SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

From Support Calls to Insights

Using Automatic Speech Recognition and Natural Language Processing to Drive Product Roadmaps

A Technical Paper prepared for SCTE by

Jing Qing
Principal Data Scientist I
Charter Communications
6380 S Fiddlers Green Circle, Greenwood Village, CO 80111
Jing.Qing@charter.com

Veronica Bloom
Director of Data Science
Charter Communications
6380 S Fiddlers Green Circle, Greenwood Village, CO 80111
+1 720-699-3798
Veronica.Bloom@charter.com

Michael Addonisio
Lead Data Scientist
Charter Communications
6380 S Fiddlers Green Circle, Greenwood Village, CO 80111
Michael.Addonisio@charter.com

Christy Gearheart
Data Scientist IV
IntePros Consulting, Charter Communications
6380 S Fiddlers Green Circle, Greenwood Village, CO 80111
C-Christy.Gearheart@charter.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

The goal of this project is to utilize artificial intelligence (AI) and machine learning (ML) techniques to gain insights into the drivers of support calls. These insights are used to determine areas of improvements to both processes and products to enhance our customer experience.

In this paper, we discuss the end-to-end automated pipeline to go from call center data to actionable insights utilizing machine learning techniques such as Automatic Speech Recognition (ASR) and Natural Language Processing (NLP). Discussion of the pipeline architecture, use cases of automated call disposition, and call topic trend analysis are also included. Broadly, the pipeline converts recorded call audio conversations to text transcripts using automatic speech recognition, then uses those transcripts to train classification models to predict call disposition. Unsupervised topic modeling extracts new trends and topics from the call-volume data over time to identify new or emerging call drivers. These call drivers subsequently drive new feature development and product roadmaps.

Please note: Charter has a longstanding commitment to protecting the privacy and security of its customers. To provide our customers with technical support and high quality customer service as well as to determine areas of improvement, customers are informed that communications between customer service agents and customers may be recorded when authorized by the customer for quality and training purposes.

2. Technical Approach

2.1. Call disposition Flow Overview

Current call dispositions are collected through a manual process in a call tracking tool, where call center agents select a call disposition and enter notes into the system. If a call is transferred to another agent, it is split into call segments where each agent selects a disposition for their corresponding segment. Dispositions are predefined, and selected via drop-down menu.

While this approach enables quick selection and consistent responses, there are some limitations. Selection of disposition is subjective and can vary from agent to agent. Only one disposition can be selected per call, even when multiple issues are discussed. Changes to the disposition response list requires both a software update and agent training, and cannot be applied retroactively. Furthermore, due to the nature of a manual dispositioning process, some call segments are missing a recorded disposition in the tracking tool.

These limitations can be addressed through a machine learning approach. Using an automated machine learning approach enables scalable, consistent, and reliable call disposition labeling. Nuances in how an individual agent labels a given call segment are removed, as all call segments are systematically evaluated. Furthermore, assignment of multiple labels to a single call can bring to light underlying factors affecting multiple call reasons.

2.2. Machine Learning Pipeline Overall Architecture Review

We have developed a pipeline for data processing and modeling, in combination with cloud AI services such as cloud transcribe and cloud NLP services for an end-to-end call disposition architecture. Additional call insights are gained through the implementation of our topic modeling framework. The pipeline is implemented in Python 3 through boto3 (Amazon Web Services 2015). See Figure 1.

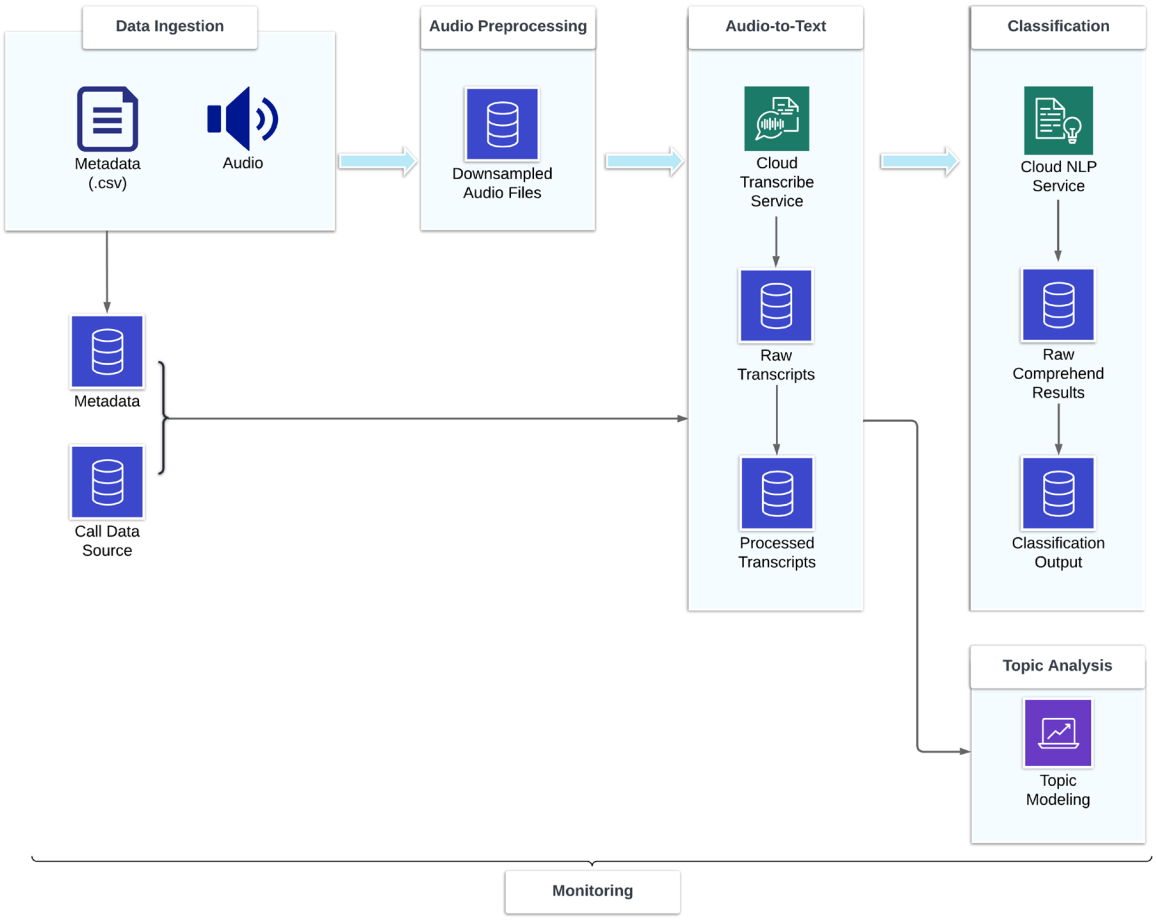

Figure 1 – ML Pipeline Architecture Overview

2.3. Data Ingestion

Every morning, call audio recordings and associated metadata from the previous day are ingested into the data science cloud storage.

2.4. Audio Preprocessing

To prepare the audio data for the cloud transcribe service for the best transcript output, we preprocess audio recordings after the audio files are ingested. In this step, we downsample the audio to a sampling rate of 8,000 Hz, and convert the audio files into .wav format through a parallel batch process. The downsampling and conversion are implemented through the open-source library librosa (McFee, et al. 2022). This pipeline is also batch parallelized and implemented through joblib (Joblib developers 2020).

2.5. Convert audio into text transcripts

There are multiple automatic speech recognition services available on the market to convert audio to text transcripts. In our pipeline, we utilize a cloud transcribe service to convert downsampled audio files into text transcripts. We harness built-in features such as custom vocabulary to improve the transcript quality, channel identification to add data dimensionality for analysis, and transcript redaction for data privacy and compliance. We experimented with different settings and decided on the configuration for production

based on the best word error rate (WER) evaluation metric. We describe these features and the WER evaluation in the next sections.

2.5.1. Custom vocabulary

Using custom vocabulary can help to improve transcribe accuracy for domain specific words and phrases. We worked with subject matter experts (SME) to identify technical vocabulary words and phrases commonly used in call center conversations for our use case. With sound-alike and phonetic annotations, we created a custom vocabulary for the cloud transcribe process to ensure Spectrum specific terms are represented correctly in the final transcripts.

2.5.2. Channel identification

Since the audio recordings contain two audio channels, we are able to generate transcripts for the full conversation, as well as separated transcripts by each audio channel. The transcripts by channel could be used in further analysis, such as associating topics in different channels to call issues and resolutions (Pratik K. Biswas 2021).

2.5.3. Redacting transcripts

We enabled redaction in the cloud transcribe service, where Personally Identifiable Information (PII) such as name and address are removed from the transcript output.

2.5.4. Word Error Rate (WER)

Speech-to-text is quite challenging for call center audio recordings. The audio quality is usually low, and there could be background noise such as a TV playing, dogs barking, or the customer asking questions. There are at least two people on the call, sometimes talking at the same time, and the speaking style is conversational and usually loosely structured. Thus, it is crucial that we measure the quality of the speech-to-text translations.

The most common metric for speech recognition accuracy is called word error rate (WER) (Seyfarth and Zhao 2020). WER counts the number of incorrect words identified during recognition, and divides the sum by the total number of words provided in the human-labeled transcript (N). WER = (I + D + S)/N, where I stands for Insertion Error: words incorrectly added in the transcript; D stands for Deletion Error: words undetected (deleted) in the transcript; S stands for Substitution Error: words substituted between reference and hypothesis. The lower the WER, the more accurate the system.

We randomly selected 15 transcripts and labeled them to evaluate WER. The WER ranged between 5% and 43%, with an average WER of 19%. By comparison, a benchmark analysis of nearly 3,000 call center conversations across five domains transcribed with automatic speech recognition (ASR) tools versus transcribed by two professional annotators found WER for call center conversations within the telecommunications domain to be between 17.62% and 23.31% (Szymański, et al. 2020).

2.6. Classification

2.6.1. Identify call dispostion categories

The data science team worked with SMEs to identify eight high-level categories to train a classification model to predict call dispositions.

2.6.2. Multi-label classification approach

In the multi-label classification, individual classes represent different categories, but these categories are somewhat related and not mutually exclusive. As a result, each document has at least one class assigned to it, but can have more. In call center calls, a conversation can discuss multiple topics, justifying the use of multi-label classification.

We implemented the multi-label classification model through a cloud NLP service. For each input transcript, the model returns three labels with the highest predicted scores. Note, in some cases, even the highest predicted score is low if the model is not confident in the classification of any of the eight categories. Therefore, we applied category-specific score thresholds to the predicted labels, and only keep labels where the score is higher than the threshold as the final predicted label(s).

2.6.3. Create labels through LabelStudio

To train a classification model, it requires us to provide the cloud NLP service with examples of call transcripts with their appropriate classification labels. In order to do this, the data science team worked with SMEs to understand what to look for in calls in order to provide example labels. This team listened to approximately 1,000 calls in order to provide examples of about 100 calls for each category.

To support labeling the call data, we set up a cloud instance to run Label Studio (Tkachenko, et al. 2020), an open source annotation tool. Label Studio allowed us to create a custom interface that includes both audio and text media for annotation. The annotator was required to label each document with at least one label and optionally up to three labels. Each document was labeled at least twice, and, if there was no conesenus, a third time. An example of the annotation layout can be seen in Figure 2. We consider the call categories labeled in Label Studio through this process as the “ground truth” data.

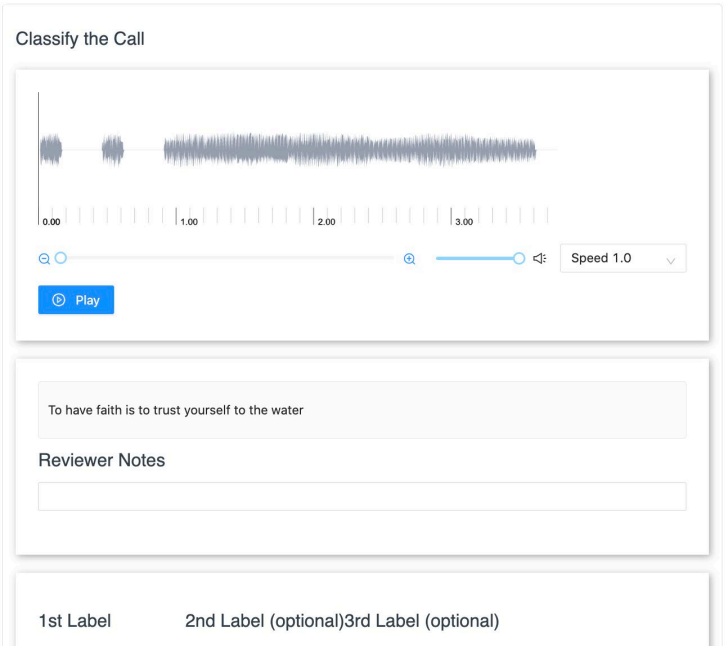

Figure 2 – Label Studio call labeling interface

2.6.4. Train Comprehend classification model and model evaluation

Once we obtained approximately 100 example calls for each category, we split our transcripts into 675 files for training and 191 files for validation. We experimented with various transcribe and comprehend configurations in the pipeline to determine precision and recall scores, and combined them into a single representative micro F1 score. A micro F1 score balances precision and recall across individual categories and provides a single quality metric for multi-label binary problems. The best performing comprehend model has a micro F1 score of 60% on both the training and validation set. Finally, we evaluated the fit of our model based on a random selection of 90 transcripts of previously unseen data, and examined the model's eight categorical predictions compared to the manually-curated call dispositions supplied by our call center agents.

We found the classification model trained with the cloud NLP service obtained a micro F1 score of about 59% on the evaluation set across the categories, with some categories performing better or worse than others. For example, one category obtained an F1 score of about 84%, while another category obtained an F1 score of about 25%. We also found the classification model performed very similarly to agent-based dispositions (58%) against labeled ground truth data, with similar performance across categories. See Figure 3. Future work will focus on improving the comprehend model with additional labeled data and refining the classification categories.

Figure 3 – Agent vs. ML classification F1 score performance

2.6.5. Inference

After training the classification model, we ran the model on a month of call transcripts from one month to understand which of these eight categories are driving the majority of calls. Recall that calls can be classified in multiple categories. We found the majority of calls for our use case, 40%, came from requiring support in one specific area. We also see a large portion of these calls, 27%, fall into the "Other" category. In the next section, we show how utilizing topic modeling can give us more granular

insights into the call drivers in the Other category, and will break down the top category into actionable subcategories.

2.7. Topic Modeling

When we shared the ML disposition results with stakeholders, they immediately wanted to know if we could find out more granular call reasons and if we could identify improvements to features and workflows to provide improved product experience. To answer these questions, we applied topic modeling to further analyze the call transcripts.

Topic modeling is an unsupervised method for discovering topics in a large collection of documents. Topics can then be used to find high-level summaries of these documents, search for documents of interest, and group similar documents together. In practice, we implemented the open-source library Top2Vec (Angelov 2020) for our analysis. Some benefits of the Top2Vec algorithm include automatic identification of the number of topic clusters and the ability to identify topic hierarchies.

The topic modeling process includes the following steps: text preprocessing such as normalization, training the topic model, interpreting topics, and visualizing and analyzing trends.

2.7.1. Text normalization

Text normalization is the process of transforming text into a single canonical form, so it is "cleaner" or less random. We found this step necessary to obtain more interpretable topics. As mentioned earlier, the audio-to-text task is challenging for call center audio files. Even after significant efforts to optimize AWS Transcribe, there are still consistent words or phrases incorrectly represented in the transcripts. Additionally, there are text formats in the transcripts that interfere with the topic model's tokenization process. For example, "DVR" is often displayed as "D. V. R." in the transcript. Since we also use "." as a text delimiter, it would have been split into three separate letters "D", "V", "R", and the individual letters lose the meaning compared to "DVR" as a whole. Through normalization, we are able to fix consistent transcribe errors, and normalize words and phrases so they will be represented correctly for topic modeling.

In order to handle the text normalization task on a large volume of call transcripts, we needed a process that is optimized and parallelizable. We chose HuggingFace's tokenization pipeline (Wolf, et al. 2019) to normalize the data, and trained a Byte-Pair Encoding (BPE) Tokenizer on call transcripts to generate cleaned input documents for the Top2Vec model. We chose this tokenization pipeline because it is an efficient and optimized implementation, allowing us to use all cores on the data science cluster and process the text in parallel, which significantly reduced the processing time.

2.7.2. Training the Topic Model

The Top2Vec algorithm includes the following steps: the first step is to create a joint embedding of document and word vectors. Once documents and words are embedded in a vector space, the goal of the algorithm is to find dense clusters of documents, then identify which words attracted those documents together. Each dense area is a topic and the words that attracted the documents to the dense area are the topic words (Angelov 2020).

For experiments, we tested multiple embedding models, including Doc2Vec (Rehurek 2022), and pretrained embedding models such as universal-sentence-encoder (Cer, et al. 2018) and all-MiniLM-L6-v2 (Reimers and Gurevych 2019). We have found using Doc2Vec to generate the document and word embedding vectors produced the most interpretable topics. This is because call transcripts contain many

Spectrum-specific terms, and the Doc2Vec embedding is trained from scratch using all the transcripts as part of the topic modeling process. Therefore, the Doc2Vec embedding is able to represent Spectrum-specific terms well. While pretrained embedding models, even though they are trained on massive language datasets using complex neural networks, since they have never "seen" Spectrum call data before, they require additional fine tuning to perform well with our call data.

2.7.3. Topic interpretation

We trained the Top2Vec model on a month of call data. Because topic modeling produces unlabeled clusters of call transcripts, we supplemented the model by labeling each topic cluster with the most frequent keywords and bigrams. This provided some contextual meaning to each of the clusters.

2.8. Productionalization and Monitoring

Given the number of moving parts in our pipeline, we have established a monitoring framework to automatically measure data availability, reliability, and operational performance against our baseline. This enables us to identify and correct any arising issues early. Our monitoring framework has been integrated into the pipeline so every execution produces an associated monitoring report.

2.8.1. Data Ingestion

Each day, we receive call audio recordings in a cloud storage repository. Additionally, we receive a daily manifest file listing each audio file expected and its associated metadata, such as unique call identifier, agent information, and source of call. As an initial step to our pipeline, we validate the files received match the files listed in the manifest. Independently, we validate the manifest file contains the calls we expect to receive based on a separate internal call data source. By utilizing automated data validation, we are able to quickly identify and address discrepancies when these numbers deviate from one another. As a result of this monitoring, we are able to quickly identify and resolve this issues that come up during normal course of business.

2.8.2. Audio Preprocessing

We downsample and convert audio files to WAV format for ingestion into a cloud transcribe service. We monitor the number of audio files that failed to convert, the total count of files successfully converted, and the minimum, maximum, average, standard deviation, and percent quartiles for both call lengths and file sizes following downsampling. This enables us to capture any issues in file transfer or conversion in our hybrid environment, and to provide insights into the size and length of the audio files received.

Monitoring became crucial in identifying an underlying issue in our downsampling process. We were finding roughly 100 files were being produced with a zero-length audio file. Interestingly, the files producing the zero length audios were different from run to run. Through troubleshooting, we identified that the pydub python library (Robert, Webbie and others 2018) would intermittently fail to downsample a given file when run as a parallel process. In the end, this was overcome by our transition to the librosa python library.

2.8.3. Convert audio into text transcripts

As files are transcribed, we capture the total number of audio files that fail as well as the individual errors and corresponding number of files per failure reason. Capturing the reasons associated with the failures enables us to see if there are underlying errors requiring remediation. The most common error observed

stems from the file being less than .5 milliseconds, the minimum audio duration required for transcribe. Because we cannot process these files, they are logged and removed, and the pipeline continues execution.

After transcription, we parse the individual files based on their audio channels, join to their associated metadata, and combine into a single dictionary in preparation for classification. In addition to monitoring the total number of processed transcripts, we further monitor the individual word count statistics produced. For example, we monitor the mean word count across the transcribed files; when the word count drops substantially, we are able to identify if the current run has failed to appropriately transcribe the calls.

2.8.4. Classification

At the completion of classification, we validate the output file has been created and its location documented. Summary results are logged indicating the distribution of audio files across our defined classification labels. Insights are delivered to our stakeholders for actionable business intelligence.

2.8.5. Configuration Details

Finally, to ensure we have repeatable results, we log the configuration metrics, including run time, file locations, custom vocabulary file, and any process-specific parameters utilized in the run.

2.9. How AI/ML Insights Drive Product Roadmaps

With quantitative metrics in hand, our stakeholders are empowered with empirical evidence to better guide and prioritize their roadmaps. Broadly, we were able to utilize topic modeling to identify call drivers and use those call drivers to identify areas where improvements would increase efficiency. As such, these topic modeling results are driving current and future product roadmaps and supporting continual product improvement.

3. Future work

Future work is focused on improving the accuracy and efficiency of our pipeline. Using audio transcripts for call disposition is just the beginning. We want to enrich our call disposition data with additional data sources which can open up our analysis to understanding patterns, and will enable us to better understand the broader picture. Improvements to our visualization dashboard will better highlight topic insights derived from the model.

We also plan to experiment using the same tokenization pipeline on call transcripts to train a transformer-based model, such as BERT (Devlin, et al. 2018). As a result of the training process, BERT learns contextual embeddings for words. We can then use the custom trained model to produce embeddings for topic modeling. These BERT embeddings should contain more contextual information and might be able to improve the topic model.

As an unsupervised learning method, topic modeling is great in clustering semantically-similar documents together to show trends; however, it still requires human input for topic interpretation and requires effort to measure output accuracy. Conversely, supervised classification modeling, is easy to set up, but requires high-quality labels for meaningful training and evaluation. As such, we plan to augment our topic modeling process with a human-in-the-loop to create new training labels for the classification model.

4. Conclusion

We have discussed an automated end-to-end pipeline from call center data to actionable insights. In our pipeline, we developed a flow to ingest audio data, performed audio downsampling, converted audio to text, ran a classification model on high-level dispositions, and performed topic modeling for more granular insights. This pipeline is monitored at each step with measurable metrics for quality control.

While the manual call tracking tool collects high-level call dispositions, the machine learning and natural language processing approach is fast, reliable, and scalable. Machine learning elevates us from statistical metrics to deriving deep analytical insights and identifying new trends. These insights help shape product roadmaps and drive new feature development.

5. Acknowledgements

We would like to extend our special thanks to Jonathan Stribley, Sean O'Donnell, and Keara O'Brien for their partnership in these endeavors; the Call Center Technology team for their partnership in data ingestion, Sandeep Kumar Sarikonda, Pablo Lopez, and the Data Science Machine Learning Operations team for their assistance in setting up and maintaining the pipeline environment; the Data Platform team for establishing the development and production environment; Steven Naumann and the Data Engineering team for their assistance and insights with the internal call data; Laurie Porter for listening to hundreds of calls and helping us to understand the business process so we can establish a labeling guideline; Rohan Khatavkar and Aaron Osher for their help in labeling call dispositions; Cody Zeigler and Pierre Dumas from the cloud service team for their expertise and partnership in the proof of concept (POC) of the pipeline and productionalization; Mike Baldino for his support and leadership; and last, but not least, Marjorie Sedillo and Brock Bose for their support, expertise, and guidance in the development of the pipeline and model, as well as their editorial feedback on our manuscript.

Abbreviations

AI	artificial intelligence
ASR	automatic speech recognition
BERT	bidirectional encoder representations from transformers
BPE	byte-pair encoding
ML	machine learning
PII	personally identifiable information
POC	proof of concept
SME	subject matter expert
WER	word error rate

Bibliography & References

Angelov, Dimi. 2020. *Top2Vec: Distributed Representations of Topics.* arXiv. {https://arxiv.org/abs/2008.09470}.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. *Attention Is All You Need.* https://arxiv.org/abs/1706.03762.

Cer, Daniel, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, et al. 2018. *Universal Sentence Encoder.* {https://arxiv.org/abs/1803.11175}.

Devlin, Jacob, Ming-Wei Wang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Oct. https://arxiv.org/abs/1810.04805v2.

Joblib developers. 2020. *Joblib: Running Python Functions as Pipeline Jobs.* {https://joblib.readthedocs.io/}.

McFee, Brian, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, et al. 2022. *librosa. librosa.* {https://github.com/librosa/librosa}.

Pratik K. Biswas, Aleksandr Iakubovich. 2021. *Extractive Summarization of Call Transcripts.* {https://arxiv.org/abs/2103.10599}.

Rehurek, Radim. 2022. *Gensim: Topic Modeling for Humans - Doc2Vec Paragraph Embeddings.* https://radimrehurek.com/gensim/models/doc2vec.html.

Reimers, Nils, and Iryna Gurevych. 2019. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. {https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2}.

Robert, James, Marc Webbie, and others. 2018. "Pydub." GitHub. {http://pydub.com/}.

Seyfarth, Scott, and Paul Zhao. 2020. *Evaluating an automatic speech recognition service.* October 5. {https://aws.amazon.com/blogs/machine-learning/evaluating-an-automatic-speech-recognition-service/}.

Szymański, Piotr, Piotr Żelasko, Mikolaj Morzy, Adrian Szymczak, Marzena Żyła-Hoppe, Joanna Banaszczak, Lukasz Augustyniak, Jan Mizgajski, and Yishay Carmiel. 2020. "WER we are and WER we think we are." *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing (EMNLP) 2020.* Punta Cana: Association for Computational Linguistics. 3290–3295.

Tkachenko, Maxim, Mikhail Malyuk, Nikita Shevchenko, Andrey Holmanyuk, and Nikolai Liubimov. 2020. *Label Studio: Data labeling software.* {https://github.com/heartexlabs/label-studio}.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, et al. 2019. *HuggingFace's Transformers: State-of-the-art Natural Language Processing.* arXiv. {https://arxiv.org/abs/1910.03771}.

