LumenVox®
Speech Understood

TUNING STRATEGIES FOR SPEECH RECOGNITION

Executive Summary:
A look at how to perform speech tuning, from transcription to analysis to making and testing changes. Includes data about the effectiveness of speech tuning, i.e. the kinds of gains a developer can expect from investing time into performing tuning.

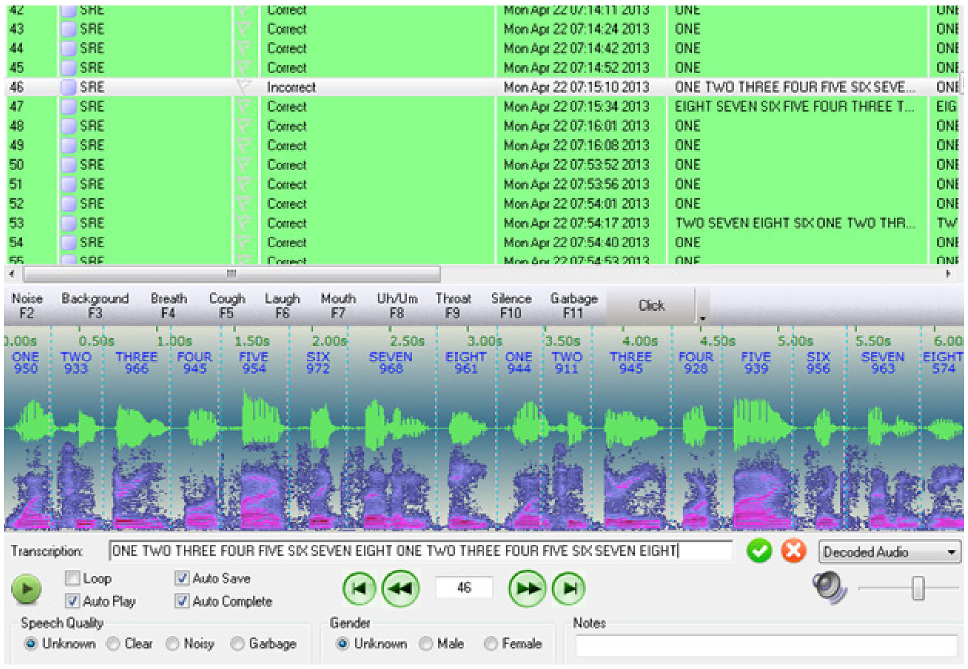
Audience:
This whitepaper is intended for a general audience with little to no experience with speech applications. It discusses speech application tuning in a way that is meant to be accessible to anyone interested in how to get the best performance out of their applications.

PRACTICAL TUNING STRATEGIES

Tuning a speech recognition application is the most effective way to improve a speech solution and deliver better caller experiences. The tuning process involves analyzing data from users in order to make prompt, grammar, and call flow enhancements to an existing speech-enabled Interactive Voice Response (IVR) application. While speech industry experts recommend that 40-50% of the total development time of a speech application be spent on tuning, this whitepaper will address the techniques to create a repeatable tuning process that reduces costs while still producing significant improvements.

The Tuning Tradition: Benefits and Challenges

After a speech recognition application is deployed, it can be set to capture data for each interaction between a user and the automatic speech recognizer (ASR). This data generally includes the audio that was spoken, any grammars that were used, the recognition result, and miscellaneous other information such as the ASR settings. After a representative data set has been collected (see the LumenVox whitepaper *Collecting Data for Speech Tuning*), speech developers traditionally transcribe and analyze every call using tuning tools such as the LumenVox Speech Tuner.



42	SRE	Correct	Mon Apr 22 07:14:11 2013	ONE	ON
43	SRE	Correct	Mon Apr 22 07:14:24 2013	ONE	ON
44	SRE	Correct	Mon Apr 22 07:14:42 2013	ONE	ON
45	SRE	Correct	Mon Apr 22 07:14:52 2013	ONE	ON
46	SRE	Incorrect	Mon Apr 22 07:15:10 2013	ONE TWO THREE FOUR FIVE SIX SEVE...	ON
47	SRE	Correct	Mon Apr 22 07:15:34 2013	EIGHT SEVEN SIX FIVE FOUR THREE T...	EIG
48	SRE	Correct	Mon Apr 22 07:16:01 2013	ONE	ON
49	SRE	Correct	Mon Apr 22 07:16:08 2013	ONE	ON
50	SRE	Correct	Mon Apr 22 07:53:52 2013	ONE	ON
51	SRE	Correct	Mon Apr 22 07:53:56 2013	ONE	ON
52	SRE	Correct	Mon Apr 22 07:54:01 2013	ONE	ON
53	SRE	Correct	Mon Apr 22 07:54:17 2013	TWO SEVEN EIGHT SIX ONE TWO THR...	TW
54	SRE	Correct	Mon Apr 22 07:54:40 2013	ONE	ON
55	SRE	Correct	Mon Apr 22 07:54:53 2013	ONE	ON

Transcription: ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT

Decoded Audio

Loop ☐ Auto Play ☑ Auto Save ☑ Auto Complete ☑

46

Speech Quality: [✓ Unknown] Clear Noisy Garbage

Gender: [✓ Unknown] Male Female

Notes:

Figure 1:
The LumenVox Speech Tuner transcription utility allows developers to accurately transcribe every utterance in a data set.

Common problems developers look for are out-of-grammar utterances, cases where confusing prompts lead to callers giving improper responses, or heavy background noise. The process is very thorough and is a good idea to do on occasion because it gives the developer a large set of data to work with. A developer will also get a complete set of out-of-grammar and confidence score data that can be used to accurately measure a number of important metrics (see the LumenVox whitepaper *Calculating Speech Recognition Accuracy* for more information about how to measure the performance of a speech recognizer).

This technique, however, may obfuscate how well the application as a whole is performing. While the accuracy metrics that are normally calculated during tuning provide a good idea of how well the ASR is performing, the application's performance can be difficult to see. It is a case where it is easy to miss the forest (is the application meeting its functional goals?) gets lots for the trees (the performance of the ASR).

- It is time consuming, often taking 5-10 times "real time" to transcribe the utterances, meaning for every hour of recorded speech it may take 5-10 hours to transcribe.
- Difficulty in validating proposed changes means there is no way of knowing the changes are going to be effective before redeploying the application.

Overcoming the Challenges

Let's look at a case study, a speech-enabled gift code redemption line, to illustrate how we can successfully approach these roadblocks. An electronics retailer wanted to create a program for their frequent buyer card customers to reward them for their loyalty. The retailer sent out a mailer with a unique gift code, an 800 number to call, and the choice of three gifts: a wireless keyboard, a food processor, or a pair of headphones.

Since there were thousands of frequent buyers, the retailer did not want to overburden their customer service department, yet it still wanted to provide this valuable marketing program. Therefore, the goal of the speech-enabled IVR system was quite simple: Callers call in, say their gift code number and the name of the gift they want, then hang up. Once the information was captured, the gift would be mailed out to them on the next business day.

The program was a success, with high call volumes. When they first started tuning the application, the electronics retailer transcribed 1,000 calls per day, which was every single record generated by the automatic speech recognizer. As you can imagine, this task became overwhelming, yet they still wanted to tune on a daily basis. So they started using the application log to focus only on the calls that failed, which instantly reduced the number of calls they needed to tune.

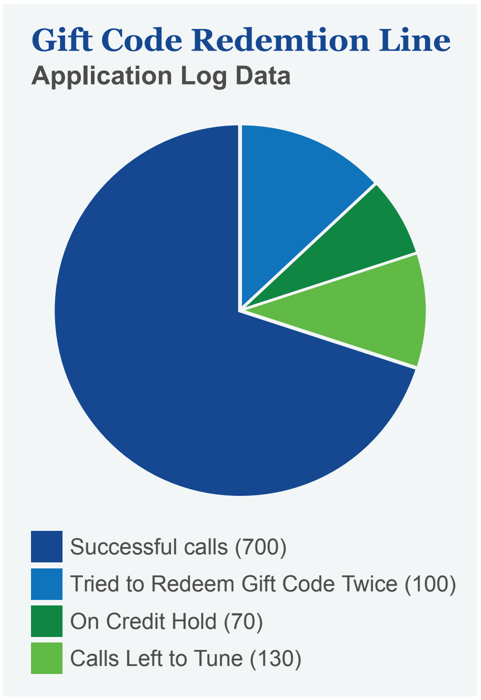


Figure 2:
By sorting out successful from unsuccessful, it is possible to drastically reduce the amount of time spent tuning by focusing only on cases where there were potential speech recognition issues.

The application logged out data such as call hang-ups, call routing information, etc. — information about how callers used the system, but not specifically about speech recognition. Figure 2 shows how they reduced the number of calls to analyze from 1000 to 130 calls.

This example illustrates that the retailer does not need to focus tuning efforts on calls from customers that either try to get two gifts (which is not allowed), or those that haven't paid their bills (they don't get to redeem a gift), because these situations do not point to a failing of speech recognition. It may be useful to review the application to help those customers in other ways, but there is no need to transcribe all of those utterances for speech tuning.

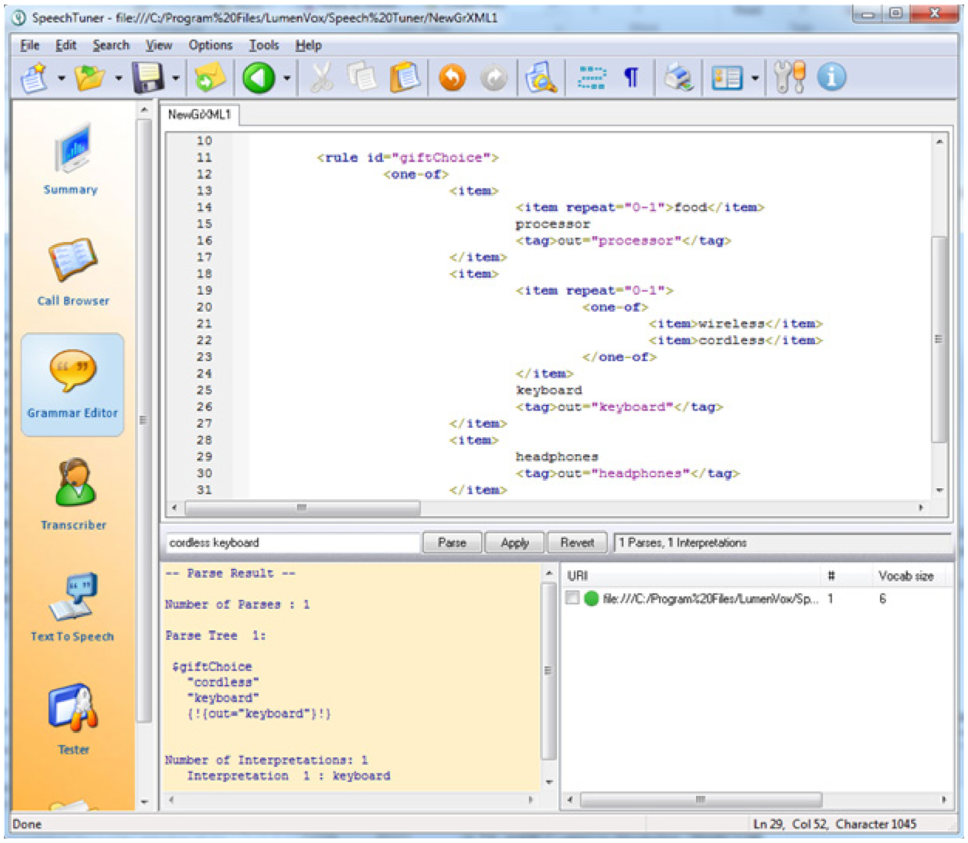
Thus the retailer found they only need to look at 13% of the calls in the system. What is most important to them is finding out why the remaining small percentage of callers became frustrated with the system and transferred to Customer Service or simply hung-up without completing the call. These 130 are the calls they need to cross compare with the results from the speech recognizer.

The Tuning Process

Now that the retailer has reduced the number of calls to analyze from 1000 to 130, they went about the task of transcribing these calls with the LumenVox Speech Tuner. Having a small set to work with helped the retailer immediately pinpoint problems.

For example, one of the gifts in the application's grammar of the application was "wireless keyboard," but many of the callers were calling it a "cordless keyboard." This caused the confidence scores for these utterances to be lowered, reducing the overall accuracy of the system.

Using the Tuner, the retailer added “cordless” as an option in the grammar, and used the built-in Grammar Editor to verify the change was correct.


SpeechTuner - file:///C:/Program%20Files/LumenVox/Speech%20Tuner/NewGrXML1
File Edit Search View Options Tools Help
Summary
Call Browser
Grammar Editor
Transcriber
Text To Speech
Tester
NewGrXML1
Parse
Apply
Revert
Done

```
        <rule id="giftChoice">
                <one-of>
                        <item>
                                <item repeat="0-1">food</item>
                                processor
                                <tag>out="processor"</tag>
                        </item>
                        <item>
                                <item repeat="0-1">
                                        <one-of>
                                                <item>wireless</item>
                                                <item>cordless</item>
                                        </one-of>
                                </item>
                                keyboard
                                <tag>out="keyboard"</tag>
                        </item>
                        <item>
                                headphones
                                <tag>out="headphones"</tag>
                        </item>
```

cordless keyboard

1 Parses, 1 Interpretations

```
-- Parse Result --

Number of Parses : 1

Parse Tree  1:

 $giftChoice
   "cordless"
   "keyboard"
   {!{out="keyboard"}!}

Number of Interpretations: 1
   Interpretation  1 : keyboard
```

URI	#	Vocab size
file:///C:/Program%20Files/LumenVox/Sp...	1	6

Ln 29, Col 52, Character 1045

Figure 3:
The Speech Tuner’s Grammar Editor allows users to test grammar changes. The user was able to add “cordless” as an option in front of “keyboard” and verify the change works properly.

After all of the required changes were done and they tested the results with the Speech Tuner, they were able to redeploy the application again with greater accuracy. And future tuning cycles took less effort, because as the quality of recognitions improved, the percentage of calls with problems decreased.

A Word of Caution

This document has focused on one particular type of tuning, which is a fast and pragmatic approach that focuses on identifying problems. This is not the only type of tuning, and for many cases it may not be the correct approach. For instance, when measuring the overall accuracy of an application , it is important to have a representative sample of data — filtering the data so that only failures are examined will obviously skew any accuracy measurements.

It's also important when making large-scale changes to grammars or settings to have a representative test set of data. One mistake developers sometimes make when tuning is to only test changes against failures, without considering the possibility that though change may help turn failures into successes, it can also have the opposite effect, turning former successes into new failures.

In general, the larger the proposed change, the more testing and test data is required. In the example with the electronics retailer, the change was very small. Adding a single option into a small grammar is usually safe, so it wasn't necessary for the retailer to do a round of in-depth tuning in this particular instance, but it's a good idea to do so periodically

CONCLUSION

Speech recognition application tuning is an absolutely essential function of a speech recognition solution deployment, but it does not have to be a painful one. If you follow the techniques outlined in this paper you can decrease your time spent while producing a dynamic and well performing speech solution.