SPEECH SEGMENT CLUSTERING FOR REAL-TIME EXEMPLAR-BASED SPEECH ENHANCEMENT

David Nesbitt, Danny Crookes, Ji Ming

School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast, BT7 1NN, UK

ABSTRACT

Exemplar-based (or Corpus-based) speech enhancement algorithms have great potential but are typically slow due to needing to search through the entire corpus. The properties of speech can be exploited to improve these algorithms. Firstly, a corpus can be clustered by a phonetic ordering into a search tree which can be used to find a best matching segment. This dramatically reduces the search space, reducing the time complexity of searching a corpus of n segments from $O(n)$ to $O(log(n))$. Secondly, clustering can be used to give a lossy compression of a speech corpus by replacing original segments with codewords. These techniques are shown in comparison with sequential search and non-compressed corpora using a simple speech enhancement algorithm. A combination of these techniques for a corpus of a quarter of WSJ0 results in a speedup of approximately 3000x.

Index Terms— speech enhancement, exemplar-based, real-time, embedded, clustering

1. INTRODUCTION

Speech enhancement algorithms that do not rely on estimation of noise have great potential in real-world applications due to the diversity and unpredictability of noise. One issue is that in a short time frame noise can be very similar to some speech sounds, which makes real-world speech enhancement particularly challenging for more traditional frame-based solutions which tend to be limited in quality unless they model properties of both the speech and the noise. Often these methods face challenges with quickly varying noise because of a reliance on a stationary estimation of noise.

Examples of frame-based solutions include spectral subtraction [1], Wiener filtering [2][3], MMSE estimators [4], and hidden Markov model based approaches [5].

On the other hand, segment-based algorithms, which take multiple frames at a time, have the advantage of greater discrimination between speech and noise. Healy et al. [6] aims to estimate an ideal mask to filter the speech from a noisy signal and does so by using a deep neural network to estimate the mask. This work is promising and, while still speaker-dependent, has moved to generalize to unseen noise samples and noise types [7] [8].

1.1. Exemplar-based matching

The exemplar-based approach uses a similarity measure to select examples of speech features from a learned corpus [9] and is typically segment-based. A variety of methods exist, using different similarity measures, including non-negative matrix factorization (NMF) [10] [11], maximum posteriori probability [12] or likelihood [13], and maximum correlation as used in DNNs [14] and sentence-based matching [15].

Exemplar-based speech enhancement algorithms can, however, be too slow to use in real-time systems, due to the requirement to search an entire corpus and compare each corpus segment with the test segment. Additionally, to ensure that the corpus is representative of all possible speech sounds belonging to the language for which it is trained, often the corpus must be composed of a large amount of speech. This does, however, result in redundancy within the corpus, with many of the segments being similar to each other.

In this paper we present a system to resolve both the lack of speed in exemplar-based methods and the large working memory requirements of representative corpora. Resolving these issues would make these algorithms feasible on a wide range of hardware, including embedded platforms. A previous system aiming for fast searching came close to, but not quite, real-time [16] - this approach is, however, unlikely to scale to lower-powered hardware. Another system for HMM exemplar-based matching used a form of clustering for hierarchical searching which resulted in a speedup of approximately 8x and reduced the size of the exemplars in the corpus [17].

This system is the subject of ongoing research and is relatively simple; we present it here to demonstrate our concept. This method could be used in more sophisticated exemplar-based speech enhancement algorithms such as wide matching [15]. Wide matching aims, for a test utterance, to find a chain of fixed length matching segments from the corpus that maximally utilizes the short and long term lingua-acoustic context of the speech. This approach was found to yield good results without any estimation of noise.

Another application of these concepts would be to facilitate the incorporation of an exemplar-based algorithm into a learning ensemble. Segmental exemplar-based algorithms are significantly distinct from other approaches, so the inclusion of the algorithm in an ensemble-based deep-learning system ought to increase the output quality of the system[18].

1.2. Distance measure

The distance measure (or conversely, similarity measure) that is used by the system to determine the distance between a test segment and a corpus segment should be acoustically meaningful, that is, the more similar two segments sound, the lower their distance should be. Ideally the distance measure should be invariant to the magnitude of the signal; however, the speed of the signal is now considered to be relatively insignificant, as recently discovered in DNN studies [19].

Common distance measures, City Block distance and Euclidean distance are not computationally complex but are variant to magnitude. Pearson's distance is invariant to volume and so is a possibility. Zero-mean Normalised Correlation Coefficient (ZNCC) is invariant to volume and it has been demonstrated experimentally and mathematically that with this measure of similarity, as the length of a segment increases, the effect of independent additive noise on the correlation between the underlying speech and a corpus segment (and thus the correlation between the noisy segment and a corpus segment) tends to zero [15]. ZNCC is expressed as:

$$R(\mathbf{x}_{t\pm L}, \mathbf{s}_{\tau\pm L}) = \frac{\sum_{l=-L}^{L}[\mathbf{x}_{t+l} - \mu(\mathbf{x}_{t\pm L})]^{\mathrm{T}}[\mathbf{s}_{\tau+l} - \mu(\mathbf{s}_{\tau\pm L})]}{|\tilde{\mathbf{x}}_{t\pm L}||\tilde{\mathbf{s}}_{\tau\pm L}|} \quad (1)$$

where $\mathbf{x}_{t\pm L}$ represents a noisy fixed-length segment between frame $t-L$ and $t+L$ and $\mathbf{s}_{\tau\pm L}$ represents a clean speech segment between frame $\tau-L$ and $\tau+L$, both segments' frames being vectors in the spectrum domain. $\mu(\mathbf{x}_{t\pm L})$ represents the mean frame vector for the noisy segment centering on t (with $\mu(\mathbf{s}_{\tau\pm L})$ expressing the same for clean speech segment $\mathbf{s}_{\tau\pm L}$) and $|\tilde{\mathbf{x}}_{t\pm L}|$ represents the zero-mean Euclidean norm for the noisy segment centering on t (with $|\tilde{\mathbf{s}}_{\tau\pm L}|$ expressing the same for clean speech segment $\mathbf{s}_{\tau\pm L}$). On this basis ZNCC was chosen as the distance measure for the system to demonstrate the concept in this paper, but the concept can be applied to other measures. From this we estimate the underlying speech segment in the noisy signal ($\hat{\mathbf{s}}_{t\pm L}$):

$$\hat{\mathbf{s}}_{t\pm L} = \arg\max_{\mathbf{s}_{\tau\pm L}} R(\mathbf{x}_{t\pm L}, \mathbf{s}_{\tau\pm L}) \quad (2)$$

1.3. Challenges addressed

We present the following novel solutions to the issues presented so far. Both solutions can be carried out at the corpus creation stage, prior to using the algorithm to enhance speech.

1.3.1. Real-time searching of corpora

Search-trees vastly reduce the search space of a search function; however, they can only be used with orderable data sets. Segments of speech cannot naturally be ordered in any useful manner. We seek to demonstrate that an approximate order can be imposed on a set of speech segments by using clustering. Our theory is that the cluster with the centroid that matches best to the test segment will contain the best matching segment, or will do so sufficiently often that the output of a simple exemplar-based speech enhancement algorithm will not be meaningfully degraded when utilizing this assumption.

1.3.2. Corpus compression

A corpus created from a smaller number of audio samples is likely to be less representative of the whole range of human speech than one created from a larger number, and thus less likely to find a good fit for unseen speech. This results in a large number of highly similar segments; so we propose using clustering to pick the most representative segments (codewords), instead of relying on a large number of audio samples to ensure good representation.

2. SPEECH ENHANCEMENT ALGORITHM

In our test system, as audio is received, the algorithm filters noise by searching for a match for each noisy test segment from a clean speech corpus. Taking a similar approach to Baby et al. [10], the speech corpus maintains full spectrum and mel representations of each segment, using the latter for matching and the former as the speech estimate in a Wiener filter, which filters the input signal [15].

3. SEARCH TREE-BASED MATCHING

3.1. Search method for the tree

The standard, linear mode of operation of the program is, for each test segment, to search through the corpus and use the best matching segment. Organizing the corpus segments into a tree structure significantly reduces the search space. Each node in the search tree is represented as a structure with a centroid and a list of child nodes. The search starts with the root node, which has an empty centroid and contains the first level segments at its children this is the first node to be searched. The search function compares each first level segment to the test segment and selects the best matching segment, which then becomes the search node. The new search nodes's children are compared to the test segment and the search function continues until a node with no children is found. This is used as the closest matching segment.

The standard, linear search is $O(n)$ in time complexity, where n is the number of segments in the corpus. The clustered matching is at best $O(log(n))$ in time complexity.

3.2. Constructing a search tree

The search tree is constructed using an iterative clustering function, used to approximate an ordering of the segments.

For clustering we used K-Means++, a variation of K-Means which results in better initial estimates of the centroids [20]. K-Means++ initially selects each centroid after the first preferring greater distances from already selected centroids. This means that the initial centroid estimates are

sufficiently spread out, but with sufficient randomness that the initialization is not deterministic. This was chosen instead of Hierarchical-Agglomerative (HA) clustering (which is more typical when the size of clusters inherent in a data set is not known a priori) as its quality in this application was not found to be any worse than HA clustering, yet it scales much better to large data sets.

A clean-speech segment in the frequency-domain is too high-dimensional and sparse to be accurately clustered. A lower-dimensional representation can be obtained with a mel filter bank. Experimentation showed that filtering into 40 frequency bins resulted in both good representation of speech and good clustering.

To cluster the corpus into a search tree we first take all the possible valid segments in the corpus and find initial centroids for W clusters using the K-Means++ initialization, with W being given as program input. Using those initial centroids each segment is assigned to the closest centroid. Like the search algorithm, ZNCC is used for the distance measure to determine the closeness of two segments.

The distance between two segments for the purpose of clustering is given by the following formula:

$$d(\mathbf{x}, \mathbf{y}) = 1 - R(\mathbf{x}, \mathbf{y}) \tag{3}$$

where $d(\mathbf{x}, \mathbf{y})$ is the distance between segment $\mathbf{x}$ and segment $\mathbf{y}$ and $R(\mathbf{x}, \mathbf{y})$ is a function giving the ZNCC for segment $\mathbf{x}$ and segment $\mathbf{y}$, as per (1).

The next step is to recalculate the centroid based on the cluster's members. For cluster C_c composed of N segments $C_c = \{\mathbf{S}_0, \mathbf{S}_1, ..., \mathbf{S}_{N-1}\}$, its centroid $\mathbf{c}_c$ is calculated as:

$$\mathbf{c}_{c,f,b} = \frac{\sum_{i=0}^{N-1} \mathbf{S}_{i,f,b}}{N} \tag{4}$$

where $\mathbf{c}_{c,f,b}$ is the power spectrum for centroid $\mathbf{c}_c$ belonging to cluster C_c, at frame f of the segment and frequency bin b of the frame.

This is repeated until a stop condition is reached. At this point, the centroids are set as the first level nodes. For each cluster the segments belonging to it are taken and the clustering process is repeated recursively until all the segments belong to a node of no more than W segments.

One drawback to this method is the larger memory usage of the generated corpora. With a search tree, each possible segment is represented individually, meaning that many frames are represented multiple times. Replacing leaf node centroids with references to all the original segments would resolve the issue, but the corpus would remain too large for many applications as the full corpus would still need to be retained. This issue will be addressed below.

4. CLUSTERING FOR CORPUS COMPRESSION

A solution to the issue of the size of a representative corpus can be found in a lossy compression of the corpus using clustering. Clustering can be used to determine groups of similar segments. When using a sufficiently large number of clusters the large set of training sentences can be approximated with the centroids of these groups. These centroids (or codewords) can then be used for search: either linear search which offers a representative set of segments to search, or tree-based search as described above.

For clustering for compression, Linde-Buzo-Gray initialization was used to obtain initial estimates of the centroids due to its ability to scale well with large data sets [21]. Euclidean distance was used for the initial clustering, then once the full number of codewords were found, the clusters were refined using ZNCC as the distance measure. Additionally, it was found that taking the medoids of the cluster, rather than the centroids on the last pass of the algorithm resulted in output that sounds more natural.

Combining corpus compression and search trees enables the speech enhancement algorithm to use a corpus that is almost as representative as a large corpus and able to function in real time on hardware without large amounts of memory.

5. EXPERIMENTAL STUDIES

5.1. Test Basis

Using our test system we can observe the reduction in time of matching with these two methods, along with memory usage and the associated quality tradeoffs.

Corpora were created using the WSJ0 training corpus using a search tree width W of 8 and a segment length L of 15. A quarter of the WSJ0 corpus was taken as the training set because, even with this, computing clustered and compressed corpora took significant time on the Intel Xeon hardware that was used to create the corpora and perform testing. Corpora were also created from smaller subsets of the full WSJ0 training corpus (approximately a 128th, a 256th and a 512th of the full set) to compare the benefits of compression over simply using a small input sample set. These sizes were selected to provide similar time factors to the compressed full sample set corpora.

The test set consisted of 330 samples from the WSJ0 testing set. Each sample was noised with the Aurora-4 set and two highly non-stationary noises, resulting in 2640 test files for each corpus to process. The Aurora-4 noised set were noised at 5-15 SNR with the following noises: airport, babble, car, restaurant, street, train. The non-stationary noises (a mobile phone polyphonic ringtone and a pop song) were used to noise samples at 0 SNR. The test files were enhanced by the algorithm then the STOI, PESQ and Segmental SNR (SSNR) intelligibility measures were found for the output, shown in Tables 1 and 2 averaged over all the samples for a given set.

5.2. Search-Tree based Enhancement

By clustering a corpus into a tree structure for search, the time taken to enhance audio with comparable quality is drastically reduced, as can be seen in Table 1.

The real time factor is a measure of how processing can keep up with real-time. A factor of 100 means 100x slower

Table 1: Comparison of enhancement using linear searching and tree-based searching with an uncompressed corpus created from a quarter of the WSJ0 training set ('WSJ0 / 4'). The factor of real time is shown, with a larger number indicating that enhancement took longer. The amount of working memory used is shown in megabytes.

				Aurora 4			Non-stationary		
Sample Set	Search method	Memory	Real Time Factor	STOI	PESQ	SSNR	STOI	PESQ	SSNR
WSJ0 / 4	Linear	2070	100.3579	0.8749	2.4975	-0.5760	0.9017	2.5034	-0.7546
WSJ0 / 4	Tree-Based	40641	0.0490	0.8870	2.4503	-0.7055	0.9040	2.4138	-0.8606

Table 2: Comparison of enhancement using tree-based searching with corpora of various sizes. An uncompressed corpus is compared with corpora of increasing numbers of codewords. An alternative to compression is shown with uncompressed corpora composed of smaller subsets of WSJ0. The amount of working memory used is shown in megabytes.

				Aurora 4			Non-stationary		
# Codewords	Sample Set	Memory	Real Time Factor	STOI	PESQ	SSNR	STOI	PESQ	SSNR
Uncompressed	WSJ0 / 4	40641	0.0490	0.8870	2.4503	-0.7055	0.9040	2.4138	-0.8606
512	WSJ0 / 4	13	0.0375	0.8485	2.2333	-1.2578	0.8789	2.2557	-1.1678
1024	WSJ0 / 4	26	0.0383	0.8542	2.2533	-1.2913	0.8830	2.2730	-1.1425
2048	WSJ0 / 4	51	0.0388	0.8654	2.3155	-1.1827	0.8873	2.2895	-1.1499
4096	WSJ0 / 4	103	0.0396	0.8779	2.3536	-1.0160	0.8932	2.2981	-1.0847
Uncompressed	WSJ0 / 512	311	0.0418	0.8545	2.2615	-1.1407	0.8866	2.2920	-1.0675
Uncompressed	WSJ0 / 256	650	0.0423	0.8583	2.2962	-1.0834	0.8889	2.3224	-1.0240
Uncompressed	WSJ0 / 128	1235	0.0434	0.8665	2.3173	-1.0234	0.8928	2.3352	-0.9897

than real-time.

The speed of matching is four orders of magnitude faster. However, this comes at a cost. With the search-tree, the uncompressed corpus takes up 40GB in working memory, instead of just 2GB.

5.3. Corpus Compression

To reduce the memory required for the corpus, we have implemented corpus compression into codewords.

In our most extreme case, compressing our quarter of WSJ0 corpus to only 512 codewords results in a size of only 6MB, but with a noticable decline in quality across intelligibility measures. As the number of codewords is increased the quality increases, approaching that of the uncompressed corpus at 4096 codewords with a reasonable size of 45MB. However, at this point linear search exceeds real-time functioning (1.69x). Generally speaking, compressing a larger corpus gives better results than using a smaller, uncompressed corpus.

5.4. Both Methods Combined

The results for testing carried out with tree-based searching on compressed corpora are shown in Table 2, in comparison with small, uncompressed corpora. In terms of quality, the 4096 codeword corpus is reasonably close to the uncompressed corpus across the quality measures. In some quality measures the smaller, uncompressed corpora beat the 4096 codeword corpus, but they do so at a cost to corpus size. It is possible that going to 8192 codewords would give further quality improvement, but the clustering will take on the order of weeks.

With this combination of methods, the 4096 codeword corpus satisfies several key requirements when being used for clustered matching: it provides comparable quality to the uncompressed corpus with linear matching, it is sufficiently quick for real-time and its size in memory is a order of magnitude smaller than the uncompressed corpus without a search tree.

6. CONCLUSION

We have proposed an approach to searching a speech corpus, which maintains representiveness while reducing the time to enhance speech by four orders of magnitude, well under the real-time barrier. When compression is used, this reduction in processing time is also accompanied by a decrease in memory usage of up to two orders of magnitude, when compared to the uncompressed corpus without a search tree. These benefits are obtained while retaining much of the output quality. Additionally, in quality and memory requirements, a larger, compressed corpus performs better than simply using a smaller, uncompressed corpus.

These improvements enable the searching of more representative corpora in real-time, along with the possibility to use more sophisticated enhancement in real-time applications and low power embedded platforms to perform the enhancement.

7. REFERENCES

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, Apr 1979.

[2] T. V. Sreenivas and Pradeep Kirnapure, "Codebook constrained wiener filtering for speech enhancement," *IEEE Trans. Speech Audio Process*, vol. 4, pp. 383–389, 1996.

[3] Yi Hu and Philipos C. Loizou, "Speech enhancement based on wavelet thresholding the multitaper spectrum," in *IEEE TRANS. SPEECH AUDIO PROC*, 2004, pp. 59–67.

[4] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, Apr 1985.

[5] Hossein Sameti and Li Deng, "Nonstationary-state hidden markov model representation of speech signals for speech enhancement," *Signal Processing*, vol. 82, pp. 205–227, January 2002.

[6] Eric W. Healy, Sarah E. Yoho, Yuxuan Wang, and DeLiang Wang, "An algorithm to improve speech recognition in noise for hearing-impaired listeners," *The Journal of the Acoustical Society of America*, vol. 134, no. 4, pp. 3029–3038, 2013.

[7] Eric W. Healy, Sarah E. Yoho, Jitong Chen, Yuxuan Wang, and DeLiang Wang, "An algorithm to increase speech intelligibility for hearing-impaired listeners in novel segments of the same noise type," *The Journal of the Acoustical Society of America*, vol. 138, no. 3, pp. 1660–1669, 2015.

[8] Jitong Chen, Yuxuan Wang, Sarah E. Yoho, DeLiang Wang, and Eric W. Healy, "Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises," *The Journal of the Acoustical Society of America*, vol. 139, no. 5, pp. 2604–2612, 2016.

[9] Emre Yilmaz, Deepak Baby, and Hugo Van hamme, "Noise robust exemplar matching for speech enhancement: applications to automatic speech recognition," in *INTERSPEECH*, 2015.

[10] Deepak Baby, Tuomas Virtanen, Jort F. Gemmeke, and Hugo Van hamme, "Coupled dictionaries for exemplar-based speech enhancement and automatic speech recognition," *IEEE Transactions on Audio Speech and Language Processing*, vol. 23, no. 11, pp. 1788–1799, November 2015.

[11] Bhiksha Raj, Rita Singh, and Tuomas Virtanen, "Phoneme-dependent nmf for speech enhancement in monaural mixtures," in *INTERSPEECH*, 01 2011, pp. 1217–1220.

[12] Ji Ming, Ramji Srinivasan, and Danny Crookes, "A corpus-based approach to speech enhancement from nonstationary noise," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 822–836, 2011.

[13] R. M. Nickel, R. F. Astudillo, D. Kolossa, S. Zeiler, and R. Martin, "Inventory-style speech enhancement with uncertainty-of-observation techniques," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. mar 2012, IEEE.

[14] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 7–19, 01 2015.

[15] Ji Ming and Danny Crookes, "Speech enhancement based on full-sentence correlation and clean speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 3, pp. 531–543, 1 2017.

[16] Anna Ogawa, Keizo Kinoshita, Toshikazu Hori, Takeshi Nakatani, and Atsushi Nakamura, "Fast segment search for corpus-based speech enhancement based on speech recognition technology," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1557–1561.

[17] R. M. Nickel and R. Martin, "Memory and complexity reduction for inventory-style speech enhancement systems," in *2011 19th European Signal Processing Conference*, Aug 2011, pp. 196–200.

[18] X. L. Zhang and D. Wang, "A deep ensemble learning method for monaural speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 967–977, May 2016.

[19] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[20] David Arthur and Sergei Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA, 2007, SODA '07, pp. 1027–1035, Society for Industrial and Applied Mathematics.

[21] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, Jan 1980.