سوال اول

۱. تحلیل نیازمندی‌ها و تعیین اهداف

- تعیین کاربران هدف: شناسایی کاربران اصلی و نیازهای خاص آنها.
- تعریف موارد استفاده: مشخص کردن موارد دقیق مورد استفاده از محصول ASR (مثل تبدیل گفتار به متن، فرمان‌های صوتی و غیره).
- تعریف معیارهای عملکرد: تعیین معیارهای کلیدی عملکرد مانند دقت، سرعت، و میزان تأخیر مجاز.

۲. جمع‌آوری و آماده‌سازی داده‌ها

- جمع‌آوری داده‌های گفتاری: جمع‌آوری داده‌های صوتی مرتبط با زبان و گویش‌های مختلف.
- حاشیه‌نویسی داده‌ها: ترانسکرایب کردن داده‌های صوتی به متن برای استفاده در آموزش مدل.
- تنوع داده‌ها: اطمینان از وجود تنوع در صداها، لهجه‌ها، سنین، جنسیت‌ها و محیط‌های مختلف.

۳. انتخاب و آموزش مدل‌های ASR

- انتخاب الگوریتم: انتخاب الگوریتم مناسب برای تشخیص گفتار (مثل مدل‌های شبکه عصبی عمیق، مدل‌های HMM و غیره).
- آموزش مدل: آموزش مدل با استفاده از داده‌های حاشیه‌نویسی شده.
- تنظیمات بهینه‌سازی: تنظیم هایپرپارامترهای مدل برای بهینه‌سازی دقت و عملکرد.

۴. ارزیابی و تست مدل

- تست با داده‌های جدید: ارزیابی مدل با استفاده از داده‌هایی که در فرآیند آموزش استفاده نشده‌اند.
- معیارهای ارزیابی: استفاده از معیارهایی مانند WER (Word Error Rate) برای اندازه‌گیری عملکرد مدل.
- بازخورد از کاربران: جمع‌آوری بازخورد از کاربران نهایی برای ارزیابی عملکرد مدل در شرایط واقعی.

۵. پیاده‌سازی و استقرار

- انتخاب زیرساخت: انتخاب زیرساخت مناسب برای استقرار مدل (مثل سرورهای ابری، دستگاه‌های موبایل و غیره).
- یکپارچه‌سازی با سیستم‌های موجود: اطمینان از یکپارچه‌سازی بدون مشکل مدل ASR با سیستم‌ها و برنامه‌های موجود.
- پایش و بهینه‌سازی: پایش مستمر عملکرد مدل و انجام بهینه‌سازی‌های لازم بر اساس بازخوردها و داده‌های جدید.

۶. نگهداری و به‌روزرسانی

- جمع‌آوری داده‌های جدید: ادامه جمع‌آوری داده‌های جدید برای به‌روزرسانی و بهبود مدل.
- پایش عملکرد: پایش مداوم عملکرد مدل در شرایط واقعی و شناسایی مشکلات احتمالی.
- به‌روزرسانی دوره‌ای: به‌روزرسانی مدل‌ها و الگوریتم‌ها بر اساس داده‌ها و تکنولوژی‌های جدید.

سوال دوم

۱. روش رای‌گیری وزن‌دار

در این روش، به هر مدل بر اساس عملکرد کلی آن وزنی اختصاص می‌دهیم و خروجی‌های آنها را بر اساس این وزن‌ها ترکیب می‌کنیم.

مراحل:

- اختصاص وزن‌ها: به هر مدل i، وزنی wi را بر اساس معیارهای عملکرد آن مانند دقت یا صحت بر روی یک مجموعه داده اعتبارسنجی اختصاص می‌دهیم. برای این منظور می‌توان به مدلی که بر روی دیتاست فارسی تمرین داده شده است، وزن بیشتری اختصاص داد تا احتمال انتخاب خروجی‌های آن در نهایت بیشتر شود.
- محاسبه نمرات وزن‌دار: برای هر خروجی j از مدل i، یک نمره وزن‌دار محاسبه می‌کنیم. اگر $p_{i,j}$ احتمال یا نمره اطمینان خروجی j از مدل i باشد، آنگاه نمره وزن‌دار $S_{i,j}$ عبارت است از:

$$S_{i,j} = w_i \times p_{i,j}$$

- تجمیع نمرات: نمرات را برای هر رونویسی منحصربفرد در میان تمام مدل‌ها با هم جمع می‌کنیم. اگر چندین مدل همان رونویسی (Transcript) را تولید کنند، نمرات وزن‌دار آنها را جمع خواهیم کرد.
- انتخاب خروجی‌های برتر: تمام رونویسی‌های منحصربفرد را بر اساس نمرات تجمیع شده آنها به ترتیب نزولی مرتب کرده و ۱۶ مورد برتر را انتخاب می‌کنیم.

مثال:

مدل A با وزن ۰.۶، مدل B با وزن ۰.۴

- خروجی ۱ از مدل

$$A: p\_A,1 = 0.9، S\_A,1 = 0.6 \times 0.9 = 0.54$$

- خروجی ۱ از مدل

$$B: p\_B,1 = 0.8، S\_B,1 = 0.4 \times 0.8 = 0.32$$

- اگر هر دو مدل برای خروجی ۱ همان رونویسی را داشته باشند، نمره ترکیبی برابر است با

$$0.54 + 0.32 = 0.86$$

۲. روش تجمیع رتبه

این روش رتبه‌هایی را که توسط مدل‌های مختلف اختصاص داده شده است، تجمیع می‌کند تا یک لیست رتبه‌بندی شده نهایی ایجاد کند.

مراحل:

- رتبه‌بندی خروجی‌ها: هر مدل ۱۶ خروجی را از ۱ (اولویت بالاترین) تا ۱۶ (اولویت پایین‌ترین) رتبه‌بندی می‌کند.

- نرمالیزه کردن رتبه‌ها: در صورت لزوم، رتبه‌ها را به یک مقیاس مشترک نرمالیزه می‌کنیم. برای سادگی، فرض می‌کنیم که رتبه‌های مستقیم (۱ تا ۱۶) را داریم.
- تجمیع رتبه‌ها: از یک روش تجمیع رتبه مانند شمارش بوردا یا رتبه میانه استفاده می‌کنیم. در شمارش بوردا، برای هر رونویسی، رتبه‌ها را در میان تمام مدل‌ها جمع می‌کنیم:

$$R_{final(t)} = \sum \left(i = 1 \text{ تا } M \right) rank_i(t)$$

- که در آن M تعداد مدل‌ها است و $\text{rank}_i(t)$ رتبه رونویسی t توسط مدل i می‌باشد.
- انتخاب خروجی‌های برتر: رونویسی‌ها را بر اساس رتبه‌های تجمیع شده آنها به ترتیب صعودی (مجموع رتبه پایین‌تر نشان‌دهنده اولویت بالاتر است) مرتب کرده و ۱۶ مورد برتر را انتخاب می‌کنیم.

مثال:

- مدل A رونویسی "hello" را در رتبه ۱ قرار می‌دهد، مدل B آن را در رتبه ۲ قرار می‌دهد:

$$R_{final}("hello") = 1 + 2 = 3$$

- بر اساس R_{final} مرتب کرده و ۱۶ مورد برتر را انتخاب می‌کنیم.

سوال سوم

۱. Normalizer

در Normalizer ما به دنبال این هستیم که یک متن ورودی را دریافت کنیم و آن را تا حد امکان ساده و کوتاه، و همچنین تا حد امکان لغات موجود در متون مختلف را یکسان‌سازی کنیم. این تابع یک تابع مهم در زمینهٔ پیش‌پردازش متن بوده که با کمک آن پردازش متن ساده‌تر و راحت‌تر می‌شود. عملیات موجود در این تابع در زبان فارسی به شرح زیر هستند:

- حذف علائم نگارشی
- حذف لغات اضافه و بسیار رایج نظیر «و»، «اما» و امثالهم.
- اضافه کردن نیم‌فاصله در لغات و یکسان‌سازی شکل‌های مختلف لغاتی که فاصله میانشان قرار گرفته است. نظیر تبدیل «می توانم» به «می‌توانم».
- حذف علائم خاص و نیز اعداد و ارقام با توجه به فیلد مورد بررسی
- اصلاح لغاتی که اشتباه املایی دارند.
- استانداردسازی نمایش اعداد، تاریخ، قیمت و ...

۲. Formalizer

در این تابع ما به دنبال این هستیم که کلماتی که به طور عامیانه مورد استفاده قرار می‌گیرند را به فرم رسمی‌شان تبدیل کنیم. در نتیجه در کاربردهایی که به طور ویژه با متون نوشته شده توسط کاربران چه به صورت کامنت و چه به صورت پست‌های موجود در فضای مجازی قرار داده شده‌اند، استفاده می‌شوند.

- تبدیل لغات عامیانه به شکل رسمی‌شان
- مواجهه با کلمات اختصاریافته
- بهبود دستورزبان استفاده شده در متن

۳. Lemmatizer

در این تابع به دنبال این هستیم که یک فعل یا لغت را به واژهٔ ریشه‌ای آن بازگردانیم. برای این منظور از یک لیست آماده یا مدل از پیش آموزش دیده استفاده می‌شود تا بتوان یک لغت را به ریشه‌اش بازگرداند. به عنوان مثال تبدیل گفته_شده_است به گفت#گو.

۴. Stemmer

در این تابع به دنبال این هستیم که یک فعل یا لغت را به واژهٔ ریشه‌ای آن بازگردانیم. تفاوت این تابع با تابع پیشین در این است که در این تابع مرحله به مرحله بخش‌های اضافی لغت را از آن حذف می‌کنیم تا آنکه به لغت ریشه‌ای بازگردانده شود. به عنوان مثال تبدیل کتابی به کتاب. استفاده از این روش، باعث بهبود سرعت اما کاهش کیفیت نتیجهٔ نهایی خواهد شد.

۵. Chunker

در این تابع متن ورودی را به بخش‌ها و عبارات سازندهٔ آن، نظیر عبارت اسمی یا عبارت فعلی تبدیل می‌کنیم. بنابراین نتیجهٔ نهایی این بخش، ساختار گرامری متن ورودی خواهد بود که در آن متن به چندین گروه تبدیل شده‌اند و هر یک

تگ مخصوص به خود را نیز خواهد گرفت. گروه‌های نهایی‌ای که ایجاد می‌شوند با یکدیگر همپوشانی نخواهند داشت و جدا از هم خواهند بود.

۶. Tagger

در این تابع، به هر یک از لغات موجود در متن، یک برچسب را نسبت می‌دهیم. این برچسب می‌تواند مربوط به وظایف گوناگون، نظیر NER و POS باشد.

۷. POSTagger

در این تابع، به طور ویژه هر لغت را بررسی می‌کنیم و به هر یک برچسب POS را نسبت می‌دهیم.

۸. Embedder

یک Embedder یک متن، جمله یا دنباله‌ای از لغات را به عنوان ورودی دریافت می‌کند و هر یک را به یک بردار با طول مشخص تبدیل می‌کند. این خروجی که با عنوان Embedding Vector شناخته می‌شود می‌تواند برای عملیات‌های Down-Stream مورد استفاده قرار گیرد.

۹. WordEmbedder

در این تابع، به ازای هر لغت موجود در متن ورودی، بردار Embedding متناظر با آن را خروجی می‌دهیم.

۱۰. Parser

در این تابع، ساختار دستور زبانی متن را به طور ویژه مورد بررسی قرار می‌دهیم. به طوری که در انتها یک درخت برای جملهٔ ورودی ساخته می‌شود که در این درخت جملهٔ ورودی به عنوان ریشه قرار داده می‌شود و در ادامه هر یک از بخش‌های مربوط به فعل و اسم و نیز سایر نقش‌ها، قرار داده می‌شوند. در پایان هر بخش مربوط به هر یک از عبارات قیدشده، به کلمات موجود در جمله تجزیه می‌شوند و نقش هر یک به طور جداگانه آورده می‌شوند. یکی از انواع Parserها، Dependency Parserها هستند که ارتباط میان واژگان موجود در یک متن را مشخص می‌کنند.