ANSWERING MACHINES DETECTION IN CONTACT CENTERS USING CONVOLUTIONAL NEURAL NETWORKS

Nikolay Anisimov, Mikhail Olkhovsky, Konstantin Kishinsky

Bright Patterns, Inc.
San Bruno, California, USA

ABSTRACT

This paper addresses the problem of applying convolutional neural networks (CNNs) to detect answering machines in outbound contact centers. For a training data set, we use call logs and call recordings of an outbound campaign of a real-world contact center. The data set is processed to extract features and labels of call samples. The features are represented as Mel frequency cepstral coefficients (MFCCs) extracted from initial intervals of call recordings. The convolutional neural network comprises two convolutional layers, two max polling layers, and two fully connected layers with sigmoid at the end. The model demonstrates about 94% accuracy, which corresponds to the industry standard. Some future areas of improvement for the method are identified.

***Index Terms*—** Outbound dialing, AMD, deep learning, classification

1. INTRODUCTION

Customer service has become the main business differentiator in this modern, highly competitive economy. Ideally, every business dealing with customers should have a multi-channel customer contact center. In fact, customer care and contact centers are forming a new and very important industry in the world. Indeed, according to a recent study, more than 5 million Americans work in contact centers in more than 50,000 locations across the country, representing nearly 4% of the U.S. working population[1]. Moreover, 14.5 million call center agents work worldwide [1].

Outbound dialing is an integrated and important part of contemporary call and contact centers [2], enabling businesses to contact their customers pro-actively [3, 4]. There are many useful outbound applications, such as informing customers about the readiness of prescription drugs, notifying customers about new goods and services, conducting political campaigns, and so forth.

Outbound dialing uses a special scenario of a call flow. Initially, the outbound dialing scenario automatically calls a customer and if the customer answers the call, the customer is connected to an available customer service representative or agent. This design optimizes the utilization of agents, which are the most expensive resources of contact centers, by eliminating dialing and waiting times, handling busy situations, and replacing fax and answering machines. Agents are involved in a call only when live customers answer.

One of the challenging aspects of this design lies in distinguishing answering machines from live customers. In the industry, this issue is referred to as answering machine detection [5, 6, 7, 8], and it is addressed with a component known as an Answering Machine Detector (AMD). With answering machine detection, calls accepted by answering machines (AM calls) and calls accepted by living persons (LP calls) are not distinguished by a signaling layer and are determined based on media content only. Fortunately, answering machines and living persons answer calls in different ways, and AMD should distinguish them based on these differences.

In recent years, there is a remarkable progress in application deep learning models [9] to different areas including image processing [10] and speech processing [11, 12, 13]. In deep learning model, Convolutional Neural Networks (CNN) plays an important role as a power tool for visual recognition [10]. At the same time, applications of CNN have been successively expanding to the area of signal processing with input features in a form of spectrogram [14, 15] and raw waveforms [16, 17, 18, 19].

In this paper, we present an approach to answering machine detection based on machine learning techniques using convolutional neural networks.

2. STATEMENT OF THE PROBLEM

2.1. Outbound Environment

The typical environment for the outbound dialing of a contact center is shown in Figure 1. The central part of the environment is a predictive dialer, a set of components responsible for creating and monitoring customer calls.

Outbound calls to customers are created by a call generator that takes the next number from calling lists and determines an optimal time to make a call to this number. The optimal time is determined by a predictive dialing algorithm in

[1]Refer to `http://www.jobs4america.net/facts`

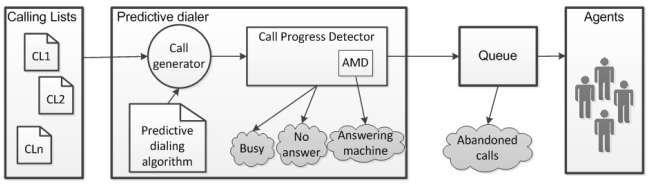

Figure 1: Outbound dialing environment

order to keep agents busy and keep abandonment rate within reasonable borders (e.g., less than 3%). When a new call is generated, it is monitored by the call progress detector (CPD). More specifically, CPD detects no-connect situations, such as busy signals and no-answer.

If, however, the call is answered, the AMD should detect whether the call was answered by a living person or by an answering machine. If an answering machine is detected, the call is dropped or a voice message is left. If the AMD detects a living person, the call is placed in a queue to wait for an available agent. If the call waits in the queue too long, it may be abandoned by a customer who has little patience.

2.2. Heuristic Approach

The algorithm for separating AM calls from LP calls will be referred to as a Living Person Detection (LPD) algorithm. It is related to the AMD algorithm but better stresses the importance of living customers versus answering machines.

In practice, the most popular approach is a heuristic approach based on the observation that answering machines and live persons answer a call in different ways [8, 5, 7]. More specifically, a living person says, Hello, Hi, or Hi there and then waits for a response. Answering machines respond with a long sentence followed by an option to leave a message.

In a heuristic approach, a heuristic algorithm checks the level of a signal at some epoch and makes a decision, as shown in Figure 2.

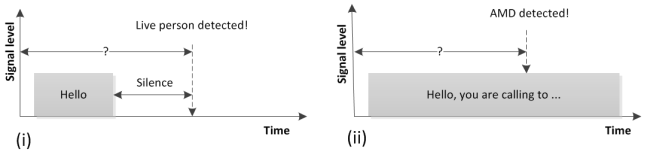

Figure 2: Heuristic algorithm

For example, the algorithm could be defined as follows: If a signal lasts over 3 seconds, it is an answering machine. Otherwise, it is a live person.

There may be many different definitions, but which definition is the best? To answer that question, we need to measure the accuracy of the algorithms.

Table 1. Data set format

Filename	Disposition
57d9814c6fcf43630829e237.wav	AM or Voicemail
57d88ade6fcf43630826ec26.wav	Product sold
57d88a026fcf43630826e565.wav	Not interested
57d88c176fcf43630826f3f6.wav	AM or Voicemail

3. USING CONVOLUTIONAL NEURAL NETWORKS

The data set for this project was collected from a real call center and is represented in the form of a call log. While running the outbound campaign with AMD enabled, the dialer stored short recordings for future technical analysis and detection algorithm improvements. Recording started when the media connection was established, and recording stopped when a detection algorithm made a decision. The length of each recording varies from several seconds up to 1 minute, depending on how long the call was in the ringing state. Recordings were saved as 8 kHz mono WAV files. For each recording, the system provided a set of meta-data. A brief description of the data set is given in the following table.

WAV files are used as a source of features for the model. A disposition gives rise to a binary label (if a disposition contains Answering then the binary label is 0; otherwise, it is 1). A connected offset (epoch when a call was answered) allows for a more meaningful segment without beep signals.

The final data set was cleaned by hand. It contains 3 classes (1 human, 2 - custom answering machine message, 3 some standard answering machine message) with 850 examples in each class. Dataset was augmented, from one original example, 20 examples were created sliding 2.5 sec window around (200 OK) connection event. Thus the dataset contains 51000 examples.

Two examples of recordings with different lengths are depicted in Figure 3.

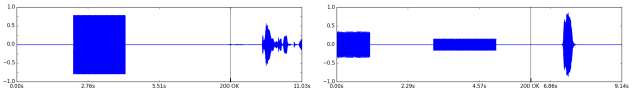

Figure 3: Examples of Call Signals

In Figure 3, the example on the left side represents a dialing beep signal with a fixed frequency in the form of a box followed by a connection event (`200 OK`) corresponding to answering the call. After connection, we hear a long uttering that is typical for an answering machine-type greeting, such as, Hello, you have reached a residence of X. After the tone, leave your message. The right example contains a short uttering of the greeting, Hello, which is from a living person.

The most meaningful part of these recordings occurs after the call answer, where figures show signals with various frequencies to be analyzed for live person detection.

3.1. Feature Extraction

An audio signal can be represented with the aid of different types of spectrograms. In this project, we use Mel frequency cepstral coefficients (MFCCs), which are successively used in speech recognition tasks. As a tool for calculating MFCC, we used the Librosa library[2].

MFCC of a signal can be found using the following steps:

1. Cut fixed segments of audio signal right after the call is answered.
2. Frame the signal into short frames.
3. Take the discrete Fourier transform for frames, and find the periodogram estimate of the power spectrum.
4. Compute the mel-spaced filter bank.
5. Take the logarithm of all filter bank energies.
6. Take the discrete cosine transform (DCT) of the log of all filter bank energies.
7. Keep the number of DCT coefficients, and discard the rest (in this project, 60 coefficients).

Figure 4 shows different representations of signals from Figure 3. In both figures, the x-axis is time and the y-axis is frequency for upper figure and N of MFCC for lower figure.

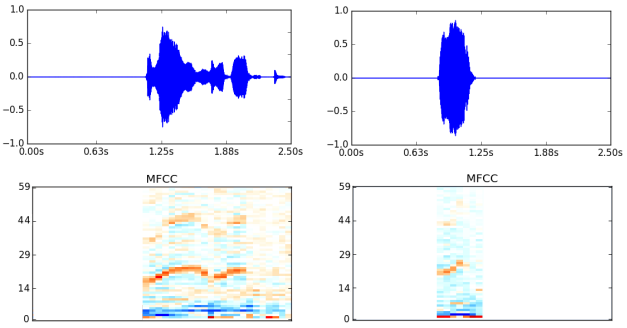

Figure 4: Representation of voice signals

3.2. Network Structure

Convolutional neural networks have three main types of layers: convolutional layer, pooling layer, and fully connected layer.

The convolutional layer is a set of filters that are applied to an input field. One filter with size $N \times M$ can be expressed as

$$f(x) = \sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} x$$

[2]Refer to https://github.com/librosa/librosa

It is applied to the input field with a specific stride by x and y (usually 1×1). The convolutional layer captures some two-dimensional structure on the input field.

The pooling layer is used to reduce the dimensionality of input data. It takes as input several values and produces one value. It is also applied with a predefined stride. Usually, the pooling layer has size 2×2, stride 2×2, and uses max function $f(x) = \max_{ij} x_{ij}$.

A fully connected layer is a regular neural network layer. Its general representation is $f(x) = Wx + b$, where W is a matrix of weights and b is a bias vector.

3.3. Implementation

3.3.1. CNN structure

A convolutional neural network that is used as a classifier of LPD is implemented using a Keras library with a TensorFlow library at the back end. Keras is a high-level neural network library that provides a convenient way to implement CNN. It can be easily switched between TensorFlow and Theano, enabling fast prototyping. It also has different types of layers and runs seamlessly on CPU and GPU.

The initial architecture of CNN was inspired by [19] and the Keras two-dimensional CNN example. We used four convolutional layers with softmax at the end. Dropout layers were added to prevent overfitting of the model.

The neural network has two convolutional layers. The first layer contains 50×6 (50 along frequency and 6 along time). The second layer has filters 3×3. Two last layers are fully connected layers. The neural network is finished by sigmoid activation function that produces a binary output. In intermediate layers ReLU, the activation function is used.

A grid search was performed to find values for the following hyperparameters:

- `conv_filters2` number of filters for the second convolutional layer
- `n_dense` number of neurons for the first fully connected layer
- `g_dropouts` dropout rate for first gaussian layer
- `dropouts` dropout rate for intermediate layers

During a series of manual experiments, batch size and the number of epochs were tuned.

3.3.2. Data Preprocessing

As mentioned previously, the data for this project contains a set of audio recordings. The process of feature extraction comprises the following steps:

1. Cut the fixed length of audio segments around the call answered. We used 2.5 second segments, 0.5 seconds before answer and 2 seconds after answer.

2. . Remove segments that are shorter than 1 second, and add zeros to segments shorter than 2.5 seconds.

3. Calculate MFCC with 60 coefficients, and remove the first coefficient because it is a measure of signal loudness and it is not very informative for speech processing tasks. The result is a matrix of 59×40.

4. Save arrays of features and labels in different files.

5. Split data into training and testing sets.

As a result, we got 38220 examples for training and 12780 examples for testing. Both sets were balanced.

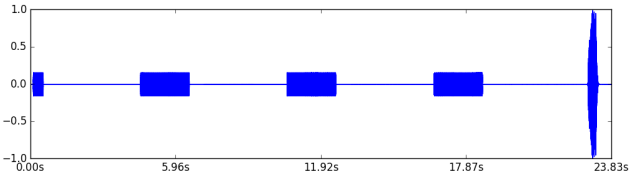

Figure 5: Audio signal before preprocessing

In Figure 6, the left side shows 2.5 second fragments around calls answered; the right side shows MFCC without the first coefficient.

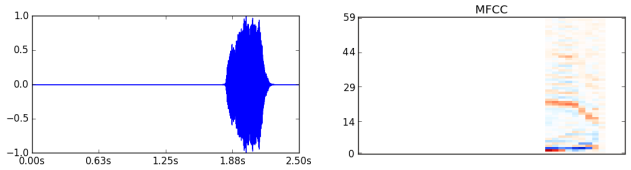

Figure 6: Audio signal after preprocessing

3.3.3. Training

The model was trained on the training data set and tested on the test data set. Some regularization techniques were applied during model training. The first input layer of CNN is batch normalization. It normalizes the input values at each batch. After each network layer, the Dropout layer is used. In this model, Dropout randomly sets its input to zero at each update during training time, which helps prevent overfitting. The Gaussian Noise layer is used as a kind of random data augmentation. It is located before the first convolutional layer. The Gaussian Noise layer randomly applies Gaussian noise to its input and it also prevents overfitting. Adam optimizer was used. A stochastic gradient descent with default parameters is used as an optimizer.

The model was trained on batches with size 256 during 20 epochs.

Model accuracy on training and test data sets during the training process is shown in Figure 7 on the left.

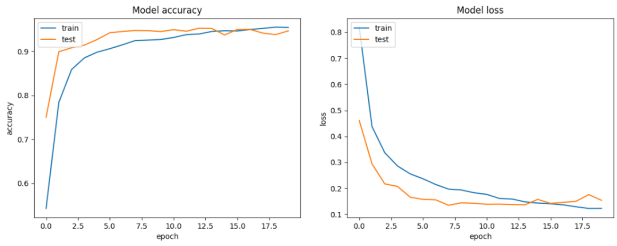

Figure 7: Model accuracy and loss

Model loss is shown in Figure 7 on the right. We can see that the model for the chosen parameters does not over-fit data. The final accuracy is 0.945 and loss is 0.186. That is our accuracy is equal to 94% and corresponds to the industry standard, see [5, 6, 7, 8].

Examples of model inputs are shown in Figure 8.

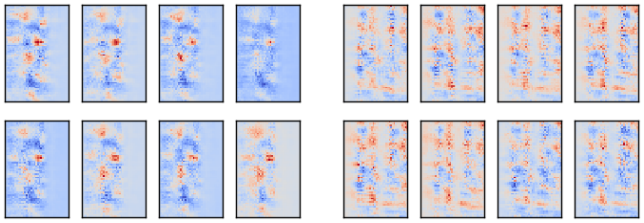

Figure 8: Examples of model inputs

In Figure 8, examples from the left side of the model provide maximum output, meaning it is some kind of living persons voice. The examples on the right side produced minimum output and show a generic representation of a non-living persons voice input. We can see that a live persons voice is more localized in time and a non-living persons voice has wider distribution.

4. CONCLUSION

In this paper, we described a method of detecting a living persons voice in outbound dialing campaigns of contact centers. The method uses a convolutional neural network as a model for classifying voice signals. As historical data, we used a call log containing call dispositions and call recordings from real contact centers. The method showed an accuracy equal to 94% that corresponds to the industry standard. This work is a first attempt to apply deep learning techniques to detecting answering machines and living persons. We plan to continue developing the method by experimenting with the depth of the architecture, improving data processing procedures, and considering other performance metrics involved, such as precision and recall.

5. REFERENCES

[1] Robert DeFrancesco, "Cloud disruption in the call center," *CMS WiRE*, 2014.

[2] Noah Gans, Ger Koole, and Avishai Mandelbaum, "Telephone call centers: Tutorial, review and research prospects," *Manufacturing and Service Operations Management*, vol. 5, no. 2, pp. 79–141, 2003.

[3] Alexander Szlam and Ken Thatcher, *Predictive dialing: Fundamentals*, Flatiron Publishing Inc., New York, 1996.

[4] Nikolay Korolev, Herbert Ristock, and Nikolay Anisimov, "Modeling and simulation of a pacing engine for proactive campaigns in contact center environment," in *Proceedings of the 2008 ACM Spring Simulation Multiconference (SpringSim'08). 2008 Business and Industry Symposium (BIS'08)*, Ottawa, Canada, April 2008, pp. 249–255, ACM Press.

[5] "Answering Machine Detection," Aspect Software, http://help.voxeo.com/go/ccxml/outbound.cpa.

[6] "Answering Machine Detection," Twilio, Inc., https://www.twilio.com/docs/api/voice/answering-machine-detection.

[7] TJ Thinakaran, "FAQ: How does Answering Machine Detection Work?," CallFire Inc., https://www.callfire.com/blog/2008/09/19/faq-how-does-answering-machine-detection-work, 2008.

[8] "Answering Machine Detection Algorithm," Cisco, https://supportforums.cisco.com/t5/collaboration-voice-and-video/answering-machine-detection-algorithm/ta-p/3117321, 2017.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, 2012.

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, 2013, pp. 6645–6649.

[13] Ian Vince McLoughlin, "Review: Line spectral pairs," *Signal processing*, vol. 88, no. 3, pp. 448–467, 2008.

[14] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014.

[15] Zhang Haomin, Ian McLoughlin, and Yan Song, "Robust sound event recognition using convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, Brisbane, Australia, April 19?24, 2015*, 2015.

[16] Pavel Golik, Zoltan Tuske, Ralf Schluter, and Hermann Ney, "Convolutional neural networks for acoustic modeling of raw time signal in lvcsr," in *INTERSPEECH*, 2015.

[17] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Dasg, "Very deep convolutional neural networks for raw waveforms," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, 5-9 March 2017*, 2017.

[18] Shuhui Qu, Juncheng Li, Wei Dai, and Samarjit Das, "Understanding audio pattern using convolutional neural network from raw waveforms," *CoRR*, vol. abs/1611.09524, 2016.

[19] Karol J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, MM '15, pp. 1015–1018, ACM.