End-to-end ASR DNN Engine

POSITIONING (EDITED – BRIAN OVERLAND)

AUG 29, 2021 (DRAFT 7)

Speech + Authentication Automated Speech Recognition, Text-to-Speech, Call Progress Analysis, Active/Passive Voice Biometrics, Fraud Detection, Multifactor Authentication

Highlights of the new engine

Here are the highlights of the features of the new LumenVox ASR engine. Subsequent sections drill down into these features, giving more details:

- End-to-end acoustical modeling with Convolutional Neural Networks (CNN). This extends the benefits of neural networks from the sound-recognition stage all the way to the production of text, removing the need for separate speech engines to accommodate dialects and accents.
- Transfer-learning techniques to improve learning efficiency.
- A streaming model that operates in an online manner, thereby boosting responsiveness and performance.
- Performance and accuracy aided by statistical n-gram models.
- Performance and accuracy aided by traditional SRGS grammars.
- Use of quantized sorted-tree n-gram Statistical Language Models (SLM), efficiently compressed in storage, enabling higher performance in less memory.

Commented [SH1]: Comma should be removed.

Commented [SH2]: Unnecessary dash

End-to-End Acoustical Modeling with Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are a more advanced version of the neural-net and DNN technologies. A CNN approach develops patterns in successive layers, each building upon the layer before it.

In the visual-pattern recognition world, CNN starts by recognizing dots and small areas on the screen. Then it builds on these items to "see" lines and curves. Finally, it builds on lines and curves to see shapes.

In the speech recognition world, CNN does something similar: it hears and recognizes small bits of sound; then it builds on sounds to recognize phonemes. Finally, it builds on phonemes to recognize words and phrases.

In contrast, the legacy model does the following:

1. The ASR produces a series of phonemes (specific pronunciations).
2. A language model is then used to make sense of these phonemes and build them into words.

The legacy approach may work until the engine encounters a new dialect or a new accent… or sometimes just idiosyncrasies in the way a person talks. At that point, it fails. Under the old model, every time the dialect or the accent changes, a new speech model is required. That can stop customer service in its tracks.

With the new speech engine, the neural-net technology is **end-to-end** because it uses the neural net to fold in the problem of dealing with different pronunciations. In other words, we let the most expert system—the neural network itself—handle the problem. In doing so, the engine takes advantage of machine-learning technology to solve the problem of variations in speech.

Transfer Learning Techniques

Transfer learning is a machine learning technique where once a model has been trained for a specific task, it is then reused to initialize a model for a different task. This is a very common approach for deep learning where pre-trained models are being used as the starting point on many fields.

In our case this means that we can reuse models trained on thousands of hours of data, an amount only available for a small fraction of languages (e.g., English), to initialize the training of models for languages where only a reduced amount of data is available (e.g., Italian).

This results in much better accuracy and performance than when training such a model from scratch using only target language data. Furthermore, transfer learning speeds up the training process considerably (days versus months). This lowers the cost of training models for new languages.

Statistical n-gram Language Models

An "n-gram" is a sequence of n words that is statistically likely. For example, "to beak" is an unlikely bi-gram (group of 2) but "to be" is likely a bi-gram; we should expect the words "to be" to appear in that order but not "to beak."

The advantage of this technology is that it aids in determining what words were probably said, thereby making the engine less likely to make a mistake. For example, let's say the speaker said something like this:

I'd like to book a fly please.

Using n-gram analysis, the software can determine that it's much more likely that the speaker meant the following sentence, but the words were not heard perfectly. "Book a flight" is a common tri-gram.

I'd like to book a flight please.

By understanding what combinations of words are more likely to appear together, the speech engine can more accurately determine what the speaker meant to say.

Quantization of Sorted-Tree ~~n-gram~~ Speech Language Models (SLM)

With the technology used by the new engine, language models ~~n-grams (common sequences of words)~~ are stored using sorted trees—a data structure that can be searched with great efficiency. The new engine also uses a compression scheme that allows larger and deeper language models ~~more of these n-grams~~ to be stored in a smaller space. More detailed language models are therefore represented in a smaller space and are more efficient, doing more with less memory.

Use of Traditional SRGS Grammars

SRGS grammars use BNF or XML style to model patterns of words that are expected to appear together and in what sequence. The new engine uses these grammars as another way to predict what patterns of words are more likely. This enables more accurate processing of sounds into text.

Commented [SH3]: Comma should be removed

Streaming Models

The new engine makes direct use of speech input in real-time. Incoming sounds are processed as received, and knowledge picked up during one pass is utilized right away. As a result, there are no multiple passes or extra-cycles of processing. Consequently, there is no latency or perceptible delay between the caller speaking and the system responding. The result is a system that is faster and more responsive and makes for happier callers.

Commented [SH4]: real-time instead of real time

Architecture

The block diagram on the next page shows the architecture of our engine. It consists of three main blocks:

1. **Audio preprocessor**. It extracts features from the audio -- Mel-Frequency Cepstral Coefficients (MFCC) — and prepares them in chunks, to be processed by the network.

2. **DNN**. This block processes the input audio features and generates sequences of characters (words) in the target language. Internally, it consists of two parts:

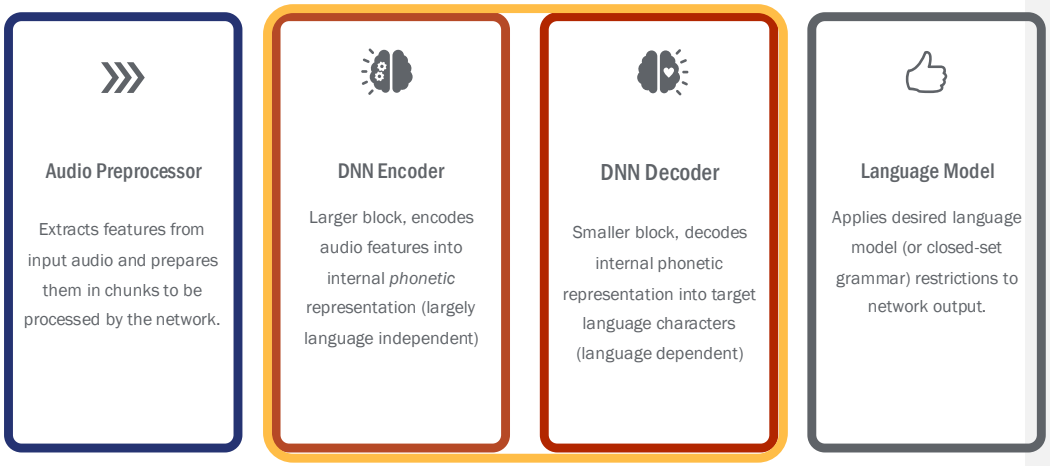
    a. An **encoder** that encodes the audio features into an internal phonetic representation. This is largely language independent.

    b. A **decoder** that transforms this internal phonetic representation into target language characters. This is language dependent.

3. **Language model**. Applies the restrictions enforced by the desired language model (or traditional closed-set grammar) to the network output.

Commented [SH5]: Comma should be removed

Commented [SH6]: Comma should be removed



A High-Level Overview: HMM vs. DNN vs. CNN

Traditionally, speech recognition engines rely on Hidden Markov Models (HMM). This is a process of statistical analysis using the behavior of an external factor to predict the behavior of an internal, hidden factor. Speech recognition engines look at the external data—the sounds produced—to predict the words intended by the speaker, although this intent is not always obvious (due to differences in the way which people pronounce words).

The next generation of technology is "Deep Learning," implemented through Deep Neural Networks (DNN). These systems involve more training than HMM systems, but that training is put to extremely good use. The neural network is fed a large sample of raw data and corresponding results; then, it uses back propagation to move toward correct pattern-recognition behavior, based on experience.

Commented [SH7]: Comma should be added after "then"

Training and Benefits of Deep Learning

Training is the process of machine learning to enable the system to recognize patterns of sounds and text. Neural networks improve HMM technology. Using linear algebra to implement back propagation, node values are revised thousands of times, until the network can recognize patterns accurately. The next section provides more details on how HMM and neural networks work and how they differ.

Commented [SH8]: Comma should be removed

Commented [SH9]: Comma should be removed

Convolutional Neural Networks (CNN) provide a still deeper level of analysis, building up levels of comprehension and recognition, as explained on page 2.

Traditional HMM systems benefit from training but have less potential for accuracy than deep-learning CNN systems have; our state-of-the-art deep learning systems achieve better results and benefit even more from training.

Limitations of Legacy Systems

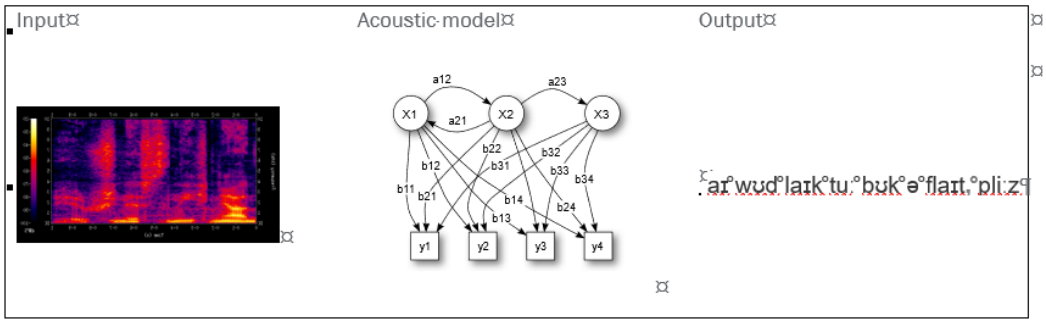
HMM-based systems are trained using audio as the input to the acoustic model; phonemes are output. A phonetic lexicon must then be used to convert the phoneme data into text.

This traditional (legacy) approach has limitations. Such a system cannot be trained for more than one dialect of a language at a time. Obtaining the training needed can be expensive because it needs to be professionally transcribed to match the exact audio contents.

Commented [SH10]: Comma should be removed

The following graphic shows how the old acoustical model analyzes sounds and produces phonemes, but these need to be in turn analyzed by a speech model for a specific language, dialect, and accent.

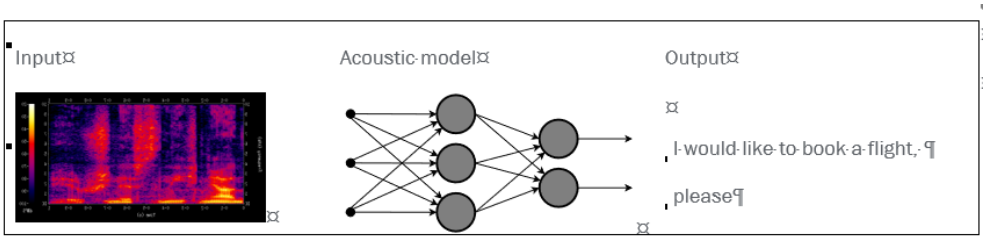
Commented [SH11]: Comma should be used instead of semicolon

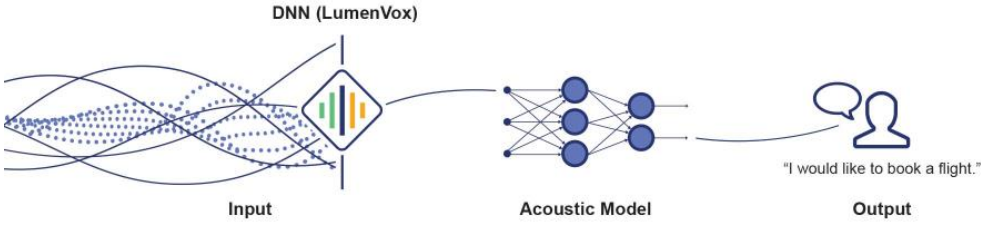


End-to-end systems

An end-to-end speech recognizer can be trained directly using the audio utterances and resulting text. There is no need to use phonetic transcriptions since the network learns an internal phonetic representation as part of the training process. With our new ASR engine, we do not need to use a phonetic dictionary (or lexicon) to transcribe the training text material.

The following graphic shows how much more efficient and accurate this process is, as it uses neural networks to take sound and produce words of text as the output.





Because this approach internally learns the phonetic representation of the audio, it is more flexible in terms of variations in the pronunciation of the same word; the neural network will learn whatever combination of audio/word it is presented with.

For each language we train the network on, we leverage hundreds or even thousands of hours of audio, covering many dialectal variations and environmental conditions. This allows us to train a single model for one language instead of having to train individual ones (e.g., one English model would cove US, UK, AU, etc.).

For the newer systems, it matters less how a word is pronounced since it will match audio with written word directly. So given enough examples, it will learn to recognize the word "**tomato**" for both US pronunciations ("**tə**ˈ**meɪ**ˌ**toʊ**") and UK pronunciations.

End-to-end systems can also be fine-tuned for specific conditions (such as telephone channel or dialect), but given the very good generalization performance of the acoustic model, it is usually enough to fine-tune the language model—using text data, much cheaper and easier to obtain than audio data.

Language model

Our engine provides out-of-the-box statistical language models trained using millions of text lines from many sources, accounting for several use cases: generic transcription, spellings, digit string recognition, etc. As mentioned earlier, the deeper learning systems benefit from more programmed expertise up-front, in the form of training that we, the developers, place into the engine before the customer sees it.

We also offer several options to customize the language modeling part of the recognition:

Commented [SH12]: add "the" before "pronunciation"

Commented [SH13]: remove one of the "we train" instances

Commented [SH14]: Remove comma

Commented [SH15]: Comma should be removed

Commented [SH16]: Remove double spacing between words

Commented [SH17]: Comma before "but"

Commented [JH18]: Should we mention language model training and adaptation somewhere? We don't do anything special, it's basic Kenser-Ney smoothing and entropy-based pruning.

Commented [JP19R18]: @Joachim Hofer That's a good point, I'll add a section. Maybe not about the specific training we use, but about the adaption possibilities of the engine.

Commented [SH20]: up-front

- Adding lists of words or phrases, which can be done on a per-request basis or caching such requests; this feature boosts the recognition of the specified words or phrases. A typical use case would be adding a list of employees that need to be recognized or specific product names (e.g., pharmaceutical drugs or specific restaurant dishes).

- Adapting the standard language model with a small amount of domain-specific text to boost recognition of in-domain audio. This is for those situations where there is not enough text data to fully train a new language model, but enough so that recognition of domain-specific text is enhanced.

Commented [SH21]: Remove comma

Commented [SH22]: domain-specific

Preliminary Results

Transcription engine

English

We run tests on non-matched data (i.e. datasets that have not been seen during training) comparing our engine with the Cobalt one:

- CSLU Multilingual (English set): landline speech, part scripted, part free, short utterances
- ICSI Meeting Corpus: very challenging set

Test Name	Dialect	Quartznet	en_GB-8kHz	en_US-8kHz	En_US_robust-8kHz
CSLU Multilingual	?	17.4 %	32.2 %	17.6 %	21.2 %
ICSI	mixed	43.3 %	55.2 %	43.7 %	41.1 %

As we can see, on those sets our engine performs better than Cobalt for the CSLU Multilingual dataset and almost on pair on the ICSI dataset.

Part of the performance drop for ICSI can be explained due to differences in the text content, that could be accounted for by retraining or fine-tuning the language model. For this, we need to have full access to the engine, only possible when developed fully in-house.

Commented [SH23]: Remove comma

Commented [SH24]: Comma should be added

Comparison with cloud offers

Note: we are showing here the word accuracy, so higher values indicate better performance

TESTSET	AMAZON TRANSCRIBE	AMAZON LEX	GOOGLE SPEECH-TO-API	MICROSOFT COGNITIVE SERVICES SPEECH	DEEPGRAM	LUMENVOX STU ASR (TRANSCRIPTION)	LUMENVOX STU ASR (GRAMMAR)
CSLU DIGITS	94.30%	91.50%	89.3%	98.40%	94.7%	98.50%	98.7 %
CSLU WORDS	37.10%	31.10%	41.5%	62.1%	60.0%	58.80%	98.7 %
CSLU PHRASES	71.00%	66.50%	73.9%	86.3%	76.8%	84.60%	98.6 %
MACROPHONE	*Not possible to normalize results*	93.40%	*Not possible to normalize results*	*Not possible to normalize results*	87.8%	95.40%	*Not applicable to grammars*

As you can see, the performance advantage for the new LumenVox ASR model is substantial.

Spanish

Note: we are showing here Word Error Rate (%), so lower values indicate better performance

We have evaluated both our own engine and Cobalt's one on several databases, two of them commercial telephone datasets (SALA II and VAHA), and four of them being open-source, non-telephone datasets.

Commented [SH25]: Remove the word "a"

	QUARTZNET + SLM	CUBIC ES-MX_8KHZ
MEXICAN SALA II	16.2 %	24.8 %
VOICE ACROSS HISPANIC AMERICA (VAHA)	9.8 %	19.3 %
MULTILINGUAL LIBRISPEECH SPANISH	15.7 %	32.2 %
MULTILINGUAL TEDX SPANISH	21.3 %	32.8 %
MOZILLA COMMON VOICE SPANISH	17.0 %	35.7 %
VOXFORGE (RANDOM SUB-SET OF WHOLE CORPUS, MAY OVERLAP WITH COBALT'S TRAINING DATA, THUS THE UNREALISTICALLY GOOD RESULTS HERE)	4.6 %	0.9 %

Our engine performs well on all datasets, whereas Cobalt performance drops specially for the open-source datasets. According to the internal README provided with Cobalt models, their engine seems to be trained mostly on telephone speech and may be limited in terms of generalization to different channels and environments.

Commented [SH26]: Either a comma after "drops" or rewording "specially"

German

For German, we did profile both engines in terms of memory usage and CPU. We trained the German models ourselves, with training from Cobalt.

Commented [SH27]: Comma should be added

Commented [SH28]: Comma should be removed

The following tables summarize the results:

Cobalt

SLM Pruning	SLM Disk Size (HCLG.fst)	Mozilla WER	# Streams	Approx. RAM	Run-time (cf. 56.3 s)
1e-7	380 MB	16.1 %	48	2.4 GB	0 m 57 s
1e-7	380 MB	16.1 %	72	3.3 GB	1 m 06 s
1e-7	380 MB	16.1 %	96	3.8 GB	1 m 27 s
1e-7	380 MB	16.1 %	120	4.2 GB	1 m 48 s
1e-8	2.3 GB	12.0 %	48	4.2 GB	0 m 57 s
1e-8	2.3 GB	12.0 %	72	5.1 GB	1 m 07 s
1e-8	2.3 GB	12.0 %	96	5.8 GB	1 m 26 s
1e-8	2.3 GB	12.0 %	120	6.6 GB	1 m 48 s

Own engine.

This is with 2 second chunks and up to 120 simultaneous streams:

SLM Pruning	SLM Disk Size (KenLM bin)	Mozilla WER	# Streams	Approx. RAM	Run-time (cf. 56.3 s)
1e-8	150 MB	9.7 %	48	5.0 GB	0 m 57 s
1e-8	150 MB	9.7 %	72	5.0 GB	0 m 58 s
1e-8	150 MB	9.7 %	96	5.0 GB	0 m 58 s
1e-8	150 MB	9.7 %	120	5.0 GB	1 m 01 s
1e-9	690 MB	8.2 %	48	5.9 GB	0 m 57 s

1e-9	690 MB	8.2 %	72	5.9 GB	0 m 58 s
1e-9	690 MB	8.2 %	96	5.9 GB	0 m 58 s
1e-9	690 MB	8.2 %	120	5.9 GB	1 m 01 s

As you can see, for similarly sized language models (here usually the bigger the better, but for Cobalt things get out of hand quite fast), our engine outperforms the Cobalt model in terms of WER (8.2% vs 12%) and number of parallel calls processed in real time (96 calls vs 48 calls).

Legacy engine

Warning: the results here are only preliminary. We need to wait until the engine is fully integrated into the LV stack to run these tests again.

We ran a few of the tests from the legacy ASR suite to handle grammar parsing and custom pronunciations. The results are as follow:

Commented [SH29]: Comma should be removed

Test Domain	Dialect	Legacy WER	Preliminary DNN WER
Restaurants	US	6.7 %	2.6 %
Call Router (Tuned)	US	2.7 %	1.8 %
City State	US	7.3 %	6.6 %
Digits	US	2.5 %	0.9 %
Digits	AU	2.9 %	0.2 %
Digits	IN	3.2 %	1.1 %

According to these preliminary results, the new engine outperforms the legacy ASR on all the tests we ran.

Conclusions

Here is a summary of the key parts of this white paper.

- Technologically, there is no turning back. The new ASR delivers a level of accuracy in word recognition that the legacy technology cannot match.
- Legacy systems may seem to perform well enough the short run, but deep learning with CNN has the advantage of greater knowledge placed into the network up front to the benefits of customers and users. The result is a more expert, and more accurate, speech engine.
- The performance of legacy HMM models tend to be capped at a certain level; beyond that additional training produces diminishing returns.
- Our state-of-the-art ASR engine also benefits from other technological improvements, such as compact storage of search trees containing n-grams and one-pass technology. Such improvements as these allow the engine to better predict what combinations of words were intended, thereby leading to fewer mistakes in recognizing words. It also creates a more responsive system without noticeable latency.
- End-to-end Deep Learning eliminates reliance on post-processing of phonemes to do the final translation into text (a vital step). Therefore, a single ASR, without retraining, can handle a significantly larger domain of input. There is no need for a separate engine for American vs. British vs. Australian English—or even differences in regional accents. This is a significant benefit for customers. The system is not going to suddenly fail when a person with a different dialect gets on the line.

Commented [SH30]: Remove the word "in"

Commented [SH31]: Remove comma

Commented [SH32]: "Customers" instead of "customers"