Proceedings of the LREC2020 Industry Track , pages 10–14
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Industrial ASR Troubleshooting Tool

Oleksandr Solop, Filip Sawicki, Andrzej Jeziorski, Marcin Sikora
Michał Junczyk, Tomasz Zietkiewicz
Samsung Research Institute Poland (SRPOL)

Abstract

During the development of an industrial scale Automatic Speech Recognition (ASR) system significant quantities of evaluation data have to be analyzed in order to make positive adjustments to the ASR subsystems. The scale and variety of the evaluation data, as well as the complexity of the ASR system architecture makes both manual improvements and automatic model boosting a challenging task. We present a system created to facilitate more effective improvement of the ASR recognition rate with the help of a web tool that provides aggregation, quality review and statistical analysis of ASR performance evaluation data in a human readable format. This paper presents an overview of the system and some exemplary use cases.

Keywords: ASR, error rate, tokenization, alignment, data pipeline, evaluation data

1. Introduction

Evaluation and development of industrial scale, machine learning based systems like Automatic Speech Recognition (ASR) is a challenging task due to quickly evolving product requirements and dependencies between data and system components, often with hidden feedback loops. ASR systems are evaluated in an iterative manner and as a result produce large amounts of evaluation data, which is hard to explore manually. One of the approaches to increase the efficiency of development is to focus on the issues which most significantly impact performance.

The ASR Troubleshooting Web Tool is made to address the above challenge. The tool aggregates tokens extracted from evaluation data and produces a pool of words that have a high probability of being attributed to a common error. These tokens are called "troublemakers". ASR experts can use the tool to mark issues which are hard to automatically detect and suggest corrections to models by monitoring top troublemakers in each ASR system version.

2. Literature Review

The need exists for industrial systems that continuously improve ASR, as seen in the pending US patent by call analytics firm Marchex Incorporated. The patent specifies a system that automatically provides faster and more accurate ASR for telephone calls. In addition to automated model refinement the cited patent provides a human interface for the manual review of transcriptions. Our system not only provides efficient way to detect quality issues in test data, but also provides an extended view of evaluation results thanks to data statistics, aggregated around tokens. (Marchex, Inc., 2018)

In more general terms, our work addresses problems related to machine learning pipelines deployed in a production environments. Many academic publication have been made on the topic. For example D. Sculley et al. specify how technical debt creeps it's way into machine learning systems. (Sculley et al., 2015) Our solution mitigates some of the technical debt issues listed in the article, e.g. the increased cost of analysing an individual model due to it's dependency on models preceding it in the pipe. B. Nushi et al. go more in depth into the problem of troubleshooting component-based machine learning systems and propose a human-in-the-loop approach to model refinement.(Nushi et al., 2017) In their case study on Automated Image Captioning they rely on crowdsourcing microtasks to fine-tune components. Our approach provides a boarder set of information and allows experts to make sweeping adjustments to the pipeline elements.

3. Materials and Methods

3.1. ASR evaluation data overview

An ASR test case (ASR TC) is a pair consisting of an audio/speech recording and a corresponding transcription. An organized collection of ASR TCs in a standardized format, with relevant meta-data (e.g. language, domain, acoustic conditions etc.) constitutes an ASR database (ASRDB). An Automatic Speech Recognition (ASR) system generates an orthographic text representation (hypothesis) to an audio input (audio recording). The quality of an ASR system is evaluated by comparing the similarity of the hypothesis generated for a given audio recording to a manually created transcription (reference) of the corresponding

recording, across a given set of ASR TCs. Depending on the ASR subsystem being evaluated, either a basic reference or a normalized reference is used. The basic reference contains only tokens and is used to evaluate the performance of Language Models (LM), while the normalized reference is used for Inverse Text Normalizer (ITN) evaluation, therefore it should resemble text written by humans and respect language specific normalization rules related to numeric expressions formatting, spelling of named entities, punctuation etc.
In order to generate ASR hypotheses resembling text written by humans a combination of various modules is used (see next section).

3.2. ASR system overview

The ASR system is comprised of a long short-term memory (LSTM) Acoustic Model (AM), Grapheme-to-Phoneme (G2P) model, n-gram Language Model (LM), optional n-gram Personal Language Model (PLM) and Inverse Text Normalizer (ITN) post-processing modules. All the models have been trained using *proprietary* data. An in-house built decoder has been used to generate 1-best hypothesis. The hypothesis is then processed by different ITN submodules, including:

- Preprocessor - removes tags inserted by decoder (such as start and end of utterance tags), normalizes casing by the decoder output
- Inverse Number Normalizer (INN) - performs inverse normalization of numbers by replacing numerals with appropriate numbers.
- Fixers - perform string substitutions, either ones defined by language experts as regular expressions patterns or performed automatically, by statistical models trained on ASR evaluation data.

ITN modules are connected in the form of a pipeline. Each of them can be selectively turned off. During the ASR evaluation ITN is tested multiple times with different modules turned on and off, to check them separately and provide a way to spot the exact cause of possible errors. This, together with the PLM module also being turned on and off, leads to production of different ASR hypotheses. ITN specification in the industrial ASR system needs to be quickly adapted to evolving user needs, language conventions, market trends and system architecture. Given the inter-subjective nature of language, hidden/complex organizational dependencies with sometimes conflicting requirements, it's necessary to manage the ITN functional requirements as a common "ASR text normalization protocol", available to all stakeholders, including product owners, ASR developers, testers and language experts responsible for creating references.

3.3. Data Aggregation

In order to provide a human readable view of the complex ASR evaluation pipeline, the system uses data from ASRDB evaluation sets and corresponding ASR hypotheses generated with various system settings. If a hypothesis has a word recognition rate of $x_{wrr} = 1$ compared to the normalized reference (ref-n), then it is considered to be correct, else there is a mismatch and the pair is marked as incorrect.

The Troublemaker Tool was designed using the ETL (Extract, Transform, Load) concept to generate the best data pipeline performance using available resources. The system saves only structured, filtered and polished outcomes instead of large amounts of ASR test results in various formats.

System ETL steps
1. Data loading, validation and filtration
2. Collection of mismatched tokens
3. Calculation of aggregated statistics on tokens
4. Extraction of troublemakers
5. Comparison of mismatched utterance pairs
6. Concordance and stability metric preparation
7. Saving processed data to database

During the data processing stage, the ETL system collects and counts all tokens that appear in mismatched utterance. These tokens are grouped together and used to calculate aggregated statistics. Based on the gathered metrics our system selects troublesome tokens that have both high fail frequency and the greatest influence on ASR performance. Fixing errors in tokens with the highest combination of total count, stability and frequency yields the most significant improvements in ASR. We have created several metrics to make detection of these tokens easier (see section "Troublemaker Metrics").

To make an ASR improvement suggestion with the Troublemaker, the reference and hypothesis has to be compared to find differences. Our tool makes use of a sequence matching algorithm to reject equivalent tokens from both sentences. A pairwise sequence alignment algorithm is then applied to find the best matching tokens based on the received similarity score. Moreover the entropy value for each token is calculated. It shows how spread out the error is and thus gives information on how meaningful the potential fix would be.

3.4. Troublemaker Metrics

TokenCount: the number of ASR TCs containing a given token. This metric helps to prioritize work by providing information about the "troublemaker" token's impact based it's popularity. x is equal to 1 if the token is present in the sentence i and 0 otherwise. n is the total number of ASR TCs in the evaluation set.

$$\text{TokenCount} = \sum_{i}^{n} x_i$$

FailsCount: the number of incorrect hypotheses related to the token. Incorrect hypotheses are those with a word recognition rate $wrr < 1$. $x_{i;wrr<1}$ is equal to 1 when the token in present in the sentence and the sentence is from a failing ASR reference-hypothesis pair.

$$\text{FailsCount} = \sum_{i} x_{i;wrr<1}$$

Frequency: Ratio of incorrect hypotheses to all ASR TCs containing the token. Shows an estimate of how often the token is failing.

$$\text{Frequency} = \frac{\text{FailsCount}}{\text{TokenCount}}$$

Entropy: Measure of the level of uncertainty associated with a given token. F is the Frequency of the token failing.

$$\text{Entropy} = -\left(\text{F} \cdot \log(\text{F}) + (1 - \text{F}) \cdot \log(1 - \text{F})\right)$$

Fail Coefficient: a modified frequency metric that reduces the number of most common tokens shown and include less frequent ones. Higher values indicates bigger impact on ASR system performance. User can sort troublemakers by this field to work with the most important defects first.

$$\text{Fcoeff} = \text{Frequency} \cdot \log(\text{TokenCount})$$

FCoeffR: the logarithm of the token's rank among all of the tokens based on Fcoeff values.

EntropyR: the logarithm of the token's rank among all of the tokens based on Entropy values.

Wrank: a weighted rank based on a combination of Fail Coefficient and Entropy ranks. The metrics tries to balance between token frequency, replacement stability and overall incorrectness. It is used as a main metric for comparison of different ASR system versions. The lower the rank, the more the token is worthy of the user's attention. A_1 and A_2 are adjustable weight constants that effect the ratio with which FCoeff and Entropy effect Wrank.

$$\text{WRank} = (A_1 \cdot \text{FCoeffR}) \cdot (A_2 \cdot \text{EntropyR})$$

4. Triage Process

Every ASR data report contains a change rate based on the aggregated troublemakers *Wrank* metric, which makes it easy to compare performance of any pair of ASR system versions. This simplifies results analysis, because it allows the review of data by portions.

Language experts have access to multiple tables describing top tokens per hypothesis. These tables include all metrics, which are used to locate troublemakers that are the most cost-efficient to fix, and the number of reported issues related to them. Moreover, our tool can represent two kinds of troublemakers: the reference troublemakers (tokens, which ASR rarely recognizes correctly) and hypothesis troublemakers (tokens, which ASR outputs too often). This error type separation and issue detection prevents the ASR system from over-fitting and under-fitting on these troublemakers.

After the user selects a troublemaker to work with, the concordance view opens, enabling the checking of details about a troublemaker's word stability, for example all mismatched tokens with their probabilities.The concordance view also contains a pool of random utterance pairs (reference and hypothesis) containing the selected troublemaker. A single example row consists of information about WRR, a button for audio recording playback and comparisons of hypotheses from other stages of the ASR pipeline. All troublemaker words and potential replacement words are presented in a joint column to facilitate the process of issue analysis.

Reference	wrr	Hypothesis
The cat sat on the mat	0.77	The dog sat on the mat
The big cat on the mat	0.65	The big dog sat on the mat
...		

Figure 1: Concordance view example.

An expert can report issues related to the troublemaker token using the report panel. In this panel, one can quickly assign the troublemaker to several categories, depending on the issue type. Additionally one can provide a detailed description of the problem and tag it with other relevant tokens. Different issue categories are then automatically routed to appropriate developer teams which are specialized in fixing the particular problems. An exemplary set of problem classes for a typical ASR system has been described below. This set can differ vastly, depending on the ASR system architecture.

Audio: a recording may or may not meet quality requirements, depending on the purpose of a particular ASRDB. For example, audio with an above normal

distortion level, can be reported if present in a regression test set which supposed to contain only clean audio data. In other contexts it can be a useful example of a real usage scenario. Other fairly common issues are excessive cross-talk, incomplete or unintelligible speech.

Reference: since ASRDBs references are prepared by language experts, they are prone to human errors. The typical problems are: incorrect normalization, typos and the unnecessary transcription of background speech.

AM: poor recognition in certain acoustic conditions, systematically substituted tokens with similar pronunciation or sensitivity to background speech – these all can be attributed to a problem with AM.

Reference	wrr	Hypothesis
le président de l'égypte	0.25	les présidents de l'égypte
président de france	0.67	présidents de france
...		

Figure 2: Concordance AM issue example

Grapheme-to-Phoneme (G2P): improper phonetization can lead to errors, especially when it comes to Named Entities (NE).

Reference	wrr	Hypothesis
hi bixby	0	hi pixy
bixby was ist das für ein lied	0.33	xd was ist das für ein lied
...		

Figure 3: Concordance G2P issue example

LM: a broad category, which can be as simple as a lexicon typo, or more complex, signaling a broader issue with NEs, problems with number recognition, and many others.

Reference	wrr	Hypothesis
ponme un álbum de camilo sesto	0.83	ponme un album de camilo sesto
ponme un álbum de joe dassin	0.83	ponme un album de joe dassin
...		

Figure 4: Concordance LM issue example

ITN: a fairly well defined category consisting of general normalization, capitalization and punctuation issues.

Reference	wrr	Hypothesis
playstation 4	0	play station cuatro
enciende la play playstation	0.33	enciende la play play station
...		

Figure 5: Concordance ITN issue example

5. System architecture

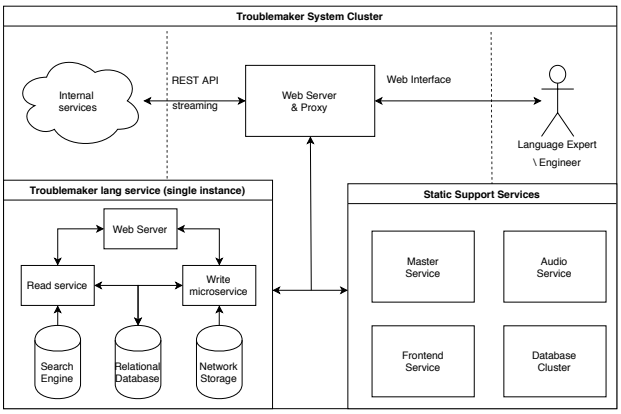

Figure 6: System architecture.

5.1. Microservice architecture

The Troublemaker Tool follows a microservices design pattern. In front of the system sits a web server that acts as an API gateway, reverse proxy, offloader and load balancer for incoming REST requests. Next there are two main groups of services. Static Support Services serve as gateway aggregators, health checkers, frontend interfaces, data management systems and provide other common utilities, like audio streaming, logging and issue reporting. Scalable Language Services consist of multiple instances of read and write microservices deployed for one specific language.

6. Conclusion

The presented system aids the development of industrial scale ASR by extracting ”troublemaking” tokens that most frequently cause errors in the ASR system or reference data, presenting information in a convenient format for ASR experts and streamlining the generation of reports. Given the early stage of development, we’re not able to measure the impact of the system on the process in a quantitative way yet. However the qualitative feedback received so far is encouraging to pursue further development and research in this area.

7. Bibliographical References

Marchex, Inc. (2018). Automatic speech recognition (asr) model training. US Patent 20180315417.

Nushi, B., Kamar, E., Horvitz, E., and Kossmann, D. (2017). On human intellect and machine failures: Troubleshooting integrative machine learning systems. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden

technical debt in machine learning systems. In C. Cortes, et al., editors, *Advances in Neural Information Processing Systems 28*, pages 2503–2511. Curran Associates, Inc.