugging Face) برای پیاده سازی سیستم استفاده خواهیم کرد. در اینجا یک طرح کلی از پیاده سازی است:

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer,
pipeline

# Load the model and tokenizer
model_name = "microsoft/phi-2"
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

لاین اول transformers کلاس‌ها و توابع خاصی را از کتابخانه Hugging Face وارد می‌کند، که یک کتابخانه محبوب برای کار با مدل‌های زبان از پیش آموزش‌دیده است. این کتابخانه دسترسی به چندین مدل را برای وظایف مختلف پردازش زبان طبیعی (NLP) فراهم کرده و از معماری‌های مختلف مانند BERT، GPT، T5 و بسیاری دیگر پشتیبانی می‌کند. این کتابخانه فرآیند بارگیری مدل‌ها و استفاده از آن‌ها را برای کارهایی مانند طبقه‌بندی متن، خلاصه‌سازی، ترجمه و پاسخ‌گویی به سؤالات ساده می‌کند.

- AutoModelForQuestionAnswering کلاسی است که روشی ساده برای بارگذاری هر مدل از پیش آموزش دیده که به طور خاص برای وظیفه پاسخگویی به سؤال طراحی شده است، ارائه می‌دهد. جزئیات معماری و پیکربندی مدل را انتزاعی می‌کند و به ما این امکان را می‌دهد که یک مدل را فقط با یک خط کد بارگذاری کنیم.
- AutoTokenizer کلاسی است که به طور خودکار tokenize مناسب را برای یک مدل مشخص انتخاب می‌کند. tokenizerها در NLP بسیار مهم هستند زیرا متن را به قالب عددی مورد نیاز مدل‌ها تبدیل می‌کنند.
- pipeline یک API سطح بالا است که پیچیدگی استفاده از مدل‌های مختلف برای وظایف مختلف NLP را انتزاعی کرده و کاربری آسان برای انجام وظایفی مانند پاسخگویی به سؤال، تولید متن، ترجمه و موارد دیگر ارائه می دهد.

```
from transformers import BartForConditionalGeneration, BartTokenizer

# Load summarization model
summarization_model_name = " The History of the Philosophy of Mind"
summarization_model =
BartForConditionalGeneration.from_pretrained(summarization_model_name)
summarization_tokenizer =
BartTokenizer.from_pretrained(summarization_model_name)
```

لاین دوم transformers دو جزء BartForConditionalGeneration و BartTokenizer را از کتابخانه Hugging Face وارد می‌کند. این مؤلفه ها به طور خاص برای وظایف تولید خلاصه سازی، با استفاده از مدل BART استفاده می شوند. BART یک مدل توالی به دنباله قدرتمند است که عناصر BERT و GPT را ترکیب می‌کند و به ویژه برای کارهایی که شامل تولید متن هستند، مانند خلاصه‌سازی و ترجمه است.

- BartForConditionalGeneration کلاسی است که مدل BART را به طور خاص برای وظایف تولید شرطی پیاده سازی می‌کند. این وظایف شامل خلاصه سازی متن می‌باشد.
- BartTokenizer کلاسی است که tokenizer مرتبط با مدل BART را ارائه می‌دهد. همان طور که گفتیم tokenizerها در NLP ضروری هستند زیرا متن را به توکن های عددی تبدیل می‌کنند که مدل بتواند آن را درک کند.

مدل BERT به صورت دوطرفه (bidirectional) آموزش دیده است. این مدل برخلاف مدل‌های قبلی، به جای پردازش متون به صورت ترتیبی (از چپ به راست یا راست به چپ)، متن را به صورت کامل و همزمان از دو طرف پردازش می‌کند. این ویژگی به BERT اجازه می‌دهد تا زمینه و معنای دقیق‌تری از کلمات در متن استخراج کند. دو وظیفه اصلی BERT که در آن آموزش دیده است شامل موارد زیر می‌شود:

۱. در روش اول (MLM) برخی از کلمات در متن با توکن [MASK] جایگزین شده و مدل باید کلمه اصلی را پیش‌بینی کند.
۲. در روش دوم (NSP) دو جمله متوالی به مدل داده می‌شود و مدل باید تشخیص دهد که آیا جمله دوم به دنبال جمله اول می‌آید یا خیر.

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load pre-trained T5 model and tokenizer
model = T5ForConditionalGeneration.from_pretrained('t5-base')
tokenizer = T5Tokenizer.from_pretrained('t5-base')

input_text = "Your input text goes here."
input_ids = tokenizer.encode(input_text, return_tensors='pt')

# Generate a summary of the input text
summary_ids = model.generate(input_ids, max_length=150, num_beams=2,
repetition_penalty=2.5, length_penalty=1.0, early_stopping=True)
summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

print(summary)
```

در کد، متن ورودی کتاب را در متغیر input_text قرار داده و سپس با استفاده از مدل T5 و tokenizer آن را به یک خلاصه تبدیل و در نهایت، خلاصه شده را چاپ می‌کند.