УДК 004.357

УДОСКОНАЛЕННЯ АКУСТИЧНОЇ МОДЕЛІ ЯДРА CMU ДЛЯ РОБОТИ З УКРАЇНСЬКОЮ МОВОЮ

Тапанар О.О.[1],Крепич С.Я.[2]
Тернопільський національний економічний університет
[1]магістрант; [2]к.т.н., старший викладач

I. Постановка проблеми

З кожним роком розробка програмного забезпечення для мобільних платформ стає все більш актуальною. Завдяки зростанню потужності апаратного забезпечення, мобільні пристрої зараз здатні ефективно виконувати значно більший ряд задач ніж 10 років тому. Зараз мобільні пристрої перетворились на повноцінних універсальних помічників, яких можна завжди тримати при собі і використовувати при першій ж необхідності практично для будь яких повсякденних і не тільки задач.

Саме тому однією з найголовніших задач зараз є підвищення ефективності взаємодії між користувачем та його мобільним пристроєм, зокрема взаємодія з пристроєм за допомогою голосового управління. Найпотужнішим розпізнавачем природної мови на мобільних пристроях Android і iOS є Google Now [1]. Існують і аналоги Google Now, які дозволяють виконувати розпізнавання на самому мобільному пристрої без необхідності підключення до мережі інтернет. Одним з них є ядро Pocketsphinx під Android з відкритим вихідним кодом. На сьогодні для Pocketsphinx розроблено тільки 13 акустичних моделей, серед яких немає української.

II. Мета роботи

Метою даної роботи є адаптація ядра Pocketsphinx для голосового розпізнавання української мови, а саме створення акустичної моделі, мовної моделі, граматики, словника, та налаштування самого ядра, для керування Android пристроями українською мовою.

III. Особливості програмної реалізації

Так виглядає мінімальний Java код для конфігурації і запуску процесів розпізнавання ядра Pocketsphinx:

```
PhonMapperphonMapper = newPhonMapper(getAssets().open("commands"));
Grammargrammar = newGrammar(names, phonMapper);
grammar.addWords(commands);
DataFilesdataFiles = newDataFiles(getPackageName(), "ua");
File hmmDir = new File(dataFiles.getHmm());
copyAssets(hmmDir);
saveFile(jsgf, grammar.getJsgf());
saveFile(dict, grammar.getDict());
rec = SpeechRecognizerSetup.defaultSetup()
        .setKeywordThreshold(1e-7f)
        .getRecognizer();
rec.addKeyphraseSearch(KP_SEARCH, commands);
rec.addGrammarSearch(COMMAND_SEARCH, jsgf);
```

У наведеному вище коді відбувається копіювання усіх необхідних для розпізнавання файлів на диск. Після копіювання файлів відбувається конфігурація самого ядра розпізнавання. Вказуються шляхи до скопійованих файлів, а також мінімальна кількість параметрів необхідна для роботи Pocketsphinx, такі як межа чутливості для активаційної фрази і активація режиму фільтрації шумів.

Як видно з цього коду, ядро конфігурується і для граматики і для розпізнавання активаційної фрази. Це робиться для того, аби можна було швидко переключатись між тим, що в даний момент необхідно розпізнавати. Іншими словами, є можливість налаштування ядра для постійного прослуховування і розпізнавання ряду команд, по вказаній граматиці, або очікування спеціальної активаційної фрази, розпізнаючи яку ядро почне розпізнавати інші команди, це дуже корисно у випадку коли необхідно мінімізувати кількість помилкових спрацьовувань. Так виглядає запуск процесу розпізнавання активаційної фрази:

```
rec.startListening(KP_SEARCH);
```

А так запуск розпізнавання по заданій граматиці:

```
rec.startListening(COMMAND_SEARCH, 2000);
```

Другий аргумент (2000) необов'язковий – це кількість мілісекунд, після яких розпізнавання буде автоматично завершуватись, якщо ніхто нічого не говорить. Для того аби отримати результат розпізнавання, необхідно вказати слухача подій, який реалізує інтерфейс *RecognitionListener* і перевизначити наступні його методи для різноманітних цілей [2] (див.рис.1.)

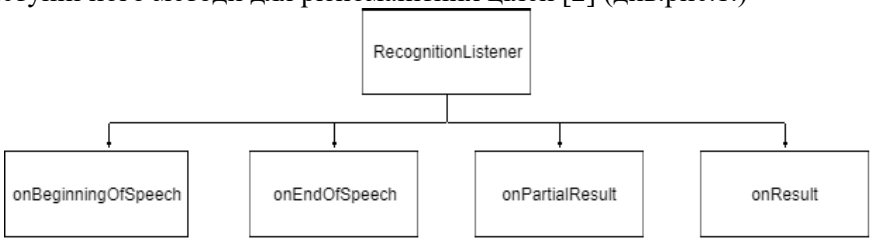
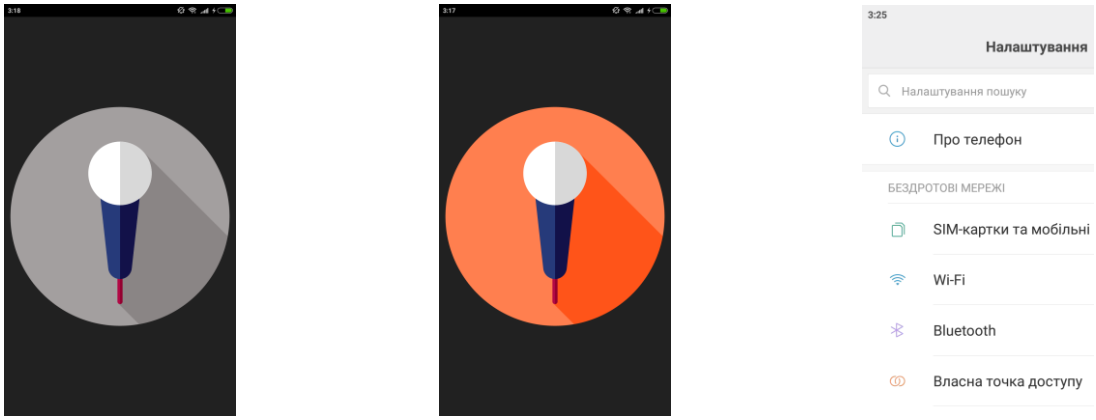


Рисунок 1 – Методи що керують логікою розпізнавання

- *onBeginningOfSpeech* – спрацьовує тоді, коли ядро отримало будь-який звук з мікрофону;
- *onEndOfSpeech* – спрацьовує тоді, коли звук припинився;
- *onPartialResult* – спрацьовує у випадку наявності проміжних результатів розпізнавання, тобто коли лише частина звукового потоку містила команди. Активаційна фраза у цьому випадку спрацьовує, а аргумент *Hypothesis* містить дані про розпізнавання, а саме результат розпізнавання типу *string* і вірогідність того, що розпізнавання було правильним типу *int*;
- *onResult* – містить кінцевий результат розпізнавання. Спрацьовує після виклику методу *Stop* у *SpeechRecognizer*.

Як результат розроблений Android додаток для голосового управління пристроєм українською мовою, на основі адаптованого ядра Pocketsphinx. Програмування Android додатку виконувалось мовою Java, мовна модель і словник створені за допомогою CMU CLMTK, файл граматики створений у форматі JSGF, у якості акустичної моделі була використана адаптована під українську мову стандартна акустична модель.

Одразу після запуску додатку він переходить у режим очікування активаційної фрази (див.рис.2.а.). Як тільки активаційна фраза буде сказана і розпізнана, додаток сповістить про це користувача звуковим сигналом і зміною палітри іконки мікрофону на яскравішу (див.рис.2.б.).



а) б) в)

Рисунок 2 – Основні скріпи розробленого додатку

Після цього додаток переходить у режим очікування команди, як тільки команда буде сказана і розпізнана, для прикладу команда «Налаштування» вона з секундною затримкою йде на виконання (див.рис. 2.в).

Висновки

У роботі проведено аналіз голосового розпізнавання на основі ядра Pocketsphinx. Виявлено проблеми ядра, що стають на заваді голосовому розпізнаванню української мови. У результаті розроблено Android додаток для голосового управління пристроєм за допомогою голосових команд українською мовою.

Список використаних джерел

1. IEEE Xplore. [Електронний ресурс]. – Режим доступу: http://ieeexplore.ieee.org/document/7472820/
2. Сайт проекту SMU Sphinx. [Електронний ресурс]. – Режим доступу: https://cmusphinx.github.io/wiki/tutorialpocketsphinx/