Continuous Speech Recognition and English Grammar correction on Android Platform

Prof.K.T.Talele
Sardar Patel Institute
of Technology
talelesir@gmail.com

Milind Jadhav
Sardar Patel Institute
of Technology
milind.jadhav27391@gmail.com

Siddhesh Salgaonkar
Sardar Patel Institute
of Technology
siddhesh21ace@gmail.com

Aniruddha Mandale
Sardar Patel Institute
of Technology
aniruddham@gmail.com

***Abstract*—This paper introduces a speech recognition system of English language sentences followed by grammar correction using LanguageTool. The acoustic model of this system is provided by CMUSphinx, and the language model is acquired from the lmtool web service. This system makes use of PocketSphinx recognition engine as a decoder. In addition, the system uses LanguageTool, an open source style and grammar checker for grammatical correction of recognized speech. The entire system is deployed on the Android platform.**

Our aim through this project is to develop a system which provides on the fly speech correction facility. Thus the operating system used is Android being the most popular and adaptable mobile platform.

Keywords- Sphinx; speech recognition; language model; LanguageTool

I. INTRODUCTION

It is often seen that when students from a vernacular medium of education in India enroll in junior colleges, they face tremendous difficulties in delivering fluent, grammatically correct and confident speeches. People from other sectors also face difficulties in speaking proper, fluent English. For them, this system will be very useful, as it is a mobile application running on the Android platform.

A mobile application running on the Android OS, This system will provide on-the-fly speech correction facility. Before delivering a speech or speaking in a group discussion these students can speak into it. It will provide users with a facility for recording their speech upon which grammatical corrections will be applied on it. After they finish speaking, the system will immediately correct their speech. Thus, it will be of good use to the vernacular pass-outs and will greatly help all the sectors of the society who want to improve their verbal and communication skills.

Pocketsphinx [1], which is developed by CMU is an embedded speech recognition engine used for fast speech recognition, has very high recognition rate in English continuous speech recognition. By means of this recognition engine and through training the acoustic model and language model, a Speech recognition system of high-performance is built. This is followed by feeding this recognized text to the LanguageTool for grammar correction.

II. STRUCTURE OF THE SYSTEM

The proposed system consists of feature extraction, acoustic model, language model, recognition engine and a grammar correction module. The block diagram in Fig.1 depicts the overall processing. In the following text, these parts of this system will be introduced separately.

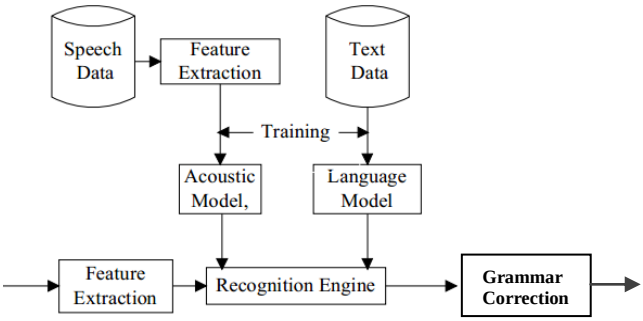

Figure 1.Basic structure of proposed system

A. *Feature Extraction*

SphinxBase, which is known as the famous public library of Sphinx speech recognition project of CMU, mainly uses Mel-Frequency Cepstral Coefficients (MFCC) to carry out the front feature extraction of speech recognition system. The processing is divided into seven stages as shown in Fig.2.

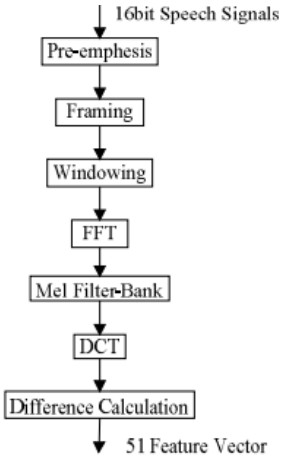

Figure 2.Flow chart of MFCC

Feature extraction, referred to the front-end processing, generates a set of 51-dimension feature vectors representing the important characteristics of speech signals. In short, MFCC converts the stream of speech data sampled at 16 KHz to 12-element Mel-scale frequency cepstrum vector and a power coefficient. [2]

B. *Language Model Training*

Cmuclmtk (CMU-Cambridge Statistical Language Modeling Toolkit) is used to train the language models. A large number of text data is taken as input. Bigram or Trigram Language Models, which mean that the probability of the occurrences of a word only depends on the previous one or two words, is produced in ARPA format. Then, the file of ARPA format can be converted into binary dump file by using Lm3g2dmp utility. The basic flow [3] in generating a language model is shown in Fig.3.

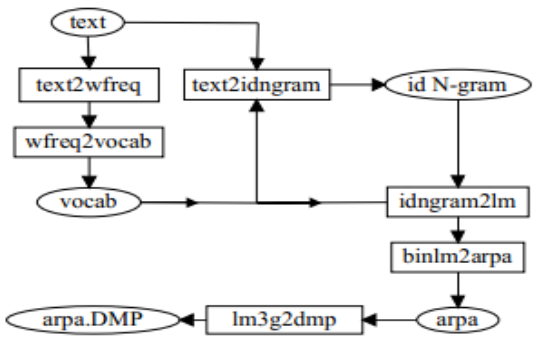

Figure 3.Flow chart of language model training

C. *PocketSphinx*

PocketSphinx is a decoding engine for speech recognition system developed at CMU. It consists of a set of libraries that include core speech recognition functions. The libraries are written in C and have been compiled on Linux and Windows XP.

The input of PocketSphinx is audio file in wave format, which can be obtained by microphone input or by reading wave files directly, and the final output of the recognition results are displayed as text. The algorithms of recognition engine mainly include four parts: the calculation of acoustic parameter, the calculation of Gaussian function, the calculation of Gaussian mixture model and Viterbi search. The search algorithm of decoder mainly uses Viterbi-Beam search algorithm [4]. In the search processing, it continually looks for the possible optimal state subsequence and records the matching information, and then prunes them according to different clipping thresholds on different levels. After processing all of the feature vectors, trace back to obtain the optimal word sequence.

D. *Grammar Correction*

LanguageTool, [5] the style and grammar checker used in this system takes a text and returns a list of possible errors. To detect errors, each word of the text is assigned its part-of-speech tag and each sentence is split into chunks, e.g. noun phrases. Then the text is matched against all the checker's pre-defined error rules. If a rule matches, the text is supposed to contain an error at the position of the match. The rules describe errors as patterns of words, part-of-speech tags and chunks. The process is divided into following tasks as explained below.

D.1. *Part-of-Speech Tagging*

Part-of-speech tagging (POS tagging or just tagging) is the task of assigning each word its POS tag. It is not strictly defined what POS tags exist, but the most common ones are noun, verb, determiner, adjective and adverb. Nouns can be further divided into singular and plural nouns, verbs can be divided into past tense verbs and present tense verbs and so on.

The more POS tags there are, the more difficult it becomes - especially for an algorithm - to find the right tag for a given occurrence of a word, since many words can have different POS tags, depending on their context. In English, many words can be used both as nouns and as verbs. For example house (a building) and house (to provide shelter). Only about 11.5% percent of all words are ambiguous with regard to their POS tags, but since these are the more often occurring words, 40% percent of the words in a text are usually ambiguous.POS tagging is thus a typical disambiguation problem: all possible tags of a word are known and the appropriate one for a given context needs to be chosen.

Even by simply selecting the POS tag which occurs most often for a given word - without taking context into account - one can assign 91% of the words their correct tag. Taggers which are mostly based on statistical analysis of large corpora have an accuracy of 95-97%.

D.2. *Phrase Chunking*

Phrase Chunking is situated between POS tagging and a full-blown grammatical analysis: whereas POS tagging only works on the word level, and grammar analysis (i.e. parsing) is supposed to build a tree structure of a sentence, phrase chunking assigns a tag to word sequences of a sentence.

Typical chunks are noun phrase (NP) and verb phrase (VP). Noun phrases typically consist of deter- miners, adjectives and nouns or pronouns. Verb phrases can consist of a single verb or of an auxiliary verb plus infinitive. For example, the dog, the big dog, the big brown dog are all examples of noun phrases. As the list of adjectives can become infinitely long, noun phrases can theoretically grow without a limit. However, what is called noun phrase here is just an example and just like in POS tagging everybody can make up his own chunk names and their meanings.

Chunking works on a POS tagged text just like POS tagging works on words: either there are handwritten rules that describe which POS tag sequences build which chunks, or a probabilistic chunker is trained on a POS tagged and chunked text. These methods can be combined by transferring the knowledge of a probabilistic chunker to rules.

As chunking requires a POS tagged text, its accuracy cannot be better than that of the POS tagger used. This is backed by the fact that even the best chunker listed on [Chunking] reaches a precision and recall of 94%, which is less than an average tagger can achieve.

D.3. Grammar Checking

Rule-based checking: In this approach, a set of rules is matched against a text which has at least been POS tagged. It has many advantages:

1) A sentence does not have to be complete to be checked, instead the software can check the text while it is being typed and give immediate feedback.
2) It is easy to configure, as each rule has an expressive description and can be turned on and off individually.
3) It can offer detailed error messages with helpful comments, even explaining grammar rules.
4) It is easily extendable by its users, as the rule system is easy to understand, at least for many simple but common error cases.
5) It can be built incrementally, starting with just one rule and then extending it rule by rule.

III. IMPLEMENTATION

The implementation of this system is divided into two parts viz.:

1) Speech Recognition
2) Grammar Correction

To carry out both these steps, we will not be attempting to break new ground. With a fair amount of academic research in both these areas, we will use two existing solutions as mentioned above.

Being Android project, the basic unit of work used is Activity. Thus there are two activities separately for speech recognition and grammar correction. The first activity is responsible for recognizing the speech and displaying it in the text form to the user. This displayed text is the used as an input for the grammar correction process in the second activity using LanguageTool.

Agreement between Indefinite Article and the following word:

If the indefinite article is followed by a word whose pronunciation starts with a vowel sound, an has to be used instead of a. If it is one of a, e, i, o, u, the word probably starts with a vowel - but there are exceptions
.

Here are some examples where the a,e,i,o,u rule applies, together with the correct indefinite article:
a test, a car, a long talk
an idea, an uninteresting speech, an earthquake

Here are some exceptions:
a university, a European initiative
an hour, an honor

This type of correction was not possible to be implemented by writing rules as mentioned above. We have used two text files containing these exceptional words for ‘a’ and ‘an’ for checking .Thus we created our own algorithm which works as follows:

Algorithm:

Create a string array words[] to store space delimited words of a sentence.

for i=0 to array.length-2 do
  if
    word precedes with ‘a’ and starts with a vowel and if it is not exception then replace ‘a’ by ‘an’.
  if
    word precedes with ‘a’ and is an exception then replace ‘a’ by ‘an’.
  if
    word precedes with ‘an’ and it does not start a vowel and if it is not exception then replace ‘an’ by ‘a’.
  if
    word precedes with ‘an’ and is an exception then replace ‘an’ by ‘a’.

IV. CONCLUSION

In this paper, a continuous speech recognition system is built with grammar correction on the mobile Android platform. Seeing from the experiment results, a limited-vocabulary continuous speech recognition system which is built based on the Sphinx system has good performance.

Grammar correction capabilities are highly extensive with exceptional performance. The error rate is almost negligible when tested with wide range of incorrect grammar sentences. Our ultimate aim would be to achieve full scale English language recognition which is currently limited because of limited processing power of handheld devices. We plan to implement the present system on other popular mobile platforms such as iOS, Blackberry, and Windows Phone OS etc.

REFERENCES

[1] CMU Sphinx documentation wiki, http://cmusphinx.sourceforge.net/wiki/

[2] *Yun Wang; Xueying Zhang; , "Realization of Mandarin continuous digits speech recognition system using Sphinx," Computer Communication Control and Automation (3CA), 2010 International Symposium on , vol.1, no., pp.378-380, 5-7 May 2010*

[3] LanguageModelingToolkit, http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html

[4] Jun Yuan, “Optimization and application of the Viterbi algorithm in HMM continuous speech recognition” electronic technique. Shanghai, vol.2, pp.48-51, 2001.

[5] *LanguageTool* style and grammar checker http://www.languagetool.org/

[6] *A rule based style and grammar checker* http://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf