TRAINING PHONEME MODELS FOR SINGING WITH "SONGIFIED" SPEECH DATA

Anna M. Kruspe
Fraunhofer IDMT, Ilmenau, Germany
kpe@idmt.fraunhofer.de

ABSTRACT

Speech recognition in singing is a task that has not been widely researched so far. Singing possesses several characteristics that differentiate it from speech. Therefore, algorithms and models that were developed for speech usually perform worse on singing.
One of the bottlenecks in many algorithms is the recognition of phonemes in singing. We noticed that this recognition step can be improved when using singing data in model training, but to our knowledge, there are no large datasets of singing data annotated with phonemes. However, such data does exist for speech.
We therefore propose to make phoneme recognition models more robust for singing by training them on speech data that has artificially been made more "song-like". We test two main modifications on speech data: Time stretching and pitch shifting. Artificial vibrato is also tested. We then evaluate models trained on different combinations of these modified speech recordings. The utilized modeling algorithms are Neural Networks and Deep Belief Networks.

1. INTRODUCTION

Automatic speech recognition has been a field of research for more than 30 years now and encompasses a large variety of research topics. However, speech recognition algorithms have so far only rarely been adapted to singing. One of the reasons for this seems to be that most of these tasks get harder when using singing because singing data has different characteristics, which are also often more varied than in pure speech [13]. For example, the typical fundamental frequency for women in speech is between 165 and $200Hz$, while in singing it can reach more than $1000Hz$. Other differences include harmonics, durations, pronunciation, and vibrato.
Speech recognition in singing has many interesting practical applications, such as automatic lyrics-to-music alignment, keyword spotting in songs, language identification of musical pieces or even full lyrics transcription.
A first step in many of these tasks is the recognition of phonemes in the audio recording. We showed in [12] that phoneme recognition tends to act as a bottleneck in tasks such as language identification and keyword spotting in singing. Other publications also demonstrate that phoneme recognition on singing is more difficult than on speech [15] [6] [13]. This is further compounded by the models which have usually been trained on pure speech data.
As shown on a small scale in [6] and [12], recognition gets better when singing is used as part of the training data. The big problem with this is the lack of phoneme-annotated singing data sets.
When there is a scarcity of suitable training data, attempts are often made to generate such data artificially. For example, this is often done when models for noisy speech are required [11] [7]. In this paper, we therefore propose to make existing speech data sets more "song-like" and use these modified datasets to train models for phoneme recognition in singing. We test this procedure with the commonly used TIMIT speech dataset [10] and train Neural Networks (NNs) and Deep Belief Networks (DBNs) on modified versions of it. We then test the models' performances on an unaccompanied singing dataset and on the test section of TIMIT.
This paper is structured as follows: We first give an introduction to the state of the art in section 2 and describe the datasets in section 3. We then present our new approach in section 4. Section 5 contains our experiments and their results. Finally, we give a conclusion in section 6 and suggest future work in section 7.

2. STATE OF THE ART

As described in [13] and in [12], there are significant differences between speech and singing data, such as pitch and harmonics, vibrato, phoneme durations and pronunciation. This makes phoneme recognition on singing harder than on speech.
Several approaches to this task have been published. In [5], Gruhne et al. describe a classical approach that employs feature extraction and various machine learning algorithms to classify singing into 15 phoneme classes. It also includes a step that removes non-harmonic components from the signal. The best result of 58% correctly classified frames is achieved with Support Vector Machine (SVM) classifiers. The approach is expanded upon in [17].
Fujihara et al. describe an approach using Probabilistic Spectral Templates to model phonemes in [4]. The pho-

neme models are gender-specific and only model five vowels, but also work for singing with instrumental accompaniment. The best result is 65% correctly classified frames.
Mesaros presented a complex approach that is based on Hidden Markov Models which are trained on Mel-Frequency Cepstral Coefficients (MFCCs) and then adapted to singing using three phoneme classes separately [15] [14]. The approach also employs language modeling and has options for vocal separation and gender and voice adaptation. The achieved phoneme recognition rate (accuracy) on unaccompanied singing is -6.4% without adaptation and 20% with singing adaptation using 40 phonemes (the negative value is equivalent to a Levenshtein distance of 1.064, which means that there were more insertion, deletion, or substitution errors than phoneme instances). The results also improve when using gender-specific adaptation (to an average of 18.75%) and even more when language modeling is included (to 33.4%).
Finally, Hansen presents a system in [6] which combines the results of two Multilayer Perceptrons (MLPs), one using MFCC features and one using TRAP (Temporal Pattern) features. Training is done with a small amount of singing data. Viterbi decoding is then performed on these posterior probabilities. On a set of 27 phonemes, this approach achieves a recall of up to 48%.
It should be obvious from this overview that comparing these approaches is not easily possible. Each one uses a different dataset, a different phoneme set, and different evaluation measures.

3. DATASETS

3.1 Speech data

For training our phoneme recognition models, we used the well-known TIMIT speech dataset [10]. Its training section consists of 4620 phoneme-annotated English utterances spoken by native speakers. Each utterance is a few seconds long.
The test section of TIMIT contains similar 1680 similarly phoneme-annotated utterances. We used it to test the general performance of our models.

3.2 Singing data

To test the performance on singing data, we used the data set previously presented in [6] and [12]. It consists of the vocal tracks of 19 commercial pop songs in studio quality. We use unaccompanied singing to avoid a possible source of interference. They do not contain background music, but have been post-processed (e.g. EQ, compression, reverb). Some of them contain choir singing. Of these 19 songs, 12 were annotated with time-aligned phonemes and could therefore be used for our phoneme recognition experiments. We split these 12 songs into 562 clips, each of which roughly represents a line of the songs' lyrics.

4. PROPOSED APPROACH

An overview of our approach is shown in figure 1. We first generate five variants of the TIMIT speech dataset (training set). MFCC features are then extracted from these new datasets and used to train two models per dataset: A Neural Network and a Deep Belief Network.
Similarly, MFCCs are extracted from the TIMIT Test set and from the singing dataset. The ten previously trained models are used to recognize phonemes on these test datasets. Viterbi decoding can then be used to generate phoneme sequences. Finally, the results are evaluated.

4.1 Training data modifications

In order to make the training data more "song-like", we developed several variants of this dataset. Table 1 shows an overview over the five datasets generated from TIMIT using three modifications. Dataset N is the original TIMIT training set. For dataset P, four of the eight blocks of TIMIT were pitch-shifted. For dataset T, five blocks were time-stretched and vibrato was applied to two of them. In dataset TP, the same is done, except with additional pitch-shifting. Finally, dataset M contains a mix of these modified blocks.
In detail, the modifications were performed in the following way:

Time stretching For time stretching, we used the phase vocoder from [3], which is an implementation of the Flanagan/Dolson phase vocoder [9] [2]. This algorithm works by first performing a Short-Time Fourier Transform (STFT) on the signal and then resampling the frames to a different duration and performing the inverse Fourier transform.
As demonstrated in [12], time variations in singing are mainly performed on vowels and are often much longer than in speech. We therefore used the TIMIT annotations to only pick out the vowel segments from the utterances. They were modified randomly to a duration between 5 and 100 times the original duration and then re-inserted into the utterance. This effectively leads to more vowel frames in the training data, but since there is already a large amount of instances for each phoneme in the original training data, the effects of this imbalance should be negligible.

Pitch shifting To pitch-shift the signal, we used code from the freely available Matlab tool *AutoTune Toy* [1] which also implements a phase vocoder. In this case, the fundamental frequency is first detected automatically. The signal is then stretched or expanded to obtain the new pitch and interpolated to retain the original duration.
Using the TIMIT annotations, we split the utterance up into individual words, then generate a pitch-shifted version of each word and concatenate the results. Pitches are randomly selected from a range between 60% and 120% of the original pitch.

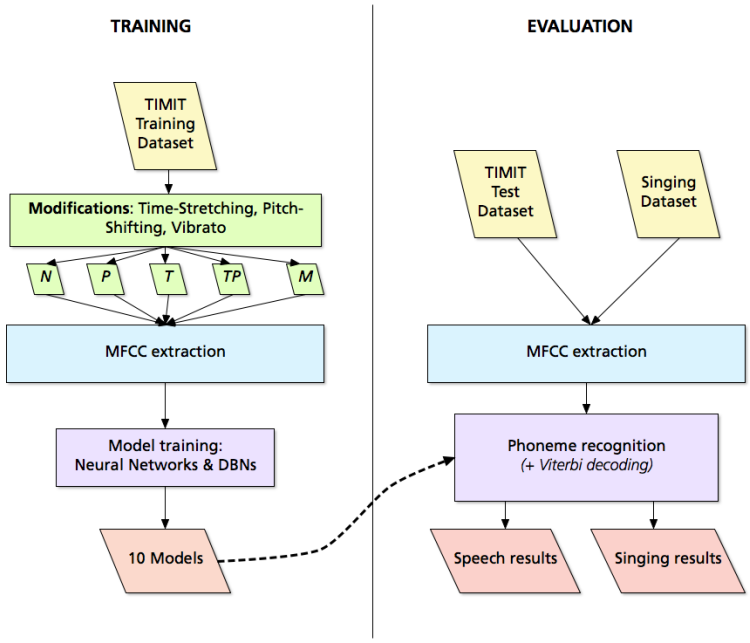


Figure 1: Overview of our phoneme recognition system

	N	P	T	TP	M
DR1	N	N	N	N	N
DR2	N	N	N	N	N
DR3	N	N	N	N	P
DR4	N	N	T	TP	TV
DR5	N	P	T	TP	TPV
DR6	N	P	T	TP	TV
DR7	N	P	TV	TPV	P
DR8	N	P	TV	TPV	TPV

Table 1: The five TIMIT variants that were used for training (rows are TIMIT blocks, columns are the five datasets). Symbols: N - Unmodified; P - Pitch-shifted; T - Time-stretched; V - Vibrato

Vibrato The code for vibrato generation was also taken from *AutoTune Toy*. It functions by generating a sine curve and using this as the trajectory for the pitch shifting algorithm mentioned above. We used a sine of amplitude 0.2 and frequency $6Hz$.
In singing, vibrato is commonly done on long sounds, which are usually vowels. Since spoken vowels are usually very short, vibrato cannot be perceived on them very well. We therefore only applied vibrato when time stretching was also applied. Vibrato was then added to the extracted and stretched vowels.

4.2 Models

Using the generated data, we trained models using two machine learning algorithms: Classical Neural Networks (NNs) and Deep Belief Networks (DBNs). Both were implemented using the Theano framework for Python [1]. In both cases, we first extracted Mel-Frequency Cepstral Coefficients (MFCCs) and retainen the first 13 plus their deltas and double-deltas as features. We also expanded the training data to use 9 context frames. The output layer represents the 39 phonemes of the CMU Sphinx phoneme set [2]. To make the training more exact, these phonemes were split into triphones, making the dimension of the output layer 117.
Our first models are traditional Neural Networks with two layers of 200 units each.
In recent publications, DBNs have been used very successfully for phoneme recognition (e.g. [16]). We therefore also trained DBNs on the speech data. We chose an architecture with three hidden layers and 300 units each. The first hidden layer is a Gaussian RBM.
Both models are used to generate posterior phoneme probabilities. The results for the triphone states of each phoneme are summed up into one probability for the phoneme. We then run a simple Viterbi decoding on these posteriors to generate phoneme sequences. In this decoding, all phonemes have equal transition probabilities, only the insertion penalty is variable (i.e., the transition probability to another phone). No language models are employed. We keep this post-processing simple on purpose so that the results of the various models are easily comparable.

4.3 Evaluation measures

As described in section 2, there is no single common evaluation measure for phoneme recognition in singing. We decided to compare our results using three measures:

[1] `http://www.deeplearning.org`, last checked 04/29/15
[2] `http://cmusphinx.sourceforge.net/`, last checked 04/29/15

Percentage of correct frames This measure describes the percentage of correctly classified frames. Correct in this case means that the exact phoneme was chosen for this frame during Viterbi decoding. A similar measure was used by Fujihara [4] and Gruhne [5].

Phoneme error rate This is the most commonly used evaluation measure in phoneme recognition for speech. It is equal to the Levenshtein distance normalized by the length of the ground truth phoneme sequence:

$$PER = \frac{D + I + S}{N} \quad (1)$$

where D are deletions, I are insertions, and S are substitutions of phonemes and N is the length of the sequence.
The accuracy measure used by Mesaros [15] [14] is the same as $1 - PER$.

Weighted phoneme error rate Mesaros also uses a measure called $correct$ which ignores insertions. This makes sense if we assume that the phoneme results are used afterwards by an algorithm that is tolerant to insertions. We decided to go one step further and assume that if algorithms are tolerant to insertions, they can also be somewhat tolerant to deletions. For cases like this, Hunt suggested a weighted error rate that punishes insertions and deletions less heavily than substitutions [8]:

$$PER_{Hunt} = \frac{0.5D + 0.5I + S}{N} \quad (2)$$

5. EXPERIMENTS

We performed our experiments by training a set of models on all five TIMIT variants where all other parameters were left equal. We then classified two sets of data with these models: The unmodified "Test" part of the TIMIT speech dataset (which was not used in training) and our singing dataset. On these phoneme posterior probabilities, we ran the described simple Viterbi algorithm. The insertion penalty was optimized to generate phoneme strings who were closest in length to the ground truth phoneme strings. The three evaluation measures described in 4.3 were then calculated on the result of the Viterbi decoder.
We tested two machine learning algorithms: Neural Networks and Deep Belief Networks.

5.1 Neural Network models

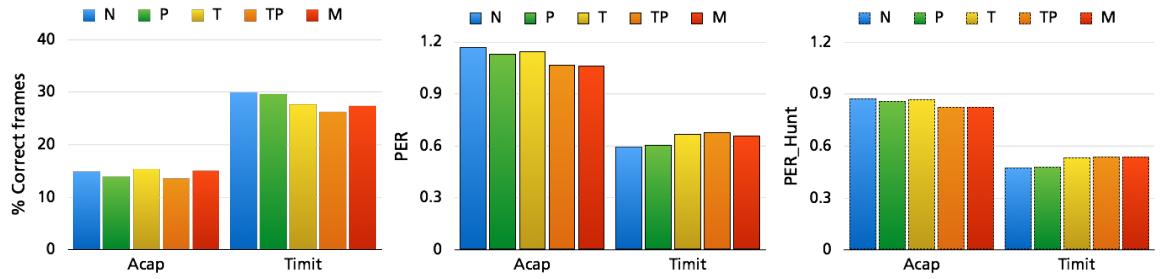
Figure 2 shows the results of the Neural Network models. As figure 2a demonstrates, results for singing are generally worse than for speech. The base result for singing is a percentage of correct frames of 14.9% (model trained on the original TIMIT dataset which is denoted as *N* here). When comparing the models trained on the various TIMIT modifications, a slight improvement is observed for the *T* and *M* variants. For the *T* dataset, which includes randomly time-stretched vowels, the result improves to 15.4%. This is a very small improvement, but it is still interesting to note. In contrast, none of the modifications improved the result on the speech data at all. The base result here is 30%. (It should be noted that much higher figures can be found in literature, but we have not yet tested improvements like language models or adaptations. Our focus for now was to compare the different TIMIT modifications).
When looking at the phoneme error rate in figure 2b instead of the pure frame accuracy, the results become more visible. The base phoneme error rate for singing is 1.16, but falls to 1.07 for the *TP* and *M* modifications. For speech, it rises from 0.6 to 0.68 (*TP*) and 0.66 (*M*) instead. The P and *T* modifications form a middle ground here. The *P* variant (randomly pitch-shifted words) does not change the results very much in either direction: It decreases the error rate on singing by 0.03 and increases it on speech by less than 0.01. The *T* variant (randomly time-stretched vowels) decrease the error on singing by just 0.02, but increase it on singing by 0.07.
If we weight insertions and deletions lower than modifications, the phoneme error rates decrease generally (see figure 2c). The described effects are still active when using this evaluation measure. The error rate falls from 0.88 to 0.83 on singing, and rises from 0.48 to 0.54 on speech. The tendency for *P* and *T* is similar here.

5.2 Deep Belief Network models

Figure 3 shows the same evaluation measures for the Deep Belief Networks. In general, the results are better and the effect of the various training sets is similar, but more pronounced.
The base percentage of correct frames is 14% here and rises to 19% when training on the randomly timed dataset *T*. On speech data, the best result is 38% for models trained on the original TIMIT data and falls for all other variants.
The phoneme error rate falls from 1 to 0.91 on the singing data. Again, the results are best when the models are trained on the *TP* or *M* datasets, with the model trained on *T* performing just slightly worse. The lowest error rate on speech is 0.41 with the *N* model.
The weighted phoneme error rate sinks from 0.77 to 0.71 on the singing data.

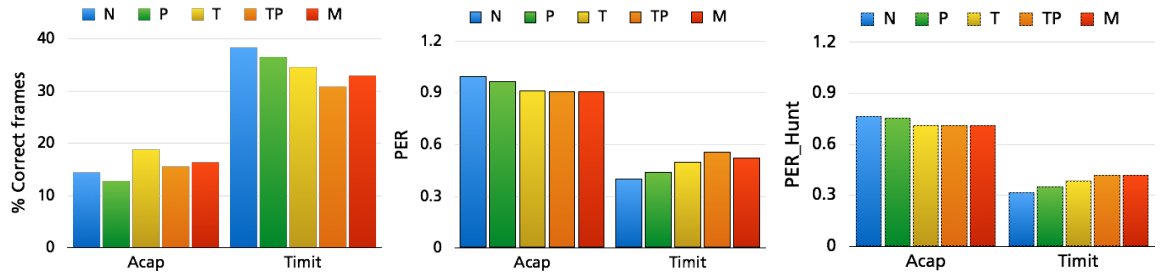
5.3 Confusion of Deep Belief Networks

After evaluating the general performance of the Deep Belief Networks, we examined the results in detail. As an example, table 2 shows the phoneme-wise results for the singing data. The first two columns lists the frame-wise precisions and recalls when using the *N* model, the two columns after that show the same values for the *M* model, and the last column lists the three phonemes with which the concerned phoneme is confused most frequently when using the *M* model (except `sil`). This leads to several interesting discoveries.
It turns out that the precisions of long vowels such as `aa`, `iy`, or `oy` improve when using the *M* model for recognition, but some consonant accuracies become worse. This makes sense since the *M* modifications place an emphasis on vowels by randomly stretching them. The consonant



(a) Percentage of correct frames (b) Phoneme error rate (c) Weighted phoneme error rate

Figure 2: Evaluation measures for the results obtained with Neural Network models on singing data (Acap) and on speech data (Timit). The models were trained with the five different Timit variants (different colors).



(a) Percentage of correct frames (b) Phoneme error rate (c) Weighted phoneme error rate

Figure 3: Evaluation measures for the results obtained with Deep Belief Network models on singing data (Acap) and on speech data (Timit). The models were trained with the five different Timit variants (different colors).

results may become worse because the training data also contains randomly pitch-shifted versions, which may not be a natural modification (this can be verified by looking at the results of the *T* model). Consonants generally seem harder to recognize since they are shorter than vowels and, for the most part, are less static over their duration. Some very bad consonant results can also be explained because they occur very rarely in the singing dataset (e.g. `zh`, `oy`). The most frequent confusion for most phonemes occurs with the `sil` state, which serves as the general "non-phoneme" state. This confusion is not displayed here.

Consonants are often confused with similar consonants (e.g. `m`/`n`) or with softer consonants that can be extended over a longer duration (e.g. `s`, `f`). This may be related to singing technique. However, they are also frequently confused with some vowels, particularly `uh` and `iy`. This might be caused by slight timing inaccuracies in the training annotations which become exaggerated by the time-stretching, or even by some merging of neighboring phonemes by the speaker.

Longer vowels are almost exclusively confused with other long vowels. This poses a contrast to speech, where they are usually confused with similar short vowels (e.g. `aa` → `ah`).

This becomes more conclusive when considering the confusions of short vowels. In the singing data, short vowels are very often confused with similar long vowels (e.g. `eh` → `ae`). When stretching out such short vowels in singing, singers will automatically change to such a longer vowel. Additionally, some vowels are confused with more "open" vowels (e.g. `ey` → `ae`). This is also caused by singing technique. These two very interesting effects could be exploited to improve phoneme recognition on singing in the future.

6. CONCLUSION

In this paper, we evaluated phoneme models trained on various artificially "songified" variants of the TIMIT speech dataset. The reason for this is the lack of phoneme-annotated singing datasets. We generated five such variants by randomly time-stretching vowels, randomly pitch-shifting words, and by adding vibrato to long vowels.

MFCC features were extracted from these datasets and then used to train two models each: A Neural Network and a Deep Belief Network. We then used these models to recognize phonemes in singing data and in unrelated speech data. No additional mechanics were used to improve the results, such as language modeling or gender or speaker adaptation.

In general, the results are not as good as the state of the art for the speech data. For the singing data, it is very hard to compare the results to the state of the art because other publications use different datasets, phoneme sets, and evaluation measures. However, this was not necessarily our goal - we were mainly interested in the comparative performance of the various models.

As expected, recognizing phonemes in speech seems to be much easier than in singing. Deep Belief Models performed better than their Neural Network counterparts in all test cases. For speech, the models trained on the unmodified TIMIT dataset always performed best. The best result is 38% correctly classified frames, a phoneme error rate of 0.41, and a weighted phoneme error rate of 0.32.

For singing, the models trained on the modified TIMIT da-

Ph.	Prec. *N*	Rec. *N*	Prec. *M*	Rec. *M*	Conf. *M*
aa	0.18	0.08	0.35	0.09	ao, ay, ow
iy	0.33	0.27	0.43	0.25	ey, uh, ae
ch	0.0	0.02	0.01	0.03	s, sh, iy
zh	0.26	0.02	0.0	0.0	iy, y, sh
eh	0.06	0.11	0.13	0.15	ae, ey, aa
ah	0.0	0.35	0.03	0.23	aa, er, ae
ao	0.39	0.17	0.25	0.14	aa, ow, l
ih	0.01	0.08	0.05	0.11	ey, iy, ae
ey	0.36	0.17	0.26	0.2	iy, ae, ay
aw	0.1	0.07	0.08	0.09	aa, ae, ay
ay	0.27	0.44	0.23	0.44	aa, ae, iy
ae	0.38	0.16	0.4	0.16	aa, ay, aw
er	0.22	0.1	0.19	0.08	aa, ae, eh
ng	0.07	0.16	0.06	0.11	n, uh, iy
sh	0.58	0.05	0.51	0.09	s, jh, z
th	0.0	0.0	0.0	0.0	dh, er, ey
oy	0.0	0.0	0.0	0.0	ao, ay, aa
dh	0.04	0.14	0.04	0.08	er ,iy ,uh
ow	0.07	0.23	0.16	0.25	ae, ae, aa
hh	0.14	0.09	0.09	0.15	iy, uh, sh
jh	0.07	0.08	0.12	0.07	y, z, sh
b	0.13	0.19	0.09	0.24	ey, m, iy
d	0.02	0.19	0.02	0.1	iy, n, er
g	0.04	0.36	0.03	0.32	y, ow, n
f	0.02	0.13	0.02	0.5	s, er, iy
k	0.02	0.37	0.05	0.31	iy, y ,uh
m	0.26	0.24	0.13	0.22	uh, n, er
l	0.1	0.14	0.11	0.15	ao, er, ow
n	0.18	0.28	0.2	0.25	uh, er, uw
uh	0.23	0.02	0.15	0.02	er, ih, eh
p	0.01	0.15	0.01	0.1	er, iy, l
s	0.41	0.41	0.44	0.46	z, iy, er
r	0.16	0.24	0.17	0.18	er, aa, ao
t	0.0	0.29	0.0	0.42	s, sh, iy
w	0.24	0.26	0.19	0.2	ao, l, uw
v	0.0	0.02	0.0	0.25	er, aa, m
y	0.31	0.08	0.16	0.12	iy , y, uh
z	0.28	0.18	0.28	0.17	s, iy, n
uw	0.13	0.35	0.12	0.33	uh, er, iy

Table 2: Results per phoneme (singing data): Precision and recall with the *N* and *M* models, and most frequent confusions with *M* model (except `sil`)

taset produced better results. The best result is for singing is 18% correctly classified frames, a phoneme error rate of 0.91, and a weighted phoneme error rate of 0.71. The improvement over the models trained on the unmodified TIMIT data is 6% for the correctly classified frames, 0.09 for the phoneme error rate, and 0.06 for the weighted phoneme error rate.

The models trained on data that was only pitch-shifted only showed a very slight difference when compared to the original data. MFCCs are supposed to be pitch-invariant, and pitch-shifting therefore does not seem to make a big difference. This modification might be useful when using other features, though. A bigger improvement on singing data was achieved when training the models on time-stretched speech data. In fact, this dataset generated the highest percentage of correctly classified frames. In this time-stretched dataset, we also applied vibrato to the stretched vowels, which happens naturally in singing. However, since the effect of pitch-shifting seemed to be small, we assume that vibrato did not have a big effect either.

There were also two datasets where both modifications (time-stretching and pitch-shifting) were mixed. Both produced the best phoneme error rates in singing.

The results were also analyzed on a phoneme-wise basis. It turned out that vowels were recognized more exactly with the modified models, while consonants were recognized somewhat worse. This may be caused by the emphasis of the generated data on longer vowels.

The most interesting effect seen in the confusion matrices is the confusion of short vowels with similar longer vowels. This has a foundation in singing technique and would be interesting to further explore to improve phoneme recognition in singing.

In general, we showed that phoneme recognition in singing can be improved when training models on artificial singing data. This finding can now be used to improve other approaches. For example, it can be combined with the techniques described in [15].

7. FUTURE WORK

As described in section 5.3, many phoneme confusions may arise from inexact or unnatural time stretching on the speech recordings. A more natural approach to this is required and we need to make sure that stretched vowels do not "leak" into neighboring consonants. We also noticed that short vowels in singing often shift towards their long versions. We will exploit this interesting effect in future phoneme recognition approaches, e.g. by allowing these confusions or composing vowels of several states.

In this paper, we tried to apply three characteristics of singing to speech recordings, but there are more, such as different pronunciations and different forming of sounds. Such other characteristics could also be tested in a similar way. Conversely, we could also attempt to make our features and models more robust to these variations. In the past, this has often been done by adapting models trained on speech to singing in some way (also see section 2). Adaptations to gender or voice also proved helpful.

We kept the approach fairly simple for now, but the results could be improved by employing language modeling in the recognition process. We will implement this in future versions.

A possible alternative would be creating a dataset from polyphonic music data by using the lyrics and force-aligning them.

Finally, it will be interesting to see how the results of this phoneme recognition approach can be applied to practical tasks, such as lyrics-to-music alignment, keyword spotting, and language identification. For these purposes, the algorithm must also be tested on accompanied singing data.

8. REFERENCES

[1] C. Arft. AutoTune Toy, 2010. Web resource, Last checked: 4/29/15.

[2] M. Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[3] D. P. W. Ellis. A phase vocoder in Matlab, 2002. Web resource, Last checked: 04/29/15.

[4] H. Fujihara, M. Goto, and H. G. Okuno. A novel framework for recognizing phonemes of singing voice in polyphonic music. In *WASPAA*, pages 17–20. IEEE, 2009.

[5] M. Gruhne, K. Schmidt, and C. Dittmar. Phoneme recognition on popular music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, September 2007.

[6] J. K. Hansen. Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients. In *9th Sound and Music Computing Conference (SMC)*, pages 494–499, Copenhagen, Denmark, 2012.

[7] H.-G. Hirsch and D. Pearce. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ISCA ITRW ASR*, pages 29–32, 2000.

[8] M. J. Hunt. Figures of merit for assessing connected-word recognisers. *Speech Communication*, 9(4):329–336, 1990.

[9] R. M. Golden J. L. Flanagan. Phase vocoder. *Bell System Technical Journal*, pages 1493–1509, November 1966.

[10] J. S. Garofolo et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus. Technical report, Linguistic Data Consortium, Philadelphia, 1993.

[11] C. Jankowski, A. Kalyanswamy, S. Basson, and J. Spitz. NTIMIT: A phonetically balanced, continuous speech telephone bandwidth speech database. *ICASSP*, pages 109–112, 1990.

[12] A. M. Kruspe. Keyword spotting in a-capella singing. In *15th International Conference on Music Information Retrieval (ISMIR)*, Taipei, Taiwan, 2014.

[13] A. Loscos, P. Cano, and J. Bonada. Low-delay singing voice alignment to text. In *Proceedings of the ICMC*, 1999.

[14] A. Mesaros and T. Virtanen. Automatic recognition of lyrics in singing. *EURASIP J. Audio, Speech and Music Processing*, 2010, 2010.

[15] A. Mesaros and T. Virtanen. Recognition of phonemes and words in singing. In *ICASSP*, pages 2146–2149. IEEE, 2010.

[16] A.-R. Mohamed, G. E. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech, Lang. Process*, pages 14–22, 2012.

[17] G. Szepannek, M. Gruhne, B. Bischl, S. Krey, T. Harczos, F. Klefenz, C. Dittmar, , and C. Weihs. *Classification as a tool for research*, chapter Perceptually Based Phoneme Recognition in Popular Music. Springer, Heidelberg, 2010.