به نام خالق رنگین کمان

گزارش ورکشاپ هضم – ستاره باباجانی

سوال 1: توسعه یک محصول تشخیص خودکار گفتار (ASR) شامل چندین مرحله است، از برنامه ریزی اولیه تا استقرار و بهبود مستمر. در اینجا به بررسی هر مرحله میپردازیم:

1. برنامه ریزی و تجزیه و تحلیل نیازمندی:
   - اهداف: اهداف سیستم ASR مانند target languages (به عنوان مثال فارسی)، موارد استفاده (مانند رونویسی، دستورات صوتی) و معیارهای عملکرد را به وضوح باید مشخص کرد.
   - تحقیق بازار: راه حل های موجود ASR را تجزیه و تحلیل کرد، نیازهای مخاطب هدف را درک کرده و ارزش های منحصر به فرد را شناسایی کنیم.
2. جمع آوری داده ها و پیش پردازش:
   - جمع آوری داده ها: مجموعه داده بزرگی از ضبط های صوتی و transcriptions مربوط به آنها را به زبان فارسی جمع آوری کنیم. این ممکن است شامل مجموعه داده های عمومی، ضبط های اختصاصی یا crowdsourcing باشد.

- پیش پردازش داده: داده های صوتی را تمیز و پیش پردازش کنیم. این شامل کاهش نویز، normalization و تقسیم صدا به قطعات قابل مدیریت است. داده های متنی باید tokenized، normalized و برای آموزش annotate شوند.

3. توسعه:

- استخراج ویژگی: استخراج ویژگی های مرتبط از داده های صوتی. ویژگی های مشترک شامل Mel-frequency cepstral coefficients (MFCCs)و طیف نگاری است.
- انتخاب مدل: مدل های یادگیری ماشین مناسب را انتخاب کنیم. برای ASR، اغلب ترکیبی از مدل‌های یادگیری عمیق مانند شبکه‌های عصبی کانولوشن(CNN) ، شبکه‌های عصبی بازگشتی (RNN) یا ترنسفورمرها بهتر است.
- آموزش: مدل ها را با استفاده از داده های از پیش پردازش شده آموزش میدهیم. که شامل تقسیم داده‌ها به مجموعه‌های آموزشی، اعتبارسنجی و تست و تنظیم دقیق ابرپارامترها برای بهینه‌سازی عملکرد میشود.

4. ادغام با Hazm:

- پیش پردازش متن: برای پیش پردازش متن فارسی از Hazm استفاده کنیم. این شامل tokenization، stemming،

lemmatization و part-of-speech tagging برای آماده‌سازی transcriptions است.

- مدل سازی زبان: با استفاده از ابزار Hazm یک مدل زبان برای فارسی ایجاد کرده تا دقت سیستم ASR را با پیش بینی احتمال توالی کلمات بهبود یابد.

5. آزمایش و ارزشیابی:

- معیارهای عملکرد: سیستم ASR را با استفاده از معیارهایی مانند نرخ خطای کلمه(WER) ، نرخ خطای کاراکتر (CER) و دقت ارزیابی میکنیم.
- تست کاربر: آزمایش کاربر را برای جمع آوری بازخورد و شناسایی زمینه های بهبود انجام میدهیم. این می تواند شامل سناریوهای دنیای واقعی باشد که در آن سیستم ASR توسط کاربران نهایی استفاده می شود.

6. Deployment:

- انتخاب پلتفرم: بستری را برای Deployment انتخاب میکنیم. مانند cloud-based services، برنامه های کاربردی تلفن همراه یا embedded systems.
- مقیاس پذیری: اطمینان حاصل میکنیم که سیستم می تواند loads مختلف را تحمل کند و عملکرد را حفظ کند. این

ممکن است شامل استفاده از distributed computing و زیرساخت cloud باشد.

7. نگهداری و بهبود:

- Monitoring: به طور مداوم بر عملکرد سیستم ASR در استفاده در دنیای واقعی نظارت میکنیم تا مشکلات شناسایی و برطرف شوند.
- به روز رسانی: به طور منظم سیستم را با داده های جدید به روز کرده و مدل ها را برای بهبود دقت و انطباق با نیازهای متغیر کاربر بازآموزی می کنیم.
- بازخورد کاربر: مکانیزمی را برای کاربران به منظور ارائه بازخورد اجرا کرده و از این بازخورد برای ایجاد پیشرفت های بیشتر استفاده میکنیم.

8. ملاحظات اخلاقی و حقوقی:

- حریم خصوصی: اطمینان حاصل میکنیم که سیستم با قوانین و مقررات حفظ حریم خصوصی در مورد استفاده و ذخیره داده های صوتی مطابقت دارد.
- سوگیری و Fairness: سوگیری های احتمالی را در مجموعه داده و مدل بررسی کرده تا از شناسایی منصفانه و بی طرفانه در جمعیت های مختلف کاربر اطمینان حاصل کنیم.

سوال 2: برای انتخاب 16 خروجی از تعداد زیادی خروجی تولید شده توسط چندین مدل باید از روش هایی استفاده کنیم که بتواند به طور موثر بهترین ها را رتبه بندی و ادغام کند. روش های پیشنهادی به شرح زیر هستند:

1. Ensemble Voting: این روش خروجی های مدل های مختلف را جمع می کند تا محتمل ترین transcription صحیح را تعیین کند. خروجی هر مدل به عنوان یک رای در نظر گرفته می شود و بیشترین خروجی ها در بین مدل ها انتخاب می شوند. مراحل آن به شرح زیر است:
    1. جمع آوری خروجی ها: 16 خروجی برتر از هر مدل را جمع آوری میکنیم.
    2. شمارش فرکانس: فرکانس هر خروجی منحصر به فرد را در همه مدل ها میشماریم.
    3. رتبه بر اساس فرکانس: خروجی ها را بر اساس تعداد فرکانس آنها رتبه بندی میکنیم.
    4. انتخاب 16 بالا: 16 خروجی برتر با بالاترین فرکانس را انتخاب میکنیم.

این روش نقاط قوت مدل های مختلف را ترکیب می کند که به طور بالقوه منجر به نتایج دقیق تری می شود و همینطور پیاده سازی و درک آن آسان است. اما ممکن است این کار به نفع خروجی های رایج تر باشد.

2. Confidence Scoring and Weighted Averaging: این روش امتیازهای اطمینان را به خروجی هر مدل اختصاص می دهد و از این امتیازها برای وزن دادن به اهمیت هر خروجی در فرآیند انتخاب نهایی استفاده می کند. مراحل آن به شرح زیر است:

1. تخصیص امتیازات اطمینان: هر مدل یک امتیاز اطمینان برای خروجی های خود ارائه می دهد.
2. رتبه بندی وزن دار: هر خروجی را در امتیاز اطمینان آن ضرب میکنیم.
3. نمرات مجموع: امتیازهای وزنی را برای هر خروجی منحصر به فرد در بین مدل ها جمع میکنیم.
4. رتبه بر اساس نمره کل: خروجی ها را بر اساس امتیازات جمع آوری شده رتبه بندی میکنیم.
5. انتخاب 16 برتر: 16 خروجی برتر را با بالاترین امتیازات کل انتخاب میکنیم.

این روش از اطلاعات اضافی (نمرات اطمینان) برای بهبود دقت انتخاب استفاده می کند و بین مدل ها بر اساس سطح اطمینان آنها تعادل برقرار می کند. هرچند اجرای پیچیده‌تری به دلیل نیاز به امتیازدهی و وزن دهی اطمینان دارد.

3. Context-Aware Ranking with Language Model Integration: در این روش خروجی ها را با یک مدل زبان ادغام می

کنیم تا اطمینان حاصل شود که خروجی های انتخاب شده نه تنها دقیق هستند، بلکه از نظر متنی مرتبط و منسجم در فارسی هستند. مراحل آن به شرح زیر است:

1. جمع آوری خروجی ها: 16 خروجی برتر از هر مدل را جمع آوری میکنیم.
2. امتیاز مدل زبان: از یک مدل زبان فارسی از پیش آموزش دیده (مثلاً یک مدل مبتنی بر ترنسفورمرها) برای امتیازدهی به هر خروجی بر اساس انسجام متنی و احتمال آن استفاده کنیم.
3. ترکیب وزن دار: نمرات مدل زبان را با امتیازات اطمینان خاص مدل (در صورت وجود) ترکیب کنیم.
4. رتبه بر اساس امتیاز ترکیبی: خروجی ها را بر اساس امتیازات ترکیبی آنها رتبه بندی میکنیم.
5. انتخاب 16 برتر: 16 خروجی برتر را با بالاترین امتیازات ترکیبی انتخاب میکنیم.

این روش اطمینان حاصل می کند که خروجی های انتخاب شده نه تنها دقیق هستند، بلکه از نظر زمینه نیز مناسب هستند علاوه بر اینکه با استفاده از یک مدل زبان، کیفیت کلی رونوشت های انتخاب شده را افزایش می دهد. هرچند که به یک مدل زبان آموزش دیده و منابع محاسباتی اضافی نیاز دارد.

سوال 3:

1. نرمال ساز: نرمال ساز متن را به فرم استاندارد یا متعارف تبدیل می کند. این تابع با انجام عملیاتی مانند lowercasing، حذف diacritics، normalizing punctuation و گسترش contractions، یکنواختی متن را تضمین می کند. این تابع برای کاهش تغییرات در متن‌هایی که معنی یکسانی دارند بسیار مهم است.
2. Formalizer: این تابع متن غیر رسمی یا محاوره ای را به نسخه رسمی تر تبدیل می کند. اغلب در زمینه هایی استفاده می شود که متن از رسانه های اجتماعی، پیام های چت یا سایر منابع غیررسمی باید به شکل استانداردتری پردازش شوند. این می تواند شامل تصحیح عامیانه، اختصارات و ساختارهای غیررسمی زبان باشد.
3. Lemmatizer: یک Lemmatizer کلمات را به شکل پایه یا فرهنگ لغت خود کاهش می دهد که به عنوان lemma شناخته می شود. بر خلاف stemming، که ممکن است به سادگی suffixes را حذف کند، context را در نظر می گیرد و کلمات را به اشکال پایه معنادار کاهش می دهد. مثلاً «running» به «run» و «better» به «good» تبدیل می شود.
4. Stemmer: یک stemmer با حذف پسوندها و پیشوندها، کلمات را به شکل ریشه آنها کاهش می دهد. شکل تهاجمی تری از نرمال سازی را در مقایسه با lemmatization انجام می دهد که اغلب به ریشه

های غیر دیکشنری می انجامد. برای مثال، «running» ممکن است به «run» و «jumps» به «jump» تبدیل شود.

5. Chunker: یک chunker متن را به بخش های مرتبط نحوی مانند عبارات اسمی یا عبارات فعل تقسیم می کند. به شناسایی و گروه بندی کلماتی که با هم در یک جمله عمل می کنند کمک می کند. به عنوان مثال، در جمله « The quick brown fox jumps over the lazy dog»، chunker ممکن است «The quick brown fox» را به عنوان یک عبارت اسمی شناسایی کند.
6. Tagger: به کلمات یا نشانه های یک متن برچسب اختصاص می دهد که اغلب بخشی از گفتار یا سایر نقش های نحوی را نشان می دهد. برای حاشیه نویسی متن با metadata، مانند شناسایی قسمت های گفتار (POS) برای هر کلمه استفاده می شود. این برای تجزیه نحوی و سایر تحلیل های زبانی اساسی است.
7. POSTagger (Part-of-Speech Tagger): یک POSTagger به طور خاص تگ های part-of-speech را به هر کلمه در یک متن اختصاص می دهد. grammatical category هر کلمه مانند اسم، فعل، صفت و غیره را مشخص می کند. این اطلاعات برای درک ساختار جمله و کارهایی مانند تجزیه و شناسایی موجودیت نامگذاری شده بسیار مهم است.

8. Parser: ساختار دستوری یک جمله را تجزیه و تحلیل می کند. جمله را به اجزاء تشکیل دهنده آن، مانند عبارات و بندها، تقسیم می کند و بین آنها رابطه برقرار می کند. این برای درک معنای نحوی و گاه معنایی جملات مهم است.
9. Embedder: یک embedder فراتر از کلمات تعمیم می دهد تا embeddings را برای انواع مختلف ورودی ها (کلمات، جملات، تصاویر و غیره) شامل شود. انواع مختلف داده ها را به نمایش های برداری تبدیل می کند. به عنوان مثال، embeddings جمله مانند BERT یا sentence transformers ، کل جملات را در بردارها رمزگذاری می‌کنند و context و معنای بیشتری را نسبت به word embeddings جداگانه به تصویر می‌کشند.
10. WordEmbedder: کلمات را به نمایش های برداری پیوسته تبدیل می کند. کلمات را به بردارهای عددی تبدیل می کند که معانی و روابط معنایی را در بر می گیرد. Word embeddings مانند Word2Vec، GloVe یا FastText به مدل‌ها اجازه می‌دهد تا شباهت‌های کلمه و کاربرد متنی را درک کنند.