The AT&T Speech API: A Study on Practical Challenges for Customized Speech to Text Service

E. Gouvêa, A. Moreno-Daniel, A. Reddy, R. Chengalvarayan, D. Thomson, A. Ljolje

AT&T Labs – Research, Bedminster NJ, USA

{gouvea,antonio,aarthi,rathi,thomson,alj}@research.att.com

Abstract

AT&T has recently opened its extensive portfolio of state-of-the-art Speech Technology to external end-developers as a platform called "The AT&T Speech API". This study discusses a series of practical challenges found in an industrial deployment of speech to text services, particularly, we examine different strategies for customizing the speech to text process by considering intrinsic factors, inherent to the audio signal, or extrinsic factors, available from other sources, in an industry-grade implementation.

Index Terms: speech recognition, api, customization

1. Introduction

Powered by AT&T Watson^{SM}[1], the recently released AT&T Speech API [2, 3, 4] is a platform that exposes AT&T's state-of-the-art speech technology to external end-developers, allowing them to quickly build applications leveraging AT&T's expertise in the area. As part of the AT&T API platform, the AT&T Speech API is supported by the AT&T Developer Program that provides toolkits, sample codes, etc. The AT&T Speech API resides in AT&T's Silver Lining Cloud. It accepts POST requests via the HTTP protocol as a RESTful service. The services offered by the AT&T Speech API are multi-lingual, carrier-independent (*i.e.*, end-clients do not need to be AT&T subscribers), and device-independent; therefore end-applications are free to run on smartphones, game-consoles, or even home appliances.

One important aspect of tailoring the contexts to the different domains entails providing resources to end-developers that allow them to fine tune the Speech API for each request. This fine tuning involves several facets that we view broadly as *customization*. In our study, customization includes selecting the acoustic environment, as well as adaptation to the language-domain, possibly using metadata available at the time of the request. The metadata may include location information for a mobile application, phone-numbers and names of both caller and callee for a voicemail to text application etc.

We describe some background information about the AT&T Speech API in Section 2, and discuss customization of the acoustic model in Section 3, and of the language model in Section 4. We conclude in Section 5.

2. Background

End-developers have two methods for customizing Speech API to their own application. The first method is to select any of the built-in contexts provided by the API. There are language-contexts like web-search, voicemail-to-text (VMTT), question-answering, gaming, etc.; and there are acoustic-contexts like smartphone, far-field or PSTN microphones. The second method is to modify the recognition network on-the-fly by submitting *inline-content*.

2.1. Dynamic Hierarchical Language Models

Finite State Machine (FSM) [5, 6] is the framework upon which AT&T Watson^{SM} is implemented, fundamental for the dynamic and efficient combination of multiple recognition networks. *Dynamic Hierarchical Language Models* (DHLMs) is an FSM-based technology that allows on-the-fly combination of Language Models (LM) without any need to recompile them into a new static network. The LMs that make up a DHLM can be of any kind, including *statistical language models* (SLMs), *context-free grammars* (CFGs), and other DHLMs.

2.2. Inline-Content

One of the methods for customizing the AT&T Speech API is to share *inline-content* via the HTTP POST message. End-developers may submit POST requests to the AT&T Speech API by packaging the data into a valid MIME message. The content-type `multipart` allows the message to convey multiple sections, where inline-content can be attached to the message that includes the audio-data. The combination of DHLM and inline-content makes on-the-fly customization possible.

3. Acoustic Customization

Given that the AT&T Speech API is device independent, we anticipate diversity in acoustic conditions. For example, while it is reasonable to presume that handheld devices will be placed at a distance shorter than the average human arm-length (60 cm), we cannot do so for devices like a TV set. To address this, end-developers may select the microphone-type in each request; namely *PSTN*, *smartphone*, or *far-field*. The AT&T Watson^{SM} has the capability to load the corresponding acoustic-model efficiently on-the-fly without incurring in any noticeable delays.

To assess the effectiveness of our far-field acoustic-model, we performed an evaluation on WSJ-1 data. We loaded the standard TCB05CNP language model, and re-recorded[1] the si_et_h2 evaluation set at a distance of 2 m.

Table 1 shows how the word error rate raises quickly as the microphone is moved away from the speaker while still using a conventional smartphone *acoustic-model* (AM). Our customized AM adapted to far-field conditions, however, recovers around 85% of that loss. Additionally, close-talk recordings have a small (roughly 10% relative) increase in error even in the mismatched condition when the far-field AM is loaded.

[1] We re-recorded the far-field test-set at random times of the day in a living-room setup with an outside (but closed) window.

Table 1: Relative word error rate increases for WSJ evaluation under different acoustic conditions.

		Mic-Type (AM)	
		Smartphone	Far-field
Mic-Distance	Close-talk	0	9.9%
	Far-field[1]	179%	25.5%

4. Language Customization

The AT&T Speech API allows a number of configurations in which end-developers may customize the language-space accepted by the API's recognition-network at run-time. The simplest configuration is called `x-grammar`, and it lets end-developers replace the API's built-in recognition network with their own. The submitted recognition networks must be CFGs in SRGS format sent within a multipart message, and precede the audio (speech) part. Alternatively, end-developers may take advantage of the API's built-in SLM recognition network, and supplement it with inline-content in the form of an SRGS formatted CFG. Such inline CFG can be placed *in series* with the built-in SLM (`x-grammar-prefix`), placed *in parallel* with the built-in SLM (`x-grammar-altgram`), or placed inside the built-in SLM as an inserted-loop (`x-grammar-loop`).

The `x-grammar` configuration is useful for command-and-control applications where end-users are prompted with a list of phrases to speak. The `x-grammar-prefix` is designed for open-ended phrases where initial carrier-phrases are known, *e.g.*, "`send message` *hello mark*". The `x-grammar-altgram` supplements the built-in SLM with a language containing a set of close-ended phrases known to be important. The `x-grammar-loop` is free to switch between the built-in SLM and the inline-CFG as many times as needed. This is useful when extra information is available that may be useful to the decoder. In a voicemail task, for example, the caller is more likely to say their phone number than other arbitrary number sequences. The decoder may use this information to boost hypotheses where the caller phone number is present.

4.1. Case Study: Voicemail to Text

As a proof of concept, we evaluated the `x-grammar-loop` type with one of the contexts available from the AT&T Speech API, Voicemail-to-Text (VMTT). In our experiments, we sent CFGs containing the caller and callee phone numbers, as these are available as metadata to the carrier when the voicemail is recorded. Our test set consisted of actual voicemail messages collected by an industrial-level deployment.

For our evaluation, we tagged occurrences of phone numbers in each voicemail message. A *phone number* tag was used if the 7- or 10-digit number sequence that was spoken in the message matched either the caller or the callee phone number. An *other number* tag was used if a 7- or 10-digit number sequence was spoken, identifying a phone number, but this number did not match either the caller or the callee phone number. In this case, the phone numbers provided via the `x-grammar-loop` cannot help but should not hinder performance. The tagged data was considered correct only if the recognized sequence matched the whole tagged sequence, *i.e.*, the complete phone number.

We also explored the case where the SLM itself was modified to take advantage of metadata sent via a CFG at recognition time. We found instances of 7- to 10- digit number sequences in the LM training corpus and replaced these instances with a PHONE tag. We created an SLM with this modified training corpus. During recognition, the CFG replaces the PHONE tag wherever it appears in the SLM, as if the CFG were embedded in the SLM. We refer to this approach as the *embedded* approach. The main difference between using the unmodified SLM and the embedded one is that with the unmodified SLM, the language history is discarded when we transition from the SLM to the CFG and vice versa, whereas in the embedded approach, the history is preserved.

Table 2 presents the tagged data accuracy. The results are aggregated by the type of tagged data, *i.e.*, phone numbers or other numbers.

Table 2: Tag accuracy (%) aggregated by the tags *phone number* or *other number*, *i.e.*, phone numbers not in the metadata.

Approach	Phone number	Other number
Baseline	86.6	83.1
Loop with SLM	88.0	83.1
Embedded in SLM	96.1	83.1

The tagged data accuracy shows some differences among the approaches. Reassuringly, the accuracy of phone numbers not in the metadata was not affected. The embedded CFG has an accuracy that is dramatically better than with other ones. The better use of context during recognition seems to contribute to this result.

The use of embedded metadata, however, requires completely retraining the SLM, which is not always practical. Even if it is to a modest extent, the DHLM with Inline Content placed in a loop with the SLM improves over the baseline. It is also worth mentioning that the settings presented here were chosen to keep the word error rate at better or same level as the baseline. We could have traded off some WER in favor of better tagged data accuracy.

5. Conclusions

Our study depicted a series of real-world challenges, and presented practical solutions that lead to improvements in our currently deployed system within an industrial setting. The need for supporting accurate ASR even in diverse acoustic environments was executed in collaboration with current customers. The ability of AT&T Watson^{SM} to customize the recognition network on-the-fly, taking advantage of metadata or inline-content, allows end-developers to personalize each ASR transaction by sharing ephemeral copies of information. The results presented here represent the initial stages of our work in customization that have already been deployed to customer and target a large scale usage.

6. Acknowledgements

We thank D. Caseiro and C. Galles for several helpful discussions.

7. References

[1] V. Goffin, C. Allauzen, E. Bocchieri, D. H. Tur, A. Ljolje, and S. Parthasarathy, "The AT&T Watson speech recognizer," in *Proc. ICASSP*, September 2005.

[2] J. Donovan, "AT&T Speech API Gives Developers the Power of AT&T WATSON," July 2012. [Online]. Available: http://www.attinnovationspace.com/innovation/story/a7782925

[3] G. Di Fabbrizio, T. Okken, and J. G. Wilpon, "A speech mashup framework for multimodal mobile services," in *Proc. Int. Conf. Multimodal Interfaces*. ACM, 2009, pp. 71–78.

[4] I. Arizmendi, S. Parthasarathy, and R. C. Rose, "System and method for speech recognition services," May 4 2010, US Patent 7,711,568.

[5] M. Mohri, F. C. N. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[6] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," *Handbook on speech processing and speech communication, Part E: Speech recognition*, 2008.