INFERRING SOCIAL CONTEXTS FROM AUDIO RECORDINGS USING DEEP NEURAL NETWORKS

Meysam Asgari, Izhak Shafran, Alireza Bayestehtashk

Center for Spoken Language Understanding
Oregon Health & Science University

ABSTRACT
In this paper, we investigate the problem of detecting social contexts from the audio recordings of everyday life such as in life-logs. Unlike the standard corpora of telephone speech or broadcast news, these recordings have a wide variety of background noise. By nature, in such applications, it is difficult to collect and label all the representative noise for learning models in a fully supervised manner. The amount of labeled data that can be expected is relatively small compared to the available recordings. This lends itself naturally to unsupervised feature extraction using sparse auto-encoders, followed by supervised learning of a classifier for social contexts. We investigate different strategies for training these models and report results on a real-world application.

Index Terms— Multi-label classification, Deep neural networks, Harmonic model

1. INTRODUCTION
Low power devices and smartphone apps have made it possible to record clips of everyday life with relative ease for sharing on social media or for archiving. These recordings such as those on YouTube have a wide range of background noise and the task of transcribing the spoken utterances present in them is challenging. Social context is an alternative layer of information that can be inferred from the audio and used to annotate and index these clips. Furthermore, real-time inference of social context would be highly useful in providing personalized services on smartphones. For social scientists, psychologists and gerontologists, the ability to infer social context from audio life logs provides a convenient way to study social behaviors without perturbing the behavior itself, unlike previous methods of measurements that relied on sampling or recollections in journal entries. Through a number of studies, ranging from depression to adolescent behavior, Pennebaker and Mehl have already illustrated the value of inferring social contexts from audio life logs even when they were severely handicapped by the need to listen to audio and manually annotate them [1]. They demonstrated that social context and other information from the audio life logs can be used to quantify subjects’ social life (interaction and engagement), cognitive function, emotional conditions, or even health status [2]. In this study, our focus is on automatically inferring social contexts from life logs specifically those collected by Mehl and colleagues [3].

The audio life logs can be easily collected in large amounts, however, annotating them is relatively expensive. Supervised classifiers such as support vector machines do not have the ability of using unlabeled data. On the other hand, deep neural networks, specifically, auto-encoders can extract potentially useful features in an unsupervised manner. The layers of the network that extract these features can then be modified in a supervised manner to fine-tune the network for a classification task with limited amounts of labeled data. We investigate the use of such a framework to detect social contexts [4], [5], such as speakers’ location (e.g., in transit) or activity (e.g., watching TV or eating) in audio life logs. Moreover, the early layers that extract potentially useful features may be shared across related tasks such as the two classification tasks. In certain applications, such sharing have been shown to be beneficial and so we investigate multi-label learning on audio life logs. Most previous work on the general audio classification task have employed MFCCs. Experiments on speech recognition show that neural networks provide better performance with filter bank features. We investigate an alternative feature representation using harmonic model that captures the harmonic nature of natural sounds. We report experiments on real world samples of life logs with manual annotations.

2. CORPUS
Our corpus for this study consists of samples of snippets of audio recordings from everyday life of university students. The corpus was collected using the Electronically Activated Recorder (EAR), which records 30-seconds snippets, every 12 minutes. This sampling scheme was chosen to provide sufficient information about social lives of the students in a way that doesn’t allow full reconstruction of their private lives [1]. The corpus was collected from 96 student volunteers, who were asked to wear a lapel microphone connected to the EAR device for 4 consecutive days during their waking hours. The resulting 22,140 audio snippets or files were manually anno-
3. DEEP NEURAL NETWORKS

Deep neural networks have been demonstrated to outperform other machine learning techniques in a variety of tasks ranging from speech recognition to computer vision. In brief, a deep neural network (DNN) is a feed-forward network comprised of several layers of hidden units. These networks are trained using back-propagation [7]. The problems encountered in training large number of layers using back-propagation is better understood now. Improvement in training techniques and availability of large amounts of data and compute power have resurrected them.

3.1. Learning strategy

The DNN architecture for classifying the speaker’s location and activity is shown in Figure 1. It consists of several hidden units and two separate softmax units, in the output layer, to classify two independent classes of events (i.e., location and activity). Two steps of learning the network are described as follows: 1) Unsupervised learning of hidden layers, known as pre-training, performed in a greedy layer-wise fashion introduced by Hinton [8]. First, a sparse autoencoder is trained in the first layer in an unsupervised manner. Then, the output of this layer (feature activations in hidden layer) is fed to the higher layer as input features. Similarly, the second sparse autoencoder and eventually higher layers are trained. 2) Supervised fine-tuning performed by the back-propagation algorithm, in which estimated weights of softmax and hidden units are adjusted.

3.2. Sparse Autoencoder

The Autoencoder (AE) is a neural network with one layer of hidden units. It encodes the underlying structure of input features by learning an identity function that maps the input vector of \( \mathbf{x} \) into the output vector of \( \hat{\mathbf{x}} \), which is similar to the \( \mathbf{x} \). Suppose we have \( D \)-dimensional unlabeled speech features (like MFCCs) set \( \{x_1, x_2, x_3, \ldots, x_n\} \). The learning process begins with an encoding function \( f \) that maps input vectors to the hidden representation as shown in Equation 3.2.

\[
h = f(x) = s(W_1x + b_1)
\]

where \( s \) is a non-linear activation function such as logistic sigmoid, and \( b_1 \) is a bias vector. Then, a decoder function \( g \) maps the hidden representation to the output layer for reconstructing the vector \( x \).

\[
\hat{x} = g(h) = s(W_2x + b_2)
\]

The goal of learning is to estimate unknown parameters \( \theta = \{W_1, W_2, b_1, b_2\} \) such that the reconstruction error on the training set is minimized. We define the following objective function where the cost function, \( L \), is the traditional squared error as follow.

\[
J(\theta) = \sum_{i=1}^{n} L(x_i, g(f(x_i)))
\]

When the number of units in the hidden layer is large, the autoencoder needs sparsity constraints to provide useful representations. One effective constraint is to minimize the average activation of the hidden units [9].

\[
J(\theta) = \sum_{i=1}^{n} L(x_i, g(f(x_i))) + \beta \sum_{j=1}^{n_h} \text{KL}(p || \hat{p}_j)
\]

where \( \beta \) controls the weight of the sparsity term, \( \hat{p}_j \) is average activation of hidden unit \( j \) over whole training set, and \( p \) is a constant sparsity term close to zero (typically \( \rho = 0.05 \)) that defines the level of the sparsity. \( \text{KL}(p || \hat{p}_j) \) is the Kullback-Leibler (KL) divergence between a Bernoulli random variable with the mean of \( p \) and a Bernoulli random variable with the mean of \( \hat{p}_j \).

3.3. Softmax classifier

The classifier outputs are typically modeled using a softmax layer containing the same number of nodes as categories, in the form of 1-of-\( K \) coding scheme [10]. The conditional probability \( p(c_k | \mathbf{x}) \) of a softmax classifier is given by:

\[
p(c_k | \phi) = \frac{\exp(D_k^T \phi)}{\sum_{j=1}^{K} \exp(D_j^T \phi)}
\]
where $c_k$ is the class identifier, $D$ is the weight matrix of softmax classifier and $\phi$ is a feature vector (output of the last hidden layer). Given the feature vector, $\phi$, conditional probability for all classes, $c_1, \ldots, c_K$, is evaluated for predicting the target category employing an $\arg\max$ function as follow.

$$\hat{c} = \arg\max_{c_1, \ldots, c_K} p(c_k|\phi)$$

(6)

Unknown weights of the softmax layer is typically estimated by minimizing the cross-entropy error function given by:

$$L(D) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p(c_k|\phi_n)$$

(7)

where $N$ is the total number of training instances and $t_{nk}$ stands for a vector of binary label associated with the $n$-th data instance.

3.4. Multi-label classifier

In multi-label classification, contrary to the single-label classification, instances of data are associated with a set of class labels. Medical diagnosis, text documents, and movie genres are a few examples that often can be categorized into more than one category. Multi-label classification problem can be cast into a set of independent single-label classification problems [11], in which each classifier is independently trained over training instances of the class. For example in our classification task, one can independently learn two DNNs for separate classification of speaker’s current activity and location. However, class labels are not necessarily independent of each other and thus, correlated information is ignored in this approach. As an alternative approach, instead of employing two independent DNNs for multi-label classification, we propose to employ a single DNN with two independent softmax units in its output layer as shown in Figure 1. In this architecture, the network is learned to minimize the cross-entropy error function given by:

$$L(D_a, D_l) = -\sum_{n=1}^{N} \sum_{k=1}^{K_a} t_{nk} \log p(c_{k}^a|\phi_n) +$$

$$-\sum_{n=1}^{N} \sum_{k=1}^{K_l} t_{nk} \log p(c_{k}^l|\phi_n)$$

(8)

where $D_a$ and $D_l$ are the weight matrices, $K_a$ and $K_l$ are the number of categories, and $c^a$ and $c^l$ are class identifiers of activity and location softmax units, respectively.

3.5. Fine-tuning

After pre-training of hidden layers, weights of hidden layers, $\theta$, and softmax units, $D_a$ and $D_l$, are fine-tuned in a supervised fashion using back-propagation algorithm. The gradient of the cross-entropy error function is back-propagated and the the gradient with respect to the weights are computed at each layer, which are used to update the respective weights.

4. SPEECH FEATURES

In our experiment, we used a broad range of features to capture features associated with these categories. For our baseline system, we adopted the baseline features defined in Interspeech 2010 Paralinguistic Challenge [12] using openSMILE toolkit [13]. The features, comprised of 1582 components, can be broadly categorized into three groups: 1) loudness related features such as RMS energy and PCM loudness, 2) voicing related features like pitch frequency, jitter, and shimmer., and 3) articulatory related features such as mel-frequency cepstral coefficients and line spectral frequencies. The features computed at the frame-level were summarized into a global feature vector of fixed dimension for each recording using 21 standard statistical functions including min, max, mean, skewness, quartiles and percentile. We employ two sets of acoustic features for the comparison: 1) Mel frequency cepstral coefficients (MFCCs), which are widely used in automatic speech and speaker recognition systems, and 2) pitch-related features extracted from harmonic model of speech.

4.1. Speech Features from Harmonic Model

We extract 25 millisecond long frames using a Hanning window at a rate of 100 frames per second before computing the frame-level features using harmonic model (HM) described briefly at follow. Let $x = [x(t_1), x(t_2), \ldots, x(t_N)]^T$ denote the speech samples in a voiced frame, measured at times $t_1$, $t_2$, $\ldots$, $t_N$. The samples can be represented by the harmonic model with additive noise $n = [n(t_1), n(t_2), \ldots, n(t_N)]^T$ as follows:

$$s(t) = \sum_{h=1}^{H} a_h \cos(2\pi f_0 h t) + b_h \sin(2\pi f_0 h t)$$

$$x(t) = s(t) + n(t)$$

(9)

where $H$ denotes the number of harmonics and $2\pi f_0$ stands for the fundamental angular frequency. Assuming the noise distribution is constant during the frame and is given by $N(0, \sigma^2_n)$, estimation of the unknown parameters, $\{f_0, a_h, b_h, \sigma^2_n, H\}$, can be cast into a maximum likelihood (ML) framework [14]. However, ML estimation of the pitch period may lead to pitch halving and doubling errors. We addressed this problem in our previous work and improved the robustness of the pitch estimates by smoothing the likelihood function [15]. Given the constant number of harmonics obtained from the model order selection [15], $H = 7$, we estimate the coefficient of harmonics and introduce a 15-dimensional
feature vector, \([a_0, a_1, \ldots, a_7, b_1, \ldots, b_7]^T\). We then transform the feature vector to the log-domain followed by taking its absolute value. Moreover, given the estimate of unknown parameters, we reconstruct noise-free signal and compute other pitch-related features such as Harmonic-to-Noise Ratio (HNR), shimmer, and jitter. We refer the reader to our recent work [16] for more detail on extracting aforementioned features from harmonic model.

After extracting frame-level features, we summarize them into a global feature vector of fixed dimension for each 30-second long recordings. Features extracted from voiced regions tend to differ in nature compared to those from unvoiced regions. These differences were preserved and features were summarized in voiced and unvoiced regions separately. Each feature was summarized across all frames from the voiced (unvoiced) segments in terms of standard distribution statistics such as mean, median, variance, minimum and maximum. Ultimately, segment-level summary features of voiced and unvoiced regions were concatenated into a global vector of 182 features for each recording.

5. EXPERIMENTS

We evaluate the effectiveness of the proposed method on learning single-label and multi-label classification models. We chose a subset of our recordings which were annotated with in-apartment, in-transit, or in-restaurant from the Location class, as well as sleeping, eating, watching TV, studying, or working from the Activity class. This gave us an evaluation set containing of 1470 recordings with two-label annotations. Training and test set of recordings on this evaluation set were then defined by speaker independent subdivisions as shown in Table 1.

<table>
<thead>
<tr>
<th>Class</th>
<th>Train</th>
<th>Test</th>
<th>(\Sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in-apartment</td>
<td>384</td>
<td>126</td>
<td>510</td>
</tr>
<tr>
<td>in-transit</td>
<td>477</td>
<td>189</td>
<td>666</td>
</tr>
<tr>
<td></td>
<td>189</td>
<td>105</td>
<td>284</td>
</tr>
<tr>
<td>total</td>
<td>1050</td>
<td>420</td>
<td>1470</td>
</tr>
<tr>
<td>Activity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>computer</td>
<td>321</td>
<td>135</td>
<td>456</td>
</tr>
<tr>
<td>social</td>
<td>207</td>
<td>69</td>
<td>276</td>
</tr>
<tr>
<td>eat</td>
<td>333</td>
<td>117</td>
<td>450</td>
</tr>
<tr>
<td>work</td>
<td>189</td>
<td>99</td>
<td>288</td>
</tr>
<tr>
<td>total</td>
<td>1050</td>
<td>420</td>
<td>1470</td>
</tr>
</tbody>
</table>

Table 1. Test and training sets for location and activity.

5.1. Evaluation metrics

In order to evaluate the performance of proposed multi-label classifier, we adopt evaluation metrics that are frequently used in the literature. These metrics are different from conventional metrics adopt for single-label classification problems. To formulate the evaluation metrics, let \(\Gamma\) be our multi-label dataset containing \(N\) multi-label instances, \((x_i, y_i), i = 1, \ldots, N\). Also, \(|L|\) stands for the number of labels in the target vector \(y_i\), and \(h(x_i) = z_i\) denotes predicted labels for the instance \(x_i\). We assess the performance of our multi-label classification task using accuracy (ACU) and Hamming-Loss (HL) described at follow.

**Accuracy** For a multi-label data instance, accuracy (ACU) is the portion of labels that are correctly identified in all classes divided by the total number of labels. The main drawback of this measure is that it ignores partially correct labels. Higher the accuracy corresponds to the better performance. For the entire dataset, accuracy is computed as follow.

\[
\text{Accuracy}(z, y) = \frac{1}{N} \sum_{i=1}^{N} I(z_i = y_i)
\]

where, \(I\) is the indicator function.

**Hamming-Loss (HL)** HL is a widely used criterion in multi-label classification that counts the number of instances that are incorrectly predicted. Lower the value of HL corresponds to the better performance and defined as follow.

\[
HL(z, y) = \frac{1}{N} \sum_{i=1}^{N} \left| z_i \oplus y_i \right| / |L|
\]

where \(\oplus\) stands for symmetric difference (XOR operation) between two \(h(x_i)\) and \(y_i\) sets.

5.2. Multi-label classification

For the evaluation of multi-label classification task, we learn a DNN with two hidden layers, each containing of 1,024 sigmoid hidden units. We then stack two independent softmax units representing Location and Activity classes on top of the DNN as depicted in Figure 1. First, we pool all available data in the corpus for pre-training of weights of hidden layers in the DNN using the sparse autoencoder. Note that this step of training is conducted in an unsupervised fashion on unlabelled data. Next, we feed training data into the pre-trained network and extract activation features from the last hidden layer. Activation features are then used for independently training of two softmax units with the number of units equal to 4 and 3 corresponding to number of categories in activity and location classes, respectively. In order to estimate unknown weights of softmax units, we define a cross-entropy error function described in Equation 7 and minimize it in a supervised manner. Estimated weights of softmax units are then stacked with pre-trained weights of the DNN to construct a multi-label classifier. The last step of training is fine-tuning, in which weights of hidden layers and softmax units are jointly adjusted in order to minimize the cross-entropy error function given by Equation 8. The length of feature vector...
input into the first layer of the DNN is 182, 1582, and 39 for HM, MFCCs, and openSMILE (OS) feature sets, respectively. The In order to highlight the effect of fine-tuning, we separately evaluated our multi-label classifier in two scenarios of before and after step of fine-tuning, and independently reported their performances in the Table 2. The table shows the accuracy and hamming-loss of the DNN-based classifier trained on three types of speech features on our multi-label evaluation set. As it is seen in the results, features from harmonic model perform better than MFCCs and openSMILE features, but the improvement may not be statistically significant in this evaluation set. Also, it is seen that that fine-tuning of the model parameters significantly improve the classification performance in terms of all evaluation metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Speech Features</th>
<th>Fine-tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MFCC</td>
<td>OS</td>
</tr>
<tr>
<td>Accuracy</td>
<td>68.4</td>
<td>70.3</td>
</tr>
<tr>
<td>Hamming-Loss</td>
<td>14.8</td>
<td>11.8</td>
</tr>
</tbody>
</table>

In order to gauge the relative importance of number of hidden units, we repeated this experiment with models learned on HM features with varying number of hidden units and reported their performance in the Table 3. Note that we retained the number of hidden units equal in each hidden layer. The results show that increasing the number of hidden units improve the performance by reducing the Hamming-Loss, though accuracy does not change. Also, adding more than 1,024 hidden units does not add to the performance.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Number of hidden units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128</td>
</tr>
<tr>
<td>Accuracy</td>
<td>801.</td>
</tr>
<tr>
<td>Hamming-Loss</td>
<td>10.8</td>
</tr>
</tbody>
</table>

5.3. Single-label classification

In this experiment, we conduct a single-label classification problem for independently predicting of categories in both Activity and Location classes. For the comparison, we employ several approaches and report their performances in terms of classification accuracy, the percentage of correctly identified labels. Two strategies for learning the DNN classifier is approached. First, we independently learn two DNN classifiers using data instances of each class. Note that these DNNs have a single softmax unit representing the class categories. Second, we employ the exact DNN trained in multi-label classification task and independently test on data instances of each classes. In this scenario, parameters of the DNN is retained unchanged from the multi-label training. For the completeness, we included a support vector classifier (SVM) using radial basis function (RBF) and linear kernels implemented in scikit-learn toolkit [17]. Parameters of the optimal SVM classifiers were determined on the training set separately for each fold, via grid search and cross-validation. Table 4 reports the performance of different classifiers measured in terms of classification accuracy for Activity and Location classes, respectively. In the table, DNN-SSU and DNN-DSO denote the DNN classifiers with single softmax unit (SSU) and double softmax units (DSU) in the output layer, respectively. Models were trained on three sets of speech features. From the results, it is clear that both versions of the DNN classifier outperform the SVM classifiers. The one exception is the openSMILE features, which performed better with SVMs for location than DNNs, though well-below the performance of our HM features. DNN-DSU slightly improves the performance compared to the DNN-SSU. This might be due to the fact that learning with multi-label data captures correlated information and that improves the performance in single-label classification problem.

<table>
<thead>
<tr>
<th>Class</th>
<th>Medels</th>
<th>Speech Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Chance</td>
<td>43.4</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>71.0</td>
</tr>
<tr>
<td></td>
<td>DNN-SSU</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td>DNN-DSU</td>
<td>79.6</td>
</tr>
<tr>
<td>Activity</td>
<td>Chance</td>
<td>31.4</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>54.2</td>
</tr>
<tr>
<td></td>
<td>DNN-SSU</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td>DNN-DSU</td>
<td>81.5</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

In this work, we find that DNNs can be employed effectively to infer social contexts from audio snippets of everyday life, achieving classification accuracy as high as about 88% and 87% for speakers’ location and activity. We also find that the features extracted using harmonic models are significantly better than MFCC and OpenSmile features in these tasks.

7. ACKNOWLEDGMENT

We thank Mathias Mehl for making the EAR corpus available for this project. This research was supported by Google, Intel and IBM awards as well as NSF awards IIS 0964102 and 1027834, and NIH award K25 AG033723. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not reflect the views of the funding agencies.
8. REFERENCES


