Time-series Clustering of Spoken Word Signals

Yushi Hu, Yiyang Ou

hys98@uchicago.edu, yiyangou@uchicago.edu

Abstract

In this project, we focus on spoken words clustering. Our goal is to cluster different speech samples of the same word to the same cluster and different words to different clusters. We compare the performances of two methods: Dynamic Time Warping (DTW) approach [1], and embedding approach using deep neural acoustic word embedding model proposed by Settle *et al.* [2]. Then we augment the performance of DTW by representation learning techniques. The experiment results show that the embedding approach produces overall better results. However, the unsupervised DTW combined with suitable dimensionality reduction method can perform almost as good as the supervised neural embedding approach.

Index Terms: time-series clustering, neural acoustic embedding model, dynamic time warping, representation learning, MFCC

1. Introduction

Clustering acoustic signals (e.g. in MFCC format) is a difficult task, because signals are multidimensional, have different lengths, and involve warping/shifting distortions. The traditional unsupervised approach is to use DTW, a distance measure that accounts for distortions and varying lengths. Yet, DTW has many drawbacks and is computationally expensive. Another approach is constructing fixed vector embeddings of acoustic signals, with which we only need to apply regular vector distance measure (e.g. cosine distance) and clustering techniques. With the success of deep learning, many recent works have produced promising results by training the acoustic embedding model with supervised deep neural networks. In this project, we wish to explore a recently proposed model and compare it with the DTW approach.

The embedding model we're interested in is proposed by Settle *et al.*[2] In the paper, the model produces significantly better results than DTW in words discrimination tasks. The first step of our project is to reproduce Settle's model and run it on words clustering tasks. Because the performance of the neural network model is contingent upon training set, we then explore the influence of training set on the embedding model, i.e. containing/not containing words in test set. Concurrently, we set up a DTW clustering workflow as a baseline and investigate if we can bridge the performance gap by preprocessing the data with representation learning techniques. All clustering performances are evaluated with NMI and purity.

2. Methods

In this section, we first describe the neural acoustic word embedding model, and then representation learning + DTW method. We then discuss the clustering algorithms we use after the above steps.

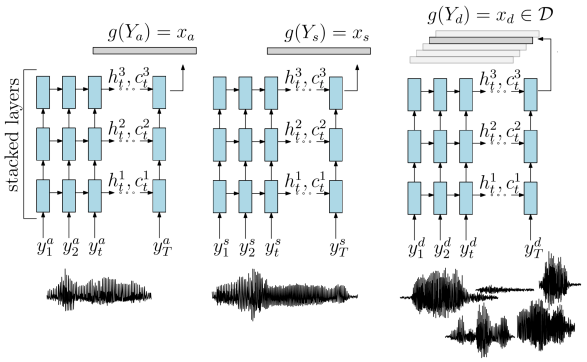

Figure 1: *Triplet Siamese training setup from Settle* et al. *[2] (unidirectional LSTM depicted for simplicity)*

2.1. Neural acoustic word embeddings (NAWE)

An acoustic word embedding maps a spoken word signal, i.e. a variable length segment $Y = y_1, y_1, \ldots, y_T$ of framed acoustic features to a fixed length embedding vector $x \in \mathbb{R}^d$. To get better performance in the clustering task, the embedding function should map signals of the same word to vectors that are close to each other, and the signals of different words to vectors that are far apart.

We used the deep bidirectional LSTM model from Settle *et al.* [2] for the embedding function. The model uses the concatenation of the final hidden vectors as the vector representation of the signal, i.e. $x = g(Y) = [\overrightarrow{h_T}:\overleftarrow{h_1}]$, where $\overrightarrow{h_T}$ and $\overleftarrow{h_1}$ are the final hidden vectors on top layer from forward and backward LSTMs, respectively.

The network is trained by Siamese weight-sharing scheme [4] with contrastive triplet loss [5,6]. It is illustrated by Fig.1. The contrastive triplet loss $L(Y_a, Y_s)$ between two signal input Y_a, Y_s with same word label is defined as

$$max\left\{0, m + d_{cos}(x_a, x_s) - \max_{x_d \in D} d_{cos}(x_a, x_d)\right\} \quad (1)$$

where $d_{cos}(x_a, x_s) = (1 - cos(x_a, x_s))$ is the cosine distance between embedding vectors. D is the set of k embedding vectors with different word labels. m is the margin in the hinge loss. The loss is forcing the cosine distances between the embedding vectors of signal with the same label to be smaller than the distances between one signal and another signal with different word label minus m. So after training, the embedding function can map signals with the same word label to vectors that are nearby, and map the signals with different word labels to vectors that are far apart. This makes it easy for clustering algorithms to separate signals into correct clusters.

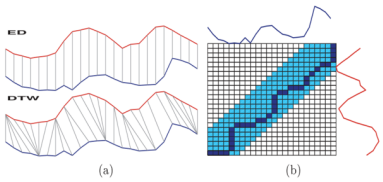

Figure 2: *Similarity computation: (a) alignment under ED (top) and DTW (bottom), (b) Warping path in darker blue and a band in lighter blue[7]*

2.2. DTW clustering + Representation Learning

2.2.1. DTW clustering

Given two one-dimensional signals $\vec{x} = (x_1, x_2, ..., x_n)$ and $\vec{y} = (y_1, y_2, ..., y_n)$. Note they can be of different length for DTW, but for simplicity of explanation and figure illustration, we just let them have same length. the simple Euclidean Distance (ED) is as follows:

$$ED(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

However, as shown in upper Fig.2 (a) [7], ED doesn't account for distortions like contractions and dilations. DTW solves this issue by offering a local nonlinear alignment of signals as shown in lower Fig.2 (a). To achieve this, an $n \times n$ distance matrix M is constructed with Euclidean Distance between x_i, y_j. A warping path W = $\{w_1, w_2, .., w_k\}$ is just a set of matrix elements that form a continuous path from lower left point to upper right point as shown in Fig.2 (b) Then DTW minimizes the sum of loss on the path:

$$DTW(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{k} w_i}$$

This path can be computed with dynamic programming by evaluating the following recurrence problem:

$$\gamma(i,j) = ED(i,j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

Then, $DTW(\vec{x}, \vec{y}) = \sqrt{\gamma(n,n)}$. It is common practice to constrain the warping path within a subset of cells, called *band*, on matrix M, as shown by lighter blue areas in Fig.2 (b). [7] Finally, to transfer DTW from this one-dimensional example to speech signals with x_i, y_j as MFCC feature vectors, we just need a vector distance measure for x_i, y_j and use these distances to construct W, and then compute DTW as the minimum path loss.

With DTW as distance measure, we then compute a pairwise distance matrix between speech signals, and feed this matrix to clustering algorithms which would be discussed later.

2.2.2. DTW with representation learning

Representation learning techniques are often used as a preprocessing step to extract low dimensional patterns and reduce noises of original data. The idea is the same in our attempt to augment DTW clustering. But we need a small trick before applying them to MFCC data. We concatenate n MFCC vectors of adjacent time frames to form a new MFCC feature vector. So, if $n = 5$ and original MFCC is of 39 dimensions, the concatenated MFCC feature vectors will have $5 \times 39 = 195$ dimensions. Note we need to add zeros as paddings for starting and ending time frames. This trick makes sense, because first, original MFCC vector is usually of small dimensions, so directly running dimension reduction technique won't be very effective, and second, speech signals often consist of phone units, which are small consecutive time frames. By concatenation, we also account for this structure in speech signals.

After this, we use different representation learning techniques, such as UMAP, PCA, etc to first transform each frame to lower dimensions, recombine these frames to speech signals, and finally run DTW and clustering algorithms.

2.3. Clustering algorithms

2.3.1. Clustering neural acoustic word embeddings

The trained neural network embedding will map speech signals $Y_1, Y_2, \ldots, Y_n$ to fixed-dimension vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$. Then we try multiple dimensionality reduction approaches on x_i, including PCA, KPCA with RBF kernel, Isomap, UMAP [3] and Variational Auto Encoder [9]. The reduced dimensionality is set to 20, 40, or 60. Finally we cluster the vectors $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n$ after dimensionality reduction with various clustering algorithms, including k-means; agglomerative clustering with single, complete, average, weighted, centroid, median, and Ward's method; spectral clustering with 5,10, and 50 neighbors [10]. All distance are measured by cosine distance.

2.3.2. Clustering distance matrix of DTW

After representation learning and dynamic time warping, we get the distance matrix D with $D_{ij} = DTW(Y_i, Y_j)$, i.e. the distance between ith and jth speech signal, meansured by DTW. Then we try agglomerative clustering with single, complete, average, weighted, centroid, median, and Ward's method and spectral clustering with distance matrix D as input.

3. Experiments

3.1. Dataset + Evaluation Metrics

In all the experiments, we use data from the Switchboard corpus of (primarily American) English conversational telephone speech. [11] The acoustic features are 39-dimensional MFCC+Δ+$\Delta\Delta$s. The data contains 31.9k signal-word pairs of 6204 words. Each speech segment's length ranges from 40 - 200. We make a small data set from these MFCC data (by choosing the top 20 words with the most number of spoken signals). The small dataset consists of 20 words, with 3645 spoken words signals. We cluster these 3645 signals to 20 clusters and measure the purity and NMI of each method. The original MFCC data are cut into three data sets, each consists of approximately 10k signals, to be used as training, development, and test set of the neural acoustic embeddings. Further adaptions of the datasets are made in the experiments and will be described later. Notice that all purity and NMI reported for each method are the highest values chosen among the results of the various clustering algorithms described in the previous section.

3.2. Clustering of neural acoustic word embeddings

In our implementation, the model is a 3-layer bidirectional LSTM with 256 dimension hidden vector in each direction. The

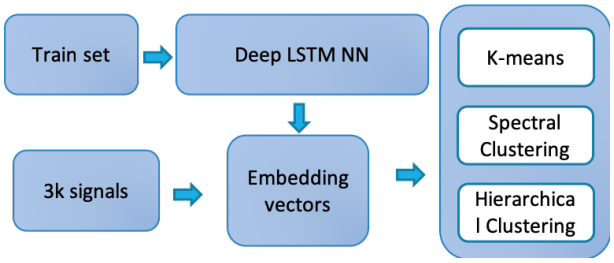

Figure 3: *NAWE experiment set up*

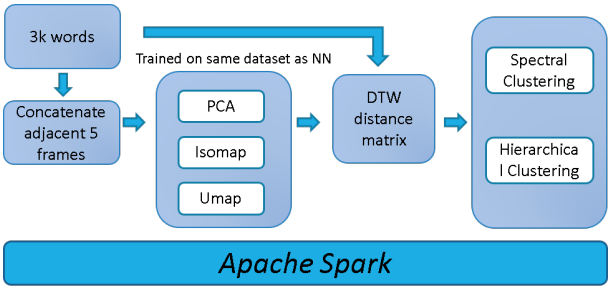

Figure 4: *DTW experiment set up*

embedding vector have 512 dimensions. The drop out rate is 0.3 between LSTM layers. In the contrastive loss, the margin is set as $m = 0.5$. The model is trained by Adam optimizer[8] with batch size 32 and learning rate 0.001. All hyper-parameters are the same as the model described in Settle *et al.*[2] . In all experiments below, we trained the network for 100 epochs.

The full experiment set up is shown in Fig. 3. Note that two neural networks are trained for neural acoustic embeddings:

The first neural acoustic word embedding model is trained with the training and development sets discussed above. Notice that these datasets contain signals of the words that are clustered and other signals. This embedding model is supervised and expected to achieve almost perfect performance.

The second neural acoustic word embedding model is trained with different training and development sets. The signals with word labels among 20 clustering words are removed from the train/dev set, and added to the test set. This embedding model is supervised, but expected to have worse performance than the first model, since it has not been trained by the signals of clustering words.

3.3. Results of embedding method and analysis

The NMI and Purity of clustering of these embeddings are shown in table 1.

Table 1: Clustering of Signal Embeddings with training set containing clustering words. Model 1's train/dev sets contain clustering words. Model 2's are not.

Model	***Model1***		***Model2***	
	NMI	Purity	NMI	Purity
Raw	0.952	0.970	0.886	0.900
PCA	0.952	0.970	0.883	0.904
KPCA-rbf	0.883	0.904	0.883	0.904
Isomap	0.954	0.973	0.881	0.897
UMAP	0.959	0.975	0.891	0.904
VAE	**0.962**	**0.978**	**0.889**	**0.915**

3.4. Clustering with DTW

As shown in Fig. 4, there are two experiment settings for DTW clustering: DTW on raw signals and DTW with representation learning techniques. For both settings, we would compute a pair-wise distance matrix on 3645 signals with DTW, and then feed this distance to clustering algorithms discussed before. For time purpose, we used FastDTW that uses a multilevel approach to approximate DTW within linear time and achieves near-optimal alignments. [12] Still, the computation of pairwise DTW distance matrix is very expensive, so we relied on Apache Spark, a distributed computing framework, to parallelize the computation. [13] We ran all DTW experiments on two 40-core machines with Apache Spark 2.4.0.

First, for raw signals, there's one hyper-parameter we need to consider: distance measure between MFCC vectors. We compared ED, correlation and cosine distance, and chose ED in the end, because it ran much faster and the other two didn't have significant improvements over ED. We then directly feed 3645 words to Spark to compute DTW distance matrix. We then locally run Spectral Clustering and Agglomerative Clustering with precomputed distance matrix.

Second, for DTW + representation learning. We trained representation learning techniques on the training and development set containing test set word. First, we concatenate adjacent 5 time frames. For example, suppose a signal $\vec{x} = \{x_1, ..., x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, ..., x_n\}$. Then x_i will be replaced with $[x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}]$ and we pad zeros for the starting and ending time frames. We flatten all the signals, considering each time frame as an individual data point. Then we experimented three representation learning techniques, UMAP, PCA, Isomap, and reduced frames to 20, 40 ,60 dimensions (Because we don't have enough time, we only tried 40 dimension for Isomap). For other hyper parameters, we tuned the number of neighbors to 30 for UMAP and Isomap instead of lower default values, because there is much noise in the data. In the end, we recombine time frames into speech signals and the rest is the same as the first setting.

Finally, we apply two clustering algorithms discussed above. Specifically, we tune the linkage method for agglomerative clustering and find that Ward's outperforms the others significantly: Ward's gives around 0.7 purity, while the others are below 0.1.

3.5. Results of DTW Clustering and analysis

The NMI/purity results are reported in Table 2, and clustering result from DTW+UMAP+Agglomerative is visualized with t-SNE. [14] Note that results of agglomerative all used Ward's method. We find interesting observations from this. 1) Agglomerative clustering is significantly better than spectral methods. But for agglomerative clustering, only Ward's produces reasonable performance. Combined with t-SNE visualization, one possible explanation is that DTW has made words in the same cluster close to each other, forming small blobs. But some blobs corresponding to different clusters aren't well separated, such as the "something" and "exactly" clusters in the graph. Since spectral clustering computes the k-nearest neighborhood,

Table 2: Clustering with DTW. All results except for the last row is run with agglomerative clustering + Ward's. The last one is with spectral clustering

Representation	*DTW + Agglomerative*	
	NMI	Purity
Raw 39-dim	0.743	0.656
PCA 20-dim	0.746	0.691
PCA 40-dim	0.775	0.721
PCA 60-dim	0.751	0.688
UMAP 20-dim	0.822	0.834
UMAP 40-dim	**0.832**	**0.850**
UMAP 50-dim	0.811	0.827
Isomap 40-dim	0.627	0.547
PCA 40-dim (Spectral)	0.132	0.212

it is negatively affected by the lack of separation between clusters, while Ward's nicely captures the blob structures. 2) In general, representation learning improves performance, especially UMAP, which increases purity from 0.6 to 0.8. However, Isomap reduces the purity compared with raw data. This is probably due to insufficient tuning, since UMAP and Isomap falls in the same class of k-neighbor based graph learning algorithms. [3] This type of method favors the preservation of local distances over global distance, while PCA preserves overall distance structure. The success of UMAP aligns with the intuition that local structures matter more than global distances for speech signals, but more works and experiments are needed to more comprehensively explain why some representation learning methods work better than others. 3) 40 dimensions seem like a sweet spot for the reduced dimensions. Higher dimension (60) probably introduces more noises for DTW which leads to worse results.

4. Conclusions and Future Work

As seen in Fig. 6, the t-SNE visualization[14] of the results from above experiments together show that neural embedding model outperforms DTW under all settings, even when we pretrained representation learning on the same training + development set for DTW or we removed test set word from training + development set for the neural network. This proved to us the supervised learning indeed has the potential to produce a generalizable model that well describes the distance between words. Yet, we also see that representation learning techniques can significantly improve the performance of DTW clustering, and specifically, Uniform Manifold Approximation and Projection (UMAP [8]) can make it perform almost as good as the embedding method.

For future work, 1) We'd like to compare these two methods on other training environments. For example, we can train representation learning and neural network model on the 3k set and see how they perform on this small training set. We can also make the training set more challenging to neural network by increasing the number of unseen words or varying words distribution in training and test set. 2) We'd like to perform similar comparisons of the two approaches on other speech tasks. For example, we can cluster speech signals containing several words, i.e. with unknown start and end point for each word. 3) We would like to try more representation learning techniques and figure out why certain methods work better than the others.

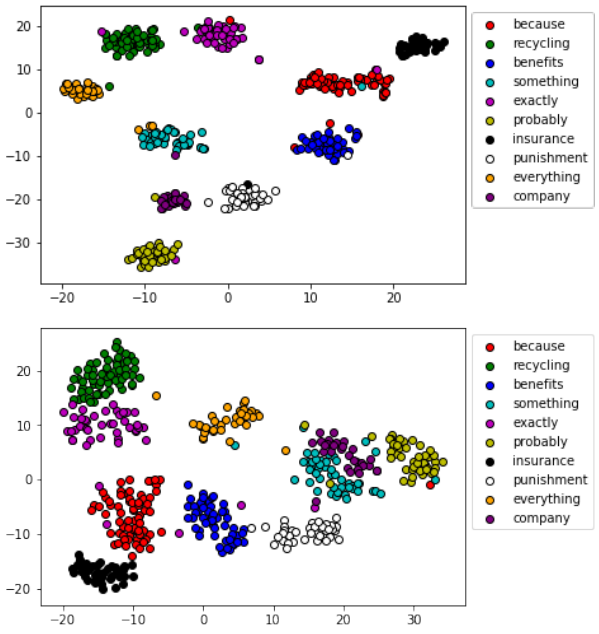

Figure 5: *Comparision of t-SNE visualization. Upper plot: Neural Embeddings. Lower plot: DTW+UMAP*

For this task, we will need more domain-specific knowledge, such as structures of phones, etc.

5. Acknowledgements

This is a 2-person project. Yushi is responsible for the dataset, the neural embedding experiment, UMAP training, and clustering algorithms; Yiyang is responsible for the DTW experiment and other representation learning techniques.

Thanks to Karen for providing the idea of clustering neural acoustic embeddings and using representation learning to augment DTW performance. Thanks to Shane for providing us the dataset and the model of neural acoustic embeddings.

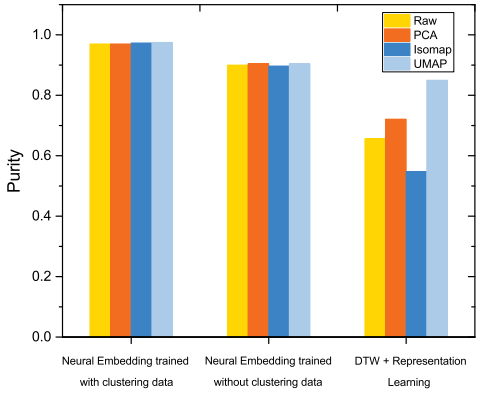

Figure 6: *Comparision of Clustering Purity*

6. References

[1] D. Berndt, J. Clifford, "Using dynamic time warping to find patterns in time series," KDD workshop, 1994 - aaai.org

[2] S. Settle, k. Levin, H. Kamper, K. Livescu "Query-by-Example Search with Discriminative Neura lAcoustic Word Embeddings," INTERSPEECH 2017

[3] McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints, February 2018.

[4] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Sackinger, and R. Shah, "Signature verification using a "Siamese' time delay neural network," Int. J. Pattern Rec. vol. 7, no.4, pp. 669-688, 1993.

[5] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in Proc CVPR, 2015.

[6] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," Trans. ACL, vol. 2, pp. 207218, 2014.

[7] J. Paparrizos and L. Gravano, "k-Shape: Efficient and Accurate Clustering of Time Series," 2015 ACM SIGMOD International Conference on Management of Data

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, 2014.

[9] Kingma, Diederik P and Welling, Max. Auto-Encoding Variational Bayes. The 2nd International Conference on Learning Representations (ICLR), 2013.

[10] F. Pedregosa et al. , "Scikit-learn: Machine Learning in Python ," JMLR, 12(Oct):2825-2830, 2011.

[11] John J Godfrey, Edward C Holliman, and Jane McDaniel, Switchboard: Telephone speech corpus for research and development, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1992, vol. 1, pp. 517520.

[12] S. Salvador and P. Chan, FastDTW: Toward accurate dynamic time warping in linear time and space, in Proc. KDD Workshop on Mining Temporal and Sequential Data, 2004.

[13] Zaharia, Matei, et al. "Spark: cluster computing with working sets. Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 2010.

[14] van der Maaten, L. and Hinton, G. E. Visualizing data using t-SNE. J. Mach. Learn.Research 9, 25792605 (2008).