Technische Universität Ilmenau
Fakultät für Elektrotechnik und Informationstechnik

Application of Speech Recognition Algorithms to Singing

PhD Thesis at Fraunhofer Institute for Digital Media Technology

Submitted by: Anna Marie Kruspe

Submitted on:

Course of study: Media Technology

Matriculation Number: 39909

Advisor: Prof. Dr.-Ing. Dr. rer. nat. h.c. mult. Karlheinz Brandenburg

Abstract

The Higgs boson or Higgs particle is an elementary particle initially theorised in 1964,[6][7] and tentatively confirmed to exist on 14 March 2013.[8] The discovery has been called "monumental"[9][10] because it appears to confirm the existence of the Higgs field,[11][12] which is pivotal to the Standard Model and other theories within particle physics. In this discipline, it explains why some fundamental particles have mass when the symmetries controlling their interactions should require them to be massless, and?linked to this?why the weak force has a much shorter range than the electromagnetic force.

Kurzfassung

Das Higgs-Teilchen gehört zum Higgs-Mechanismus, einer schon in den 1960er Jahren vorgeschlagenen Theorie, nach der alle fundamentalen Elementarteilchen (beispielsweise das Elektron) ihre Masse erst durch die Wechselwirkung mit dem allgegenwärtigen Higgs-Feld erhalten. Als einziges Teilchen des Standardmodells ist das Higgs-Boson experimentell noch nicht vollständig gesichert.

Acknowledgements

Thanks to Leonard Hofstadter and thanks to my mee-maw.

Table of Contents

1 Introduction

This is my introduction...

2 Technical Background

2.1 General processing chain

2.2 Audio features

2.2.1 Perceptive Linear Predictive features (PLPs)

PLP features, first introduced in [1], are among the most frequently used features in speech processing. They are based on the idea to use knowledge about human perception to emphasize important speech information in spectra while minimizing the differences between speakers. We use a model order of 13 in two experiments and one of 32 in another. Deltas and double deltas between frames are also calculated. We test PLPs with and without RASTA pre-processing [2].

2.2.2 Mel-Frequency Cepstral Coefficients (MFCCs)

Just like PLPs, MFCCs are frequently used in all disciplines of automatic speech recognition [3]. We kept 20 cepstral coefficients for model training. Additionally, we calculated deltas and double deltas.

2.2.3 Shifted Delta Cepstrum (SDCs)

Shifted Delta Cepstrum features were first described in [4] and have since been successfully used for speaker verification and language identification tasks on pure speech data [5] [6] [7]. They are calculated on MFCC vectors and take their temporal evolution into account. Their configuration is described by the four parameter $N - d - P - k$, where N is the number of cepstral coefficients for each frame, d is the time context (in frames) for the delta calculation, k is the number of delta blocks to use, and P is the shift between consecutive blocks. The delta cepstrals are then calculated as:

$$\Delta c(t) = c(t + iP + d) + c(t + iP - d), 0 <= i <= k \tag{2.1}$$

with $c \in [0, N-1]$ as the previously extracted cepstral coefficients. The resulting k delta cepstrals for each frame are concatenated to form a single SDC vector of the length kN. We used the common parameter combination $N = 7, d = 1, P = 3, k = 7$.

2.2.4 TempoRal Patterns (TRAP)

2.3 Machine learning algorithms

This section describes the various machine learning algorithms employed throughout this thesis. Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), and Support Vector Machines (SVMs) are three traditional approaches that are used as the basis of many new approaches, and were used for several starting experiments. i-Vector processing is a relatively new, more sophisticated approach that bundles several other machine learning techniques.
In recent years, Deep Learning has become the standard for machine learning applications []. This chapter also describes two of those new approaches that were used in this work: Deep Neural Networks (DNNs) and Deep Belief Networks (DBNs).

2.3.1 Gaussian Mixture Models

2.3.2 Hidden Markov Models

2.3.2.1 Duration modeling

2.3.3 i-Vector processing

I-Vector (identity vector) extraction was first introduced in [8] and has since become a state-of-the-art technique for various speech processing tasks, such as speaker verification, speaker recognition, and language identification [9]. To our knowledge, it has not been used for any Music Information Retrieval tasks before.
The main idea behind i-vectors is that all training utterances contain some common trends, which effectively add irrelevance to the data in respect to training. Using i-vector extraction, this irrelevance can be filtered out, while only the unique parts of the data relevant to the task at hand remain. The dimensionality of the training data is massively reduced, which also makes the training less computationally expensive. As a side effect, all feature matrices are transformed to i-vectors of equal length, eliminating problems that are caused by varying utterance lengths.

Mathematically, this assumption can be expressed as:

$$M(u) = m + Tw \tag{2.2}$$

In this equation, $M(u)$ is the GMM supervector for utterance u. The supervector approach was first presented in [10] and has since been successfully applied to a number of speech recognition problems. A music example can be found in [11]. m represents the language- and channel-independent component of u and is estimated using a Universal Background Model (UBM). T is a low-rank matrix modeling the relevant language- and channel-related variability, the so-called Total Variability Matrix. Finally, w is a normally distributed latent variable vector: The i-vector for utterance u.

Step 1: UBM training A Universal Background Model (UBM) is trained using Gaussian Mixture Models (GMMs) from all utterances. This UBM models the characteristics that are common to all of them.

Step 2: Statistics extraction 0th and 1st order Baum-Welch statistics are calculated for each of the utterances from the UBM according to:

$$N_c(u) = \sum_{t=1}^{L} P(c|y_t, \Omega) \tag{2.3}$$

$$\widetilde{F}_c(u) = \sum_{t=1}^{L} P(c|y_t, \Omega)(y_t - m_c) \tag{2.4}$$

where $u = y_1, y_2, ..., y_L$ denotes an utterance with L frames, $c = 1, ..., C$ denotes the index of the Gaussian component, Ω denotes the UBM, m_c is the mean of the UBM mixture component c, and $P(c|y_t, \Omega)$ denotes the posterior probability that the frame y_t was generated by mixture component c. As the equation shows, the 1st order statistics are centered around the mean of each mixture component.

Step 3: T matrix training Using the Baum-Welch statistics for all utterances, the Total Variability Matrix T is now trained iteratively according to:

$$w = (I + T^t \Sigma^{-1} N(u) T)^{-1} T^t \Sigma^{-1} \widetilde{F}(u) \tag{2.5}$$

using Expectation Maximization.

Step 4: Actual i-vector extraction Finally, an i-vector w can be extracted for each utterance using equation 2.5 again. This can also be done for unseen utterances, using a previously trained T.

2.3.4 Artificial Neural Networks

2.3.4.1 Deep Neural Networks

2.3.4.2 Deep Belief Networks

2.4 Evaluation

2.4.1 Evaluation of phoneme recognition and alignment tasks

2.4.2 Evaluation of language identification tasks

2.4.3 Evaluation of keyword spotting tasks

2.5 Speech recognition systems

2.5.1 Phoneme recognition

2.5.2 Forced alignment

2.5.3 Language identification

Language identification has been extensively researched in the field of Automatic Speech Recognition since the 1980's. A number of successful algorithms have been developed over the years. An overview over the fundamental techniques is given by Zissman in [3].
Fundamentally, four properties of languages can be used to discriminate between them:

Phonetics The unique sounds that are used in a given language.

Phonotactics The probabilities of certain phonemes and phoneme sequences.

Prosody The "melody" of the spoken language.

Vocabulary The possible words made up by the phonemes and the probabilities of certain combinations of words.

Even modern system mostly focus on phonetics and phonotactics as the distinguishing factors between languages. Vocabulary is sometimes exploited in the shape of language models.
Zissman mentions Parallel Phone Recognition followed by Language Modeling (PPRLM) as one of the basic techniques. It requires audio data, language annotations, and phoneme annotations for each utterance. In order to make use of vocabulary characteristics, full sentence annotations and word-to-phoneme dictionaries are also necessary. Using the audio and phoneme data, acoustic models are trained. They describe the probabilities of certain sound and sound sequences occurring. This is done separately for each considered language. Similarly, language models are generated using the sentence annotations and the dictionary. These models describe the probabilities of certain words and phrases. Again, this is done for each language.
New audio examples are then run through all pairs of acoustic and language models, and the likelihoods produced by each model are retained. The highest acoustic likelihood, the highest language likelihood, or the highest combined likelihood are then considered to determine the language. This approach achieves up to 79% accuracy for ten languages [12].
Another approach uses the idea to train Gaussian Mixture Models for each language. This technique can be considered a "bag of frames" approach, i.e. the single data frames are considered to be statistically independent of each other. The generated GMMs then describe probability densities for certain characteristics of each language. Using these, the language of new audio examples can be easily determined.
GMM approaches used to perform worse than their PPRLM counterparts, but the development of new features has made the difference negligible [13]. They are in general easier to implement since only audio examples and their language annotations are required. Allen et al. [7] report results of up to 76.4% accuracy for ten languages. Different backend classifiers, such as Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) [6] have also been used successfully instead of GMMs.

2.5.4 Keyword spotting

3 State of the art

3.1 From speech to singing

Singing presents a number of challenges for speech recognition when compared to pure speech [14] [15] [16]. The following factors make speech recognition on singing more difficult than on speech, and make it necessary to adapt existing algorithms.

Larger pitch fluctuations A singing voice varies its pitch to a much higher degree than a speaking voice. It often also has very different spectral properties.

Larger changes in loudness In addition to pitch, loudness also fluctuates much more in singing than in speech.

Higher pronunciation variation Singers are often forced by the music to pronounce certain sounds and words differently than if they were speaking them.

Larger time variations In singing, sounds are often prolonged for a certain amount of time to fit them to the music. Conversely, they can also be shortened or left out completely.

In order to research this effect more closely, a small experiment was performed on a speech corpus and a singing corpus (*TIMIT* and *ACAP*, see chapter 4): The standard deviations for all the phonemes in each data set were calculated. The result is shown in figure 3.1, confirming that the variation in singing is much wider. This is particularly true for vowels.

Different vocabulary In musical lyrics, words and phrases often differ from normal conversation texts. Certain words and phrases have different probabilities (e.g. higher focus on emotional topics in singing).

Background music This is the biggest interfering factor when considering polyphonic recordings. Harmonic and percussive instruments add a huge amount of spectral

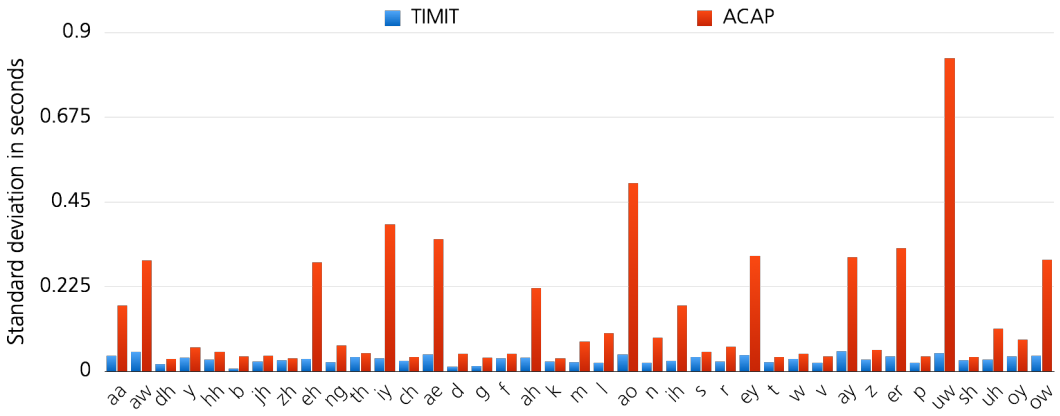


Figure 3.1: Standard deviations of phoneme durations in the *TIMIT* and *ACAP* data sets.

components to the signal, which lead to confusion in speech recognition algorithms. Ideally, these components should be removed or suppressed in a precursory step. This could be achieved, for example, by employing source separation algorithms. However, such algorithms add additional artifacts to the signal, and may not even be sufficient for this purpose at the current state of research.

Voice activity detection (VAD) could be used as a non-invasive first step to at the very least discard segments of songs that do not contain voice. However, such algorithms often make mistakes in the same cases that are problematic for speech recognition algorithms (e.g. instrumental solos)[17].

For these reasons, most of the experiments in this work were performed on unaccompanied singing. The integration of the mentioned pre-processing algorithms would be a very interesting next step of research.

The exception are the lyrics-to-singing alignment algorithms presented in Chapter 8. Those were also tested on polyphonic music, and the algorithms appear to be largely robust to these influences.

3.2 Phoneme recognition

Due to the factors mentioned above in section **??**, phoneme recognition on singing is more difficult than on clean speech. It has only been a topic of research for a few years, and there are few publications.

One of the earliest systems was presented by Wang et al. in 2003 [18]. Acoustic modeling is performed with triphone HMMs trained on read speech in Taiwanese and

Mandarin. The language model is completely restricted to lines of lyrics in the test dataset. Testing is performed on 925 unaccompanied sung phrases in these language. Due to the highly specific language model, the word error rate is just .07.

Hosoya et al. employ a similarly classical approach from ASR that employs monophone HMMs also trained on read speech for acoustic modeling (2005) [19]. These models are adapted to singing voices using the Maximum Likelihood Linear Regression (MLLR) technique [20]. Language modeling is performed with a Finite State Automaton (FSA) specific to the Japanese language, making it more flexible than the previous system.
The system is tested on five-word unaccompanied phrases, while the adaptation is performed on 127 choruses performed by different singers. The Word Error Rate is .36 without the adaptation, and .27 after adaptation.

In 2007, Gruhne et al. presented a classical approach that employs feature extraction and various machine learning algorithms to classify singing into 15 phoneme classes [21] [22]. The specialty of this approach lies in the pre-processing: At first, fundamental frequency estimation is performed on the audio input, using a Multi-Resolution Fast Fourier Transform [23]. Based on the estimated fundamental frequency, the harmonic partials are retrieved from the spectrogram. Then, a sinusoidal re-synthesis is performed, using only the detected fundamental frequency and partials. Feature extraction is then performed on this re-synthesis instead of the original audio. Extracted features include MFCCs, PLPs [24], LPCs, and WLPCs [25]. MLP, GMM, and SVM models are trained on the resulting feature vectors. The re-synthesis idea comes from a singer identification approach by Fujihara [26].
The approach is tested on more than 2000 separate, manually annotated phoneme instances from polyphonic recordings. Only one feature vector per phoneme instance is calculated. The phoneme set is reduced down to 15 classes. Using SVM models, 56% of the tested instances were classified correctly. This is significantly better than the best result without the re-synthesis step (34%).
In [27] (2010), the approach is expanded by testing a larger set of perceptually motivated features, and more classifiers. No significant improvements are found when using more intricate features, and the best-performing classifier remains SVM.

Fujihara et al. described an approach based on spectral analysis in 2009 [28]. The underlying idea is that spectra of polyphonic music can be viewed as the weighted sum

of two types of spectra: One for the singing voice, and one for the background music. This approach then models these two spectra as probabilistic spectral templates. The singing voice is modeled by multiplying a vocal envelope template, which represents the spectral structure of the singing voice, with a harmonic filter, which represents the harmonic structure of the produced sound itself. This is analogous to the source-filter model of speech production [29]. For recognizing vowels, five such harmonic filters are prepared (a - e - i - o - u). Vocal envelope templates are trained on voice-only recordings, separated by gender. Templates for background music are trained on instrumental tracks. In order to recognize vowels, the probabilities for each of the five harmonic templates are estimated. As a side product, the algorithm also estimates the fundamental frequency of the singing voice.
As described, the phoneme models are gender-specific and only model five vowels, but also work for singing with instrumental accompaniment. The approach is tested on 10 Japanese-language songs. The best result is 65% correctly classified frames, compared to the 56% with the previous approach by this team, based on GMMs.

Also in 2009, Mesaros et al. picked Hosoya's approach back up by using MFCC features and GMM-HMMs for acoustic modeling [30] and adapting the models for singing voices. These models are trained on the CMU ARCTIC speech corpus [1]. Then, different Maximum Likelihood Linear Regression (MLLR) techniques for adapting the models to singing voices are tested [20].
The adaptation and test corpus consists of 49 voice-only fragments from 12 pop songs with durations between 20 and 30 seconds. The best results are achieved when both the means and variances of the Gaussians are transformed with MLLR. The results improved slightly when not just a single transform was used for all phonemes, but when they were grouped into base classes beforehand, each receiving individual transformation parameters. The best result is around .79 Phoneme Error Rate on the test set.
In [31] and [32], language modeling is added to the presented approach. Phoneme-level language models are trained on the CMU ARCTIC corpus as unigrams, bigrams, and trigrams, while word-level bigram and trigram models are trained on actual song lyrics in order to match the application case. The output from the acoustic models is then refined using these language models. The approach is tested on the clean singing corpus mentioned above, and on 100 manually selected fragments of 17 polyphonic pop songs. To facilitate recognition on polyphonic music, a vocal separation algorithm is

[1] `http://festvox.org/cmu_arctic/`

introduced [33].
Using phoneme-level language modeling, the phoneme error rate on clean singing is reduced to .7. On polyphonic music, it is .81. For the word recognition approach, the word error rate is .88 on clean singing, and .94 on the polyphonic tracks.
A more detailed voice adaptation strategy is tested in [34]. Instead of adapting the acoustic models with mixed singing data, they are adapted gender-wise, or to specific singers. With the gender-specific adaptations, the average phoneme error rate on clean singing is lowered to .81 without language modeling, and .67 with language modeling. Singer-specific adaptation does not improve the results, probably because of the very small amount of adaptation data in this case.
In [35] (2014), McVicar et al. build on a very similar baseline system, but also exploit repetitions of choruses to improve transcription accuracy. This has been done for other MIR tasks, such as chord recognition, beat tracking, and source separation. They propose three different strategies for combining individual results: Feature averaging, selection of the chorus instance with the highest likelihood, and combination using the Recogniser Output Voting Error Reduction (ROVER) algorithm [36]. They also employ three different language models, two of which were matched to the test songs (and therefore not representative for general application). 20 unaccompanied, English-language songs from the RWC database [37] were used for testing; chorus sections were selected manually. The best-instance selection and the ROVER strategies improve results significantly; with the ROVER approach and a general-purpose language model, the Phoneme Error Rate is at .74 (versus .76 in the baseline experiment), while the Word Error Rate is improved from .97 to .9. Interestingly, cases with a low baseline result benefit the most from exploiting repetition information.

The final system was proposed by Hansen in 2012 [38]. It also employs a classical approach consisting of a feature extraction step and a model training step. Extracted features are MFCCs and TRAP (TempoRAl Pattern) features. Then, Multilayer Perceptrons (MLPs) are trained separately on both feature sets. The assumption is that each feature models different properties of the considered phonemes: Short-term MFCCs are good at modeling the pitch-independent properties of stationary sounds, such as sonorants and fricatives. On the flip-side, TRAP features are able to model temporal developments in the spectrum, forming better representations for sounds like plosives or affricates.
The results of both MLP classifiers are combined via a fusion classifier, also an MLP. Then, Viterbi decoding is performed on its output.

The approach is trained and tested on a data set of 12 vocal tracks of pop songs, which were manually annotated with a set of 27 phonemes. The combined system achieves a recall of .48, compared to .45 and .42 for the individual MFCC and TRAP classifiers respectively. This confirms the assumption that the two features complement each other. The phoneme-wise results further corroborate this.

3.3 Lyrics-to-audio alignment

In contrast to the other tasks discussed in this chapter, the task of lyrics-to-audio alignment has been the focus of many more publications. A comprehensive overview until 2012 is given in [15].
A first approach was presented in 1999 by Loscos et al. [39]. The standard forced alignment approach from speech recognition is adapted for singing. MFCCs are extracted first, and then a left-to-right HMM is employed to perform alignment via Viterbi decoding. Some modifications are made to the Viterbi algorithm to allow for low-delay alignment. The approach is trained and tested on a very small (22 minutes) database of unaccompanied singing, but no quantitative results are given.
The first attempt to synchronize lyrics to polyphonic recordings was made by Wang et al. in 2004 [40]. They propose a system, named "LyricAlly", to provide line-level alignments for karaoke applications. Their approach is heavily based on musical structure analysis. First, the hierarchical rhythm structure of the song is estimated. The result is combined with an analysis of the chords and then used to split the song into sections, based on a chorus detection algorithm. Second, Vocal Activity Detection (VAD) using HMMs is performed on each section. Then, sections of the text lyrics are assigned to the detected sections (e.g. verses, choruses). In the next step, the algorithm determines whether the individual lines of the lyrics match up with the vocal sections detected by the VAD step. If they do not, grouping or partitioning is performed. This is based on the assumption that lyrics match up to rhythmic bars as determined by the hierarchical rhythm analysis. The expected duration of each section and line is estimated using a Gaussian distributions of phoneme durations from a singing data set. In this manner, lines of text are aligned to the detected vocal segments. The approach is tested on 20 manually annotated pop songs. On the line level, the average error is .58 seconds for the starting points and $-.48$ seconds for the durations. The system components are analyzed in more detail in [41].
In 2006, the same team also presented an approach that also performs rhythm and bar analysis to facilitate syllable-level alignment [42]. For the phoneme recognition

step, an acoustic model is trained on speech data and adapted to singing using the previously mentioned 20 songs. The possible syllable positions in the alignment step are constrained to the note segments detected in the rhythm analysis step. Due to annotator disagreement on the syllable level, the evaluation is performed on the word level. On three example songs, the average synchronization error rate is .19 when allowing for a tolerance of 1/4 bar.

An approach which does not require in-depth music analysis was presented by Fujihara et al. in 2006 [43]. A straightforward Viterbi alignment method from speech recognition is refined by introducing three singing-specific pre-processing steps: Accompaniment sound reduction, Vocal Activity Detection, and phoneme model adaptation. For accompaniment reduction, the previously mentioned harmonic re-synthesis algorithm from [26] is used again. For Vocal Activity Detection, a HMM is trained on a small set of unaccompanied singing using LPC-derived MFCCs and F_0 differences as features. The HMM can be parameterized to control the rejection rate. For the phoneme model adaptation, three consecutive steps are tested: Adaptation to a clean singing voice, adaptation to a singing voice segregated with the accompaniment reduction method, and on-the-fly adaptation to a specific singer. MFCC features are used for the Viterbi alignment, which is performed on the vowels and syllabic nasals ([m],[n],[l]) only.
Ten Japanese pop songs were used for testing. Evaluation was done one the phrase level by calculating the proportion of the duration of correctly aligned sections to the total duration of the song. For eight of the ten songs, this proportion was .9 or higher when using the complete system, which the authors judge as satisfactory. Generally, the results are lower for performances by female singers, possibly because of the higher F_0s. These performances also benefit the most from the Vocal Activity Detection step, even though this also seems to perform a bit worse for female singing. All three levels of phoneme model adaptation contribute to the success of the system.
In 2008, the authors improved upon this system with three modifications: Fricative detection, filler models, and new features for the Vocal Activity Detection step [44]. Fricative detection is introduced because the previous system was only based on vowels and nasals, due to the fact that the harmonic re-synthesis discards other consonants. In the new system, fricatives are detected before this step and then retained for the alignment (stops are not used because they are too short).
The filler model is employed because singers sometimes add extraneous lyrics (like “la la la” or “yeah”) to their performances.

As mentioned above, Vocal Activity Detection does not work as well for female performances because of inaccuracies of the spectral envelope estimation in high pitch regions. For this reason, the features are replaced in the new version by comparing the power of the harmonic components directly with those of similar F_0 regions.
The approach is again evaluated on ten Japanese pop songs. The original system produces an average accuracy of .81, which is raised to .85 with the new improvements.

Mauch et al. augmented the same approach in 2010 by using chord labels, which are often available together with the lyrics on the internet [45] [46]. Chords usually have longer durations than individual phonemes, and are therefore easier to detect. In this way, they provide a coarse alignment, which can be used to simplify the shorter-scale phoneme-level alignment. A chroma-based approach is used to estimate the chords. Information about the chord alignments is directly integrated into the HMM used for alignment.
In [46], a large range of parameterizations is tested on 20 English-language pop songs. The highest accuracy for the baseline approach (without chord information) is .46. Using chord position information, this rises to .88. Interestingly, Vocal Activity Detection improves the result when not using chords, but decreases it for the version with chord alignments, possibly because the coarse segmentation it provides is already covered by the chord detection in the second case. The method is also able to cope with incomplete chord transcriptions while still producing satisfactory results.

In 2007, Wong et al. presented a specialized approach for Cantonese singing that does not require phoneme recognition [47]. Since Cantonese is a tonal language, prosodic information can be inferred from the lyrics. This is done by estimating relative pitch and timing of the syllables by using linguistic rules. On the other side, a vocal enhancement algorithm is applied to the input signal, and its pitches and onsets are calculated. Then, both sets of features are aligned using Dynamic Time Warping.
14 polyphonic songs were used for evaluation. The approach reaches an average (duration) accuracy of .75.

Lee et al. also follow an approach without phoneme recognition (2008) [48]. It is purely based on structural analysis of the song recording, which is done by calculating a self-similarity matrix and using it for segmentation. The algorithm takes structural a-priori knowledge into account, e.g. the fact that choruses usually occur most frequently and do not differ very much internally. Lyrics segments are annotated by hand,

splitting them up into paragraphs and labeling them with structural parts (“ï¿½ntro”, “verse”, “chorus”, and “bridge”). Then, Dynamic Programming is performed to match the lyrics paragraphs to the detected musical segments. A Vocal Activity Detection step is also introduced. Testing the approach on 15 English-language pop songs with 174 lyrics paragraphs in total, they obtain an average displacement error of 3.5 seconds.

Mesaros et al. also present an alignment approach that makes use of their phoneme recognition approach described above in **??** [49], but also using a harmonic re-synthesis step for vocal separation. Based on these models, they employ straightforward Viterbi alignment and obtain an average displacement error of 1.4 seconds for line-wise alignment (.12 seconds when not using absolute values). The test set consists of 17 English-language pop songs. They identify mistakes in the vocal separation step as the main source of error.

Two specialized approaches were presented in the past two years. The first one is part of a score-following algorithm by Gong et al. [50]. Here, vowels are used to aid alignment of the musical score to the recording, assuming the score contains the lyrics. This is done by training vowel templates on a small set of sung vowels. Spectral envelopes are used as features. Two different strategies for fusing the vowel and melody information are tested (“early” and “late”), as well as singer-specific adaptation of the templates via MAP (Maximum A-Posteriori). The training set consists of 160 vowel instances per singer, the test set of 8 full unaccompanied French-language songs per singer. The average displacement error is around .68 for both singer-specific and -adapted models (best strategy).

Finally, Dzhambazov et al. presented a method that integrates knowledge of note onsets into the alignment algorithm [51]. Pitch extraction and note segmentation are performed in parallel with phoneme recognition via HMMs, and both results are refined with a transition model. A variable time HMM (VTHMM) is used to model the rules for phoneme transitions at note onsets.
On a test dataset of 12 unaccompanied Turkish-language Makam performances, the method achieves an alignment accuracy of .76. For polyphonic recordings (usually with accompaniment by one or more string instruments), a vocal resynthesis step is introduced. The average accuracy in this case is .65.

3.4 Lyrics retrieval

As described in 3.2, Hosoya et al. developed a system for phoneme recognition, which they also apply to lyrics retrieval [19]. On a dataset of 238 children's songs, they obtain a recognition rate of .86 for the Top 1 result, and of .91 for the Top 10 results. In [52] and [53], more experiments are conducted. As a starting point, the number of words in the queries is fixed at 5, resulting in a recognition rate of .9 (Top 1 result). Then, a melody recognition is used to verify the matches proposed by the speech reognition step, raising the recognition rate to .93. The influence of the number of words in the query is also evaluated, confirming that retrieval becomes easier the longer the query is. However, even at a length of just three words, the recognition rate is .87 (vs. .81 without melody verification).

Similarly, Wang et al. presented a query-by-singing system in 2010 [54]. The difference here is that melody and lyrics information are weighted equally in the distance calculation. Lyrics are recognized here with a bigram HMM model trained on speech. The results are interpreted as syllables. A syllable similarity matrix is employed for calculating phoneme variety in the query, which is used for singing vs. humming discrimination. Assuming that only the beginning of each song is used as the starting point for queries, the first 30 syllables of each song are transformed into an FSM language model and used for scoring queries against each song in the database. The algorithm is tested on a database of 2154 Mandarin-language songs, of which 23 were annotated and the remainder are used as "noise" songs. For the Top 1 result, a recognition rate of .91 is achieved for the system combining melody and lyrics information, compared to .88 for the melody-only system.

As described in 3.2, Mesaros et al. developed a sophisticated system for phoneme and word recognition in singing. In [55], [56], and [32], they also describe how this system can be used for lyrics retrieval. This is the only purely lyrics-based system in literature. Retrieval is performed by recognizing words in queries with the full system, including language modeling, and then ranking each lyrics segment by the number of matching words (bag-of-words approach). The lyrics database is constructed from 149 song segments (between 9 and 40 seconds in the corresponding recordings). Out of these segments, recordings of 49 are used as queries to test the system. The Top 1 recognition rate is .57 (.71 for the Top 10).

3.5 Language identification

A first approach for language identification in singing was proposed by Tsai and Wang in 2004 [57]. At its core, the algorithm is similar to Parallel Phoneme Recognition (PPR). However, instead of full phoneme modeling, they employ an unsupervised clustering algorithm to the input feature data and tokenize the results to form language-specific codebooks (plus one for background music). Following this, the results from each codebook are run through a matching language models to determine the likelihood that the segment was performed in this language. Prior to the whole process, vocal/non-vocal segmentation is performed. This is done by training GMMs on segments of each language, and on non-vocal segments. MFCCs are used as features.
The approach is tested on 112 English- and Mandarin-language polyphonic songs each, with 32 songs performed in both languages. A classification accuracy of .8 is achieved on the non-overlapping songs. On the overlapping songs, it is only at .7, suggesting some influence of the musical material (as opposed to the actual language characteristics). Misclassifications occur more frequently on the English-language songs, possibly because of accents of Chinese singers performing in English, and because of louder background music.

A second, simpler approach was presented by Schwenninger et al. in 2006 [58]. They also extract MFCC features, and then use these to directly train statistical models for each language. Three different pre-processing strategies are also tested: Automatic vocal/non-vocal segmentation, distortion reduction, and azimuth discrimination. Vocal/non-vocal segmentation is performed by thresholding the energy in high-frequency bands as an indicator for voice presence over 1 second windows. This leaves a relatively small amount of material per song. Distortion reduction is employed to discard strong drum and bass frames where the vocal spectrum is masked by using a mel-based approach. Finally, azimuth discrimination attempts to detect and isolate singing voice panned to the center of the stereo scene.
The approach is tested on three small data sets of speech, unaccompanied singing, and polyphonic music. Without pre-processing steps, the accuracy is .84, .68, and .64 respectively, highlighting the increased difficulty of language identification on singing versus speech, and on polyphonic music versus pure vocals. On the polyphonic corpus, the pre-processing steps do not improve the result.

In 2011, Mehrabani and Hansen presented a full Parallel Phoneme Recognition fol-

lowed by Language Modeling (PPRLM) approach for singing language identification. MFCC features are run through phoneme recognizers for Hindi, German, and Mandarin; then, the results are scored by individual language models for each considered language. In addition, a second system is employed which uses prosodic instead of phonetic tokenization. This is done by modeling pitch contours with Legendre polynomials, and then quantizing these vectors with previously trained GMMs. The results are then again used as inputs to language models.
The approach is trained and tested on a corpus containing 12 hours of unaccompanied singing and speech in Mandarin, Hindi, and Farsi. The average accuracy for singing is .78 and .43 for the phoneme- and prosody-based systems respectively, and .83 for a combination of both.

Also in 2011, Chandrasekhar et al. presented a very interesting approach for language identification on music videos, taking into account both audio and video features [59]. On the audio side, the spectrogram, volume, MFCCs, and perceptually motivated Stabilized Auditory Images (SAI) are used as inputs. One-vs-all SVMs are trained for each language. The approach is trained and tested on 25,000 music videos, taking 25 languages into consideration. Using audio features only, the accuracy is .45; combined with video features, it rises to .48. It is interesting to note that European languages seem to achieve much lower accuracies than Asian and Arabic ones. English, French, German, Spanish and Italian rank below .4, while languages like Nepali, Arabic, and Pashto achieve accuracies above .6. It is possible that the language characteristics of European languages make them harder to discriminate (especially against each other) than others.

3.6 Keyword spotting

Keyword spotting in singing was first attempted in 2008 by Fujihara et al. [60]. Their approach employs a phoneme recognition step first, which is again based on the vocal re-synthesis method first described in [26]. MFCCs and power features are extracted from the re-synthesized singing and used as inputs to a phoneme model, similar to Gruhne's phoneme recognition approach mentioned above in **??**. Three phoneme models are compared: One trained on pure speech and adapted with a small set of singing recordings, one adapted with all recordings, and one trained directly on singing. Viterbi decoding is then performed using keyword-filler HMMs (see **??**) to detect candidate segments where keywords may occur. These segments are then re-scored through the

filler HMM to verify the occurrence.
The method is tested on 79 unaccompanied Japanese-language songs from the RWC database [37]. The Phoneme Error Rate is .73 for the acoustic models trained on speech, .67 for the adapted models, and .49 for the models trained on singing (it should be mentioned that the same songs were used for training and testing, although a cross-validation experiment appears to show that the effect is negligible). The employed evaluation measure is "link success rate", describing the percentage of detected phrases that were linked correctly to other occurrences of the phrase in the data set. In that sense, it is a sort of accuracy measure. The link success rate for detecting the keywords is .3. The authors show that the result depends highly on the number of phonemes in the considered keyword, with longer keywords being easier to detect.

In 2012, Mercado et al. presented an approach to keyword spotting in singing based on a different principle: Dynamic Time Warping (DTW) between a sung query and the requested phrase in the song recording. In particular, Statistical Sub-sequence DTW is the algorithm employed for this purpose. MFCCs are used as feature inputs, then the costs of the warping paths are calculated from all possible starting points to obtain candidate segments, which are then further refined to find the most likely position.
The approach is tested on a set of vocal tracks of 19 pop songs (see Section **??**) as the references, and phrase recordings by amateur singers as the queries, but no quantitative results are given. The disadvantage of this approach lies in the necessity for audio recordings of the key phrases, which need to have at least similar timing and pitch as the reference phrases.

Finally, Dzhambazov et al. developed a score-aided approach to keyword spotting in 2015 [61]. A user needs to select a keyword phrase and a single recording in which this phrase occurs. The keyword is then modeled acoustically by concatenating recordings of the constituent phonemes (so-called acoustic keyword spotting). Similar to Mercado's approach, Sub-Sequence DTW is then performed between the acoustic template and all starting positions in the reference recording to obtain candidate segments. These segments are then refined by aligning the phonemes to the score in these positions to model their durations. This is done by using Dynamic Bayesian Network HMMs. Then, Viterbi decoding is performed to re-score the candidate segments and obtain the best match.
The approach is tested on a small set of unaccompanied Turkish-language recordings of traditional Makam music. The Mean Average Precision (MAP) for the best match

is .08 for the DTW approach only, and .05 for the combined approach. For the top-6 results, the MAP is .26 and .38 respectively.

4 Data sets

This chapter contains descriptions of all the data sets (or corpora) used over the course of this thesis. They are grouped into speech-only data sets, data sets of unaccompanied (=a-capella) singing, and data sets of full musical pieces with singing ("real-world" data sets).

4.1 Speech data sets

4.1.1 TIMIT

TIMIT is, presumably, the most widely used corpus in speech recognition research [62]. It was developed in 1993 and consists of 6300 English-language audio recordings of 630 native speakers with annotations on the phoneme, word, and sentence levels. The corpus is split into a training and a test section, with the training section containing 4620 utterances, and the test section containing 1680. Each of those utterances has a duration of a few seconds. The recordings are sampled at $16,000Hz$ and are mono channel.
The phoneme annotations follow a model similar to ARPABET and contain 61 different phonemes [63].

4.1.2 NIST Language identification corpora

The National Institute for Standards and Technology (NIST) regularly runs various speech recognition challenges, one of them being the Language Recognition Evaluation (LRE) task, which is offered every two to four years [1]. To this end, they publish training and evaluation corpora of speech in several languages. They consist of short segments (up to 35 seconds) of free telephone speech by many different speakers. The recordings are mono channel with a sampling rate of $8000Hz$.
The corpus for the 2003 challenge was used in this work for comparison of language

[1] `https://www.nist.gov/itl/iad/mig/language-recognition`

identification algorithms against speech data [64]. Only the English-, German-, and Spanish-language subsets were chosen, since those languages are covered by the corresponding singing dataset. To balance out the languages, 240 recordings were used for each of them, making up around 1 hour of material.

4.1.3 OGI Multi-Language Telephone Speech Corpus

In 1992, the Oregon Institute for for Science and Technology (OGI) also published a multilingual corpus of telephone recordings, called the OGI Multi-Language Telephone Speech Corpus, to facilitate multi-language ASR research. Just like the NIST corpora, it has become widely used for speech recognition tasks. Again, the English-, German-, and Spanish-language subsets were used in this work. They each consist of more than 1000 recordings per language of up to 50 seconds duration, making up a total of about three to five hours. In contrast to the NIST corpus, the recordings are somewhat less "clean" with regards to accents and background noise; the sampling rate is also $8000Hz$ on mono channel.
In many experiments, languages were balanced out by randomly discarding utterances to obtain equal numbers. For experiments on longer recordings, results on individual utterances were aggregated for each speaker and also balanced down to the lowest number, producing 118 documents per language (354 in sum).

4.2 Singing data sets

4.2.1 YouTube data set

As opposed to the speech case, there are no standardized corpora for sung language identification. For the sung language identification experiments, files of unaccompanied singing were therefore extracted from *YouTube*[2] videos. This was done for three languages: English, German, and Spanish. The corpus consists of between 116 (258min) and 196 (480min) examples per language. These were mostly videos of amateur singers freely performing songs without accompaniment. Therefore, they are of highly varying quality and often contain background noise. The audio was downloaded in the natively provided quality, but then downsampled to $8000Hz$ and averaged to a mono channel for uniformity and for compatibility with the speech corpora for language identifica-

[2] `http://www.youtube.com`, Last checked: 05/16/13

Table 4.1: Amounts of data in the three used data sets: Sum duration on top, number of utterances in italics.

hh:mm:ss ***#Utterances*** ***#Speakers*** **∅ Duration/Speaker**	**NIST2003LRE**	**OGIMultilang**	**YTAcap**
English	00:59:08 *240* - -	05:13:17 *1912* 200 00:01:34	08:04:25 *1975* 196 00:02:28
German	00:59:35 *240* - -	02:52:27 *1059* 118 00:01:28	04:18:57 *1052* 116 00:02:14
Spanish	00:59:44 *240* - -	03:05:45 *1151* 129 00:01:26	07:21:55 *1810* 185 00:02:43

tion. Most of the performers contributed only a single song, with just a few providing up to three. This was done to avoid effects where the classifier recognizes the singer's voice instead of the language.

Special attention was paid to musical style. Rap, opera singing, and other specific singing styles were excluded. All the songs performed in these videos were pop songs. Different musical styles can have a high impact on language classification results. The author tried to limit this influence as much as possible by choosing recordings of pop music instead of language-specific genres (such as Latin American music).

For many experiments, they were split up into segments of 10-20 seconds at silent points (3,156 "utterances" in sum). In most cases, these segments were then balanced for the languages by randomly discarding superfluous segments. In other experiments, a balanced set of the whole songs was used (i.e. 116 songs per language).

An overview of the amounts of data in the three corpora for language identification is given in Table 4.1.

4.2.2 Hansen's vocal track data set

This is one of the data sets used for keyword spotting and phoneme recognition. It was first presented in [38]. It consists of the vocal tracks of 19 commercial English-language pop songs. They are studio quality with some post-processing applied (EQ, compression, reverb). Some of them contain choir singing. These 19 songs are split up into 920 clips that roughly represent lines in the song lyrics. The original audio quality is $44,100Hz$ on mono channels; for compatibility with models trained on *TIMIT*, they were downsampled to $16,000Hz$.
Twelve of the songs were annotated with time-aligned phonemes. The phoneme set is the one used in CMU Sphinx[3] and TIMIT [62] and contains 39 phonemes. All of the songs were annotated with word-level transcriptions. This is the only one of the singing data sets that has full manual annotations, which are assumed to be reliable and can be used as ground truth.
For comparison, recordings of spoken recitations of all song lyrics were also made. These were all performed by the same speaker (the author).
This data set will be referred to as *ACAP*.

4.2.3 DAMP data set

As described, Hansen's data set is very small and therefore not suited to training phoneme models for singing. As a much larger source of unaccompanied singing, the *DAMP* data set, which is freely available from Stanford University[4][65], was employed. This data set contains more than 34,000 recordings of amateur singing of full songs with no background music, which were obtained from the *Smule Sing!* karaoke app. Each performance is labeled with metadata such as the gender of the singer, the region of origin, the song title, etc. The singers performed 301 English-language pop songs. The recordings have good sound quality with little background noise, but come from a lot of different recording conditions. They were originally provided in OGG format at a sampling rate of $22,050Hz$ (mono channel), but were also converted to wave format and downsampled to $16,000Hz$ for compatibility.
No lyrics annotations are available for this data set, but the textual lyrics can be obtained from the *Smule Sing!* website[5]. These are, however, not aligned in any way. Such an alignment was performed automatically on the word and phoneme levels (see

[3] `http://cmusphinx.sourceforge.net/`
[4] `https://ccrma.stanford.edu/damp/`
[5] `http://www.smule.com/songs`

section 5.3).
The selection of songs is not balanced, with performances ranging between 21 and 2038 instances. Female performances are also much more frequent than male ones, and gender often plays a role when training and evaluating models. For these reasons, several subsets of the dataset were composed by hand:

DampB Contains 20 full recordings per song (6000 in sum), both male and female.

DampBB Same as before, but phoneme instances were discarded until they were balanced and a maximum of 250,000 frames per phoneme where left, where possible. This data set is about 4% the size of *DampB*.

DampBB_small Same as before, but phoneme instances were discarded until they were balanced and 60,000 frames per phoneme were left (a bit fewer than the amount contained in *TIMIT*). This data set is about half the size of *DampBB*.

DampF and DampM Each of these data sets contains 20 full recordings per song performed by singers of one gender, female and male respectively (6000 each).

DampFB and DampMB Starting from *DampF* and *DampM*, these data sets were then reduced in the same way as *DampBB*. *DampFB* is roughly the same size, *DampMB* is a bit smaller because there are fewer male recordings.

DampTestF and DampTestM Contains one full recording per song and gender (300 each). These data sets were used for testing. There is no overlap with any of the training data sets.

4.2.4 Structure of the final corpus

Training data sets In order to generate training data sets, we decided to balance the data by songs so that certain songs (and their phoneme distributions) would not be overrepresented. The least represented song in the original data set has 20 recordings. We therefore chose 20 to 23 recordings per song at random to generate a training data set, which we named **DampB**. For each song, as many different singers as possible were selected. Additionally, recordings with more "Loves" (user approval) were more likely to be selected (however, a large percentage of the original data set did not have such ratings). This resulted in a data set of 6,902 recordings.

We then repeated this process, but only took recordings by singers of one gender

Gender	Training	Test
Both	DampB (6,902)	
Female	DampF (6,654)	DampTestF (301)
Male	DampM (4,534)	DampTestM (300)

Table 4.2: Overview of the structure of the *DAMP*-based phonetically annotated corpora, number of recordings in brackets.

into account. In this way, we created data sets of recordings by female and male singers only, which we named **DampF** and **DampM** respectively. Due to the gender split, there were fewer recordings of some of the songs available. Male singers in particular are underrepresented in the original data set. Therefore, **DampF** contains 6,564 recordings, while **DampM** contains 4,534. These sizes are roughly in the same range as for the mixed-gender data set, which enables the comparison of models trained on all three.

Test data sets For testing new algorithms, we also created two small test data sets from the original *DAMP* data set. These are, again, split by gender, and contain one recording per song, resulting in 301 recordings for the female data set and 300 for the male one (since there was one song with not enough available male recordings). We call them **DampTestF** and **DampTestM** respectively.
For mixed-gender training, we suggest simply performing the testing on both test data sets.

A structured overview is given in table 4.2.

4.2.5 QMUL Expletive data set

This data set consists of 80 popular full songs which were collected at Queen Mary University, most of them Hip Hop. 711 instances of 48 expletives were annotated on these songs. In addition, the matching textual, unaligned lyrics were retrieved from the internet. The audio is provided at $44,100kHz$ sampling rate in stereo, but was also resampled to $16,000Hz$ and a mono track for compatibility with some models.

5 Singing phoneme recognition

5.1 Phoneme recognition using models trained on speech

As a starting point, models for phoneme recognition were trained on speech data, specifically the *TIMIT* corpus. As in many classical ASR approaches, HMMs were selected as a basis and trained using the Hidden Markov Toolkit (HTK)[66].
Additionally, Deep Neural Networks (DNN) were trained for comparison with models trained on the other data sets. They had three hidden layers of 1024 nodes, 850 nodes, and 1024 nodes again.
In both cases, 13 MFCCs were extracted, and deltas and double-deltas were calculated, resulting in a feature vector of dimension 39. As the output, a set of 39 monophones based on the one used in CMU Sphinx[1] and in the *TIMIT* annotations was used. The set is listed with examples in **??**.
The HMMs were used for aligning text lyrics to audio in some of the following approaches. To verify the quality of such an alignment, this was tested on the part of the *ACAP* singing corpus that has phoneme annotations. On average, the alignment error was 0.16 seconds. A small manual check suggests that this value may be in the range of the agreement of annotators. A closer inspection of the results also shows that the biggest contribution to this error occur because a few segments were heavily misaligned, whereas most of them are just slightly off.
The DNN models were trained to perform phoneme recognition. This system was evaluated by first generating phoneme posteriorgrams from the test audio using the DNN models, and then running Viterbi decoding on those to extract phoneme strings. Then, the Phoneme Error Rate and the Weighted Phoneme Error Rate were calculated as described in section **??**.
The results for DNN models trained on *TIMIT* are shown in figure 5.1. For validation, the model was tested on the test part of *TIMIT*, resulting in a Phoneme Error Rate of

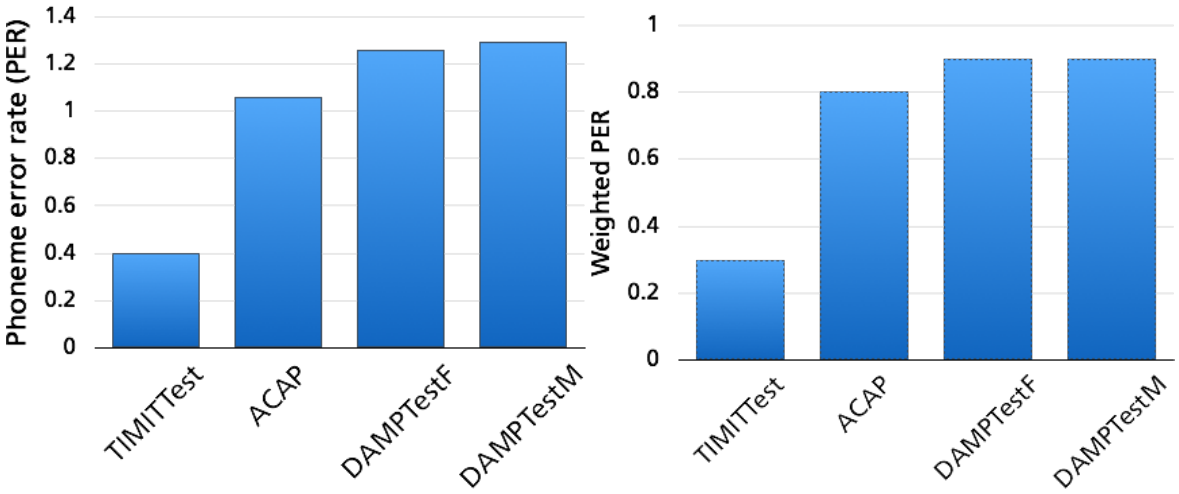
[1] `http://cmusphinx.sourceforge.net/`

.4 and a Weighted Phoneme Error Rate of .3. Higher values on these corpora can be found in literature, but those systems are usually more sophisticated and include language modeling steps and gender- or speaker-adapted models. In this scenario, those values serve as a validation of the recognition ability of the model on unseen data, and as an upper bound for the other results.
On the *ACAP* data set, the phoneme error rate is 1.06 and the weighted phoneme error rate is .8. Performing the same evaluation on the *DAMP* test data sets generates similar results: A phoneme error rate of 1.26/1.29, and a weighted phoneme error rate of .9. This demonstrates that the performance of models trained on speech leaves room for improvement when used for phoneme recognition in singing. The difference between both singing data sets can be explained by their content: *ACAP* is much smaller, and contains cleaner singing, both considering the recording quality and the singing performance. Songs in this data set were performed by professional singers, whereas the *DAMPTest* sets contain recordings by amateurs who may not always enunciate as clearly. Additionally, the phoneme annotations for the *DAMPTest* sets were generated from the song lyrics; these do not correspond to the actual performed phonemes in all cases.
It can be assumed that models trained on better-matching conditions (i.e. singing) would perform much better at this task. The problem with this approach lies in the lack of data sets that can be used for these purposes. In contrast with speech, no large corpora of phonetically annotated singing are available. In the following sections, workarounds for this problem are tested.

5.2 Phoneme recognition using models trained on "songified" speech

When there is a scarcity of suitable training data, attempts are often made to generate such data artificially from existing data for other conditions. For example, this is often done when models for noisy speech are required [67][68]. Inspired by this, one idea was making existing speech data sets more "song-like" and use these modified datasets to train models for phoneme recognition in singing. The *TIMIT* corpus was once again used as a basis for this.

An overview of the approach is shown in figure 5.2. Five variants of the training part *TIMIT* were generated first. MFCC features were then extracted from these new



(a) Phoneme error rate (b) Weighted phoneme error rate

Figure 5.1: Mean phoneme recognition results on the test data sets using acoustic models trained on *TIMIT*.

datasets and used to train models.
Similarly, MFCCs are extracted from the *TIMIT* Test set and from the *ACAP* data set. The 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.

In order to make the training data more "song-like", several variants of this dataset were developed. Table 5.1 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, the phase vocoder from [69], which is an implementation of the Flanagan/Dolson phase vocoder [70][71], was used. 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 section **??**, time variations in singing are mainly performed on vowels and are often much longer than in speech. Therefore, the *TIMIT* annotations were used to only pick out the vowel segments from the utterances.

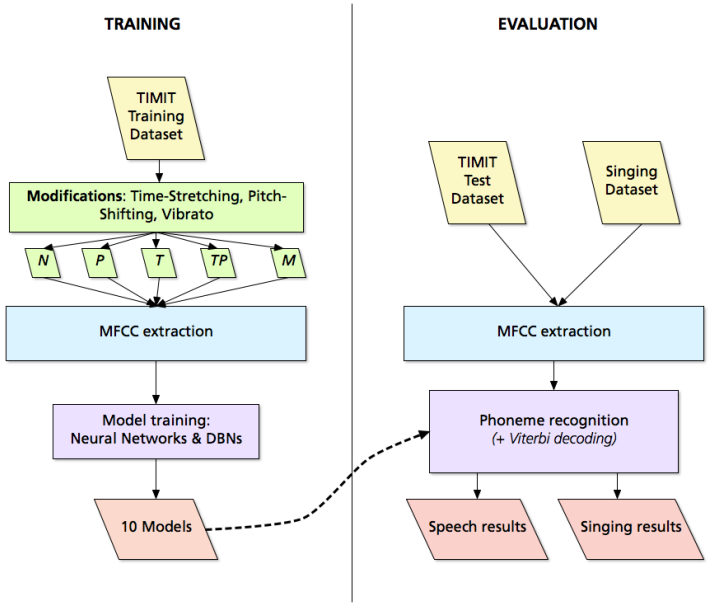


Figure 5.2: Overview of the "songified" phoneme recognition system

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, code from the freely available Matlab tool *AutoTune Toy*[72], which also implements a phase vocoder, was used. 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, utterances were split up into individual words, and then a pitch-shifted version of each word was generated and the results were concatenated. Pitches are randomly selected from a range between 60% and 120% of the original pitch.

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. A sine of amplitude 0.2 and frequency $6Hz$ was used.

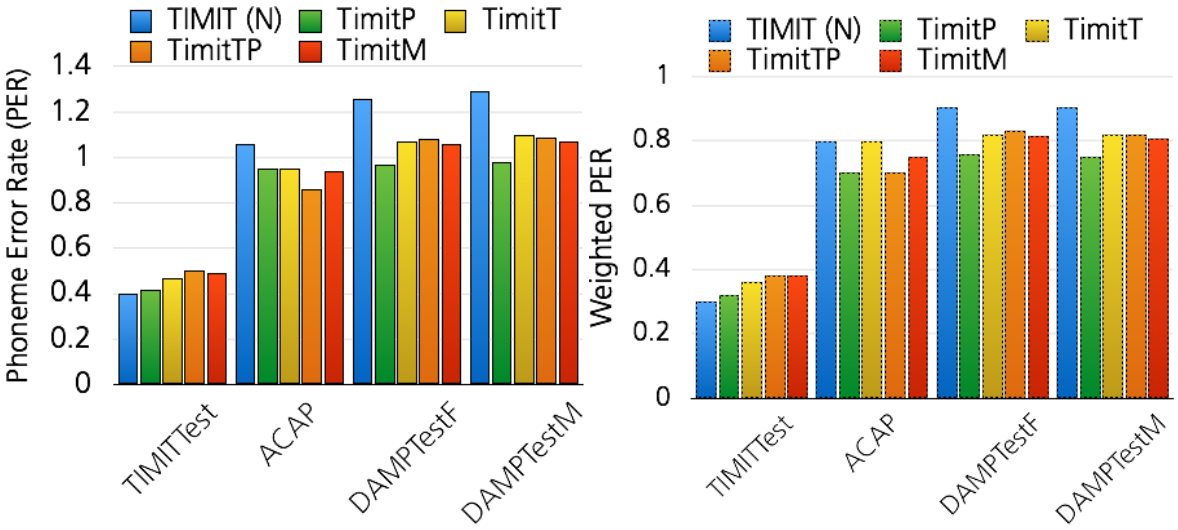
In singing, vibrato is commonly done on long sounds, which are usually vow-

	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 5.1: 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

els. Since spoken vowels are usually very short, vibrato cannot be perceived on them very well. Therefore, vibrato was only added when time stretching was also applied. Vibrato was then added to the extracted and stretched vowels.

The approach was tested on the various test data sets - namely, the test part of *TIMIT*, the *TIMIT* data set, and the two test sets chosen from the *DAMP* data set. Figure 5.3 shows the results for the DNN models. As it demonstrates, results for singing are generally worse than for speech. The base result for singing is a Weighted Phoneme Error Rate of .8 for *ACAP*, and of .9 for both *DAMPTestF* and *DAMPTestM* (model trained on the original TIMIT dataset). When comparing the models trained on the various *TIMIT* modifications, an improvement is observed for all variants. In contrast, none of the modifications improved the result on the speech data at all. The base result here .3. (see previous section). This makes sense since all of the modifications make the training data less similar to the test data set.
On singing, the Weighted Phoneme Error rate falls by .1 to .15 when using models trained on the pitch-shifted data set (*TimitP*), and by up to .08 when using the time-stretched training data (*TimitT*). The improvements for the corpora with both improvements lie in between. Vibrato does not appear to have a strong influence on the result. The higher error rates for *DAMPTest* can, again, be explained with the fact that this data set has more variation in audio and singing quality.
Pitch shifting might have a stronger effect on the recognition result because it is a stronger modification in the sense that it generates actual new feature values. In contrast, time stretching mostly generates new frames with values similar to the ones in the original data set (and only in between those). Additionally, pitch shifting may introduce more variety that is closer to sung sounds because singers do not usually



(a) Phoneme error rate (b) Weighted phoneme error rate

Figure 5.3: Mean phoneme recognition results on the test data sets using acoustic models trained on *TIMIT* and augmented versions thereof.

shape long vowels just by stretching out their short versions.
Nevertheless, it is interesting to see that both pitch shifting and time stretching improve the result for sung phoneme recognition. This indicates that more variety in the training data is a step in the right direction. However, there is still a lot of room for improvement.

5.3 Phoneme recognition using models trained on a-capella singing

In this section, the most salient tested approach for phoneme recognition in singing is presented. For this method, acoustic models were trained on actual singing recordings. A large set of such recordings was already available from the *DAMP* corpus, but no annotations were included with them. Such annotations were generated automatically from text lyrics available on the internet. In the following subsections, this process is described in detail, some variants are tested, and experiments with these new models are presented.

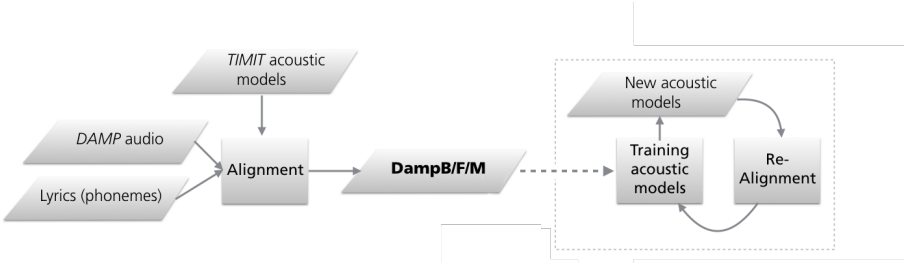


Figure 5.4: An overview of the alignment process. The right-hand part represents the optional bootstrapping.

5.3.1 Corpus construction

As mentioned above, no lyrics annotations are available for the *DAMP* data set, but the textual lyrics can be obtained from the *Smule Sing!* website[2]. All of them were English-language songs. These lyrics were mapped to their phonetic content using the CMU Pronouncing Dictionary[3] with some manual additions of unusual words. As with the other data sets, this dictionary has a phoneme set of 39 phonemes (also see appendix **??**).

An automatic alignment of these lyrics to the *DAMP* audio was then performed using the HMM acoustic models trained on *TIMIT* (see section 5.1). Viterbi alignment was performed on the word and phoneme levels using the HTK framework, using MFCCs and their deltas and double-deltas as features. This is the same principle of so-called "Forced Alignment" that is commonly used in Automatic Speech Recognition [73] (although it is usually done on shorter utterances).

Several different alignment strategies were carried out:

Monophones vs. Triphones Both monophone (i.e. one state per phoneme) and triphone (i.e. three states per triphone modeling the start, middle, and end phases) were tested. Since *TIMIT* only contains monophone annotations, this was done by first splitting the phoneme time frames evenly through three, and then retraining the *TIMIT* acoustic model and re-aligning the data set (with the assumption that the transitions between the triphone states would be "pulled" to the correct times).

One-pass alignment vs. Bootstrapping On top of a one-pass alignment using the Viterbi algorithm, bootstrapping the acoustic models to improve the alignment was also. To clarify:The alignment was first performed on the *DAMP* data sets

[2] `http://www.smule.com/songs`
[3] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

using the *TIMIT* models described above, then acoustic models were trained on the resulting phoneme annotations. Then, those models were used to re-align the *DAMP* data, which was again used to train another model. This was done over three iterations.

A modified version of the alignment algorithm was used for doing alignment with the models trained on the **DAMP** data sets. This approach is based on doing Dynamic Time Warping on the generated phoneme posteriorgrams without punishing very long states. This is similar to the approach described in **??**.

A graphical overview of the alignment process is given in figure 5.4.
Of course, errors cannot be avoided when doing automatic forced alignment. All in all, there were now four combinations of these strategies, which were compared. The next section describes how this alignment procedure was validated and what strategies performed best.
Since there is usually a large number of recordings of the same song, an approach using this information to improve the alignment results was also considered, e.g. by averaging time stamps over the alignments of several recordings. This was not done in this work because recordings tend to have different offsets from the beginning (i.e. silence in the beginning), and the singers also do not necessarily pronounce phonemes at the same time. This might be an avenue for future research, though.

5.3.2 Alignment validation

The alignment approach that was used to create the new *DAMP*-based data sets was first tested on the *ACAP* data set. To recap: This approach employs models trained on the *TIMIT* speech corpus, which are used for Viterbi alignment of the known phonemes to the singing. The result of this is then compared to the manual annotations by calculating the difference between each expected and predicted phoneme transition. As mentioned in 5.1 and shown in figure 5.5, the mean alignment error for this first approach is .16 seconds for the triphone case, and .17 for monophones.

Various models trained on the new **DampB**, **DampF**, and **DampM** training data sets were then tested for the same task. The results are also shown in figure 5.5.
Models trained on the monophonic alignments of **DampB** perform slightly better at this task with mean errors of 0.15 seconds. For the gender-specific models, the error was only calculated for the songs of the matching gender in *ACAP*, resulting in the same average value. Since the gender-specific models do not produce better results,

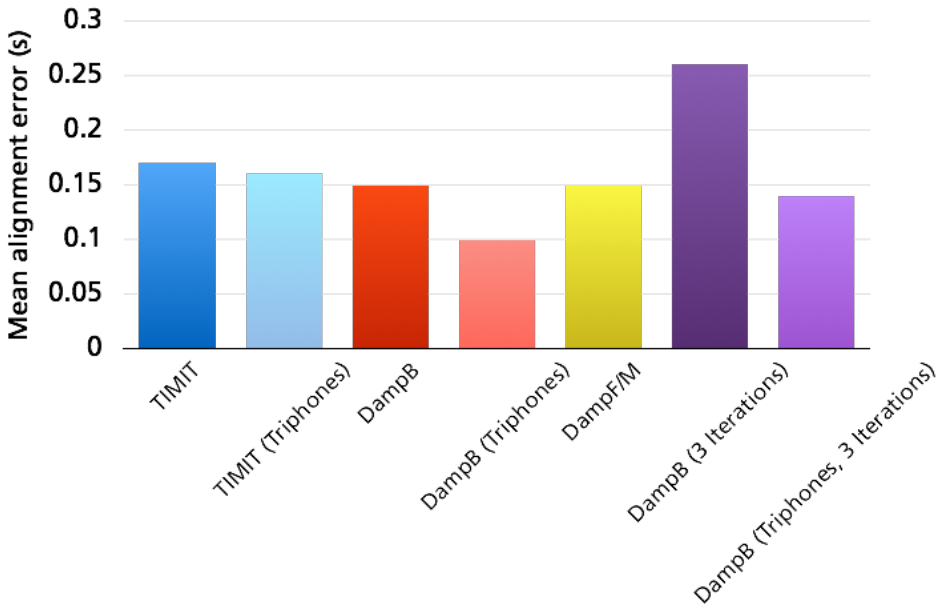


Figure 5.5: Mean alignment error in seconds on the *ACAP* data set. *TIMIT* shows the result for the same models used for aligning the new *DAMP*-based data sets.

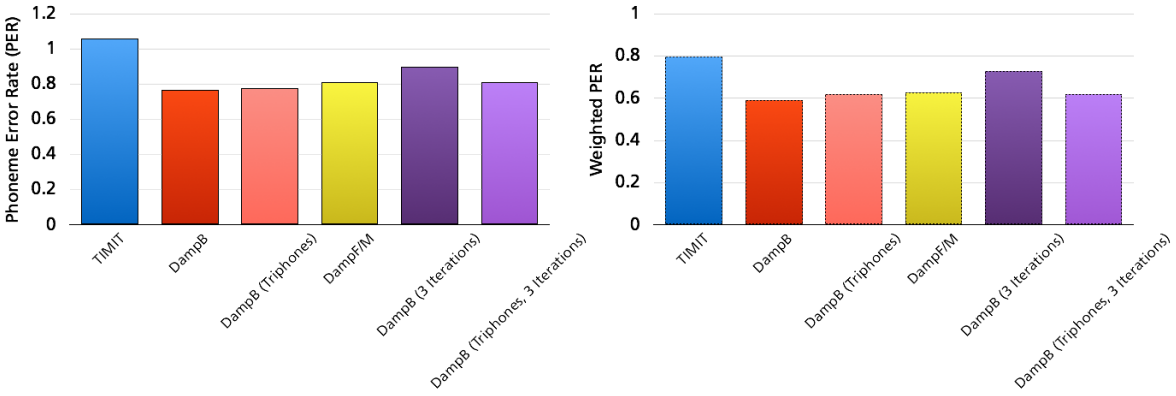
the other experiments were conducted on the mixed training set only. The triphone version of **DampB** performs even better at alignment, with a mean error of 0.1. This might happen because dedicatedly training the model for the start and end parts of phonemes might make the alignment approach more accurate at finding start and end points.

Finally, a model trained on **DampB** over three iterations was also tested for alignment. This model performs much worse at this task with a mean alignment error of 0.26. This might happen because errors in the original alignment of the phonemes may become amplified over these iterations. The effect is not as pronounced for the triphone version, but it is still present.

5.3.3 Phoneme recognition

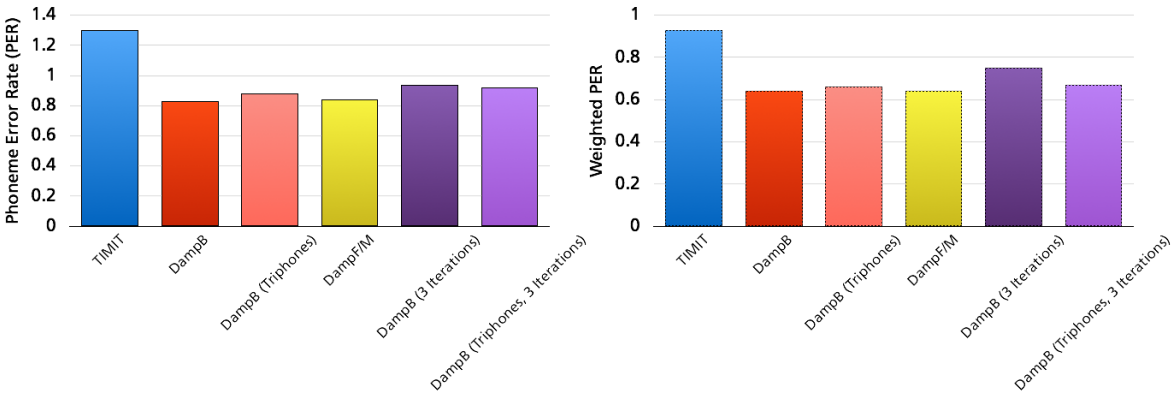
After validating the alignment strategy, phoneme recognition experiments were performed on both the *ACAP* and the *DampTest* corpora. This was possible even though there are no manual annotations for the *DampTest* sets because the expected phonemes are available from the textual lyrics. Again, the phoneme error rate and the weighted phoneme error rate were used as evaluation measures (see **??**).

The results for *ACAP* are shown in figure 5.6. In general, models trained on *DampB*



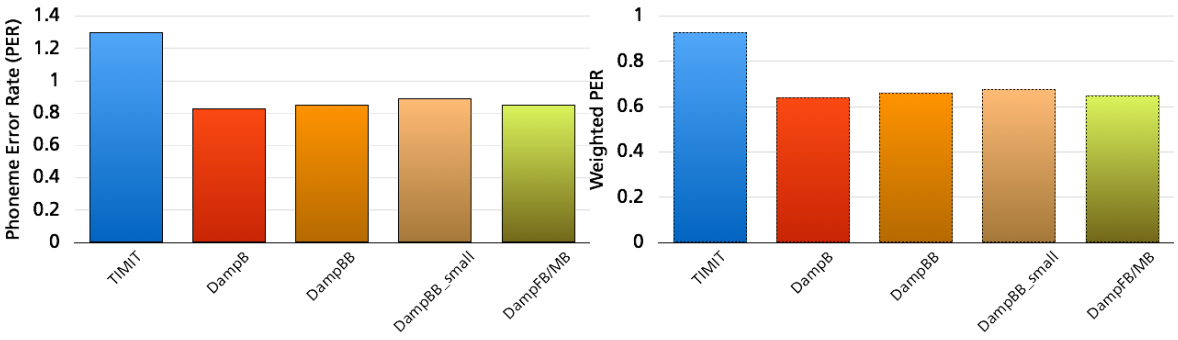
(a) Phoneme error rate (b) Weighted phoneme error rate

Figure 5.6: Mean phoneme recognition results on the *ACAP* data set using acoustic models trained on *Timit* and the new *DAMP*-based data sets.



(a) Phoneme error rate (b) Weighted phoneme error rate

Figure 5.7: Mean phoneme recognition results on the *DampTest* data sets using acoustic models trained on *TIMIT* and the new *DAMP*-based data sets.



(a) Phoneme error rate (b) Weighted phoneme error rate

Figure 5.8: Mean phoneme recognition results on the *DampTest* data sets using acoustic models trained on *TIMIT* and the new *DAMP*-based data sets.

performed much better at phoneme recognition than those trained on *TIMIT*. Compared to these speech-based models, the phoneme error rate falls from 1.06 to 0.77, while the weighted phoneme error rate falls from 0.8 to 0.59. As can be seen from both evaluation measures, using triphone alignments instead of monophones does not improve the results in this case, contrasting with the better alignment results. This might happen because more classes cause more confusion in the model, even though the triphone results were downmapped to monophones for calculating the evaluation measures. As in the alignment results, using gender-specific models does not provide an advantage over a mixed model. (Results for the gender-specific models were only evaluated on songs of the matching gender).
As already seen in the alignment validation results, training models on **DampB** over three iterations actually degrades the result. Again, this might happens because phoneme alignment errors become amplified in this way. Interestingly, the effect is not as strong for the triphone models, perhaps because the three classes per phoneme help to alleviate each other's errors.

The results for the same procedure on the *DampTest* sets are shown in figure 5.7. Results over *DampTestF* and *DampTestM* were averaged. The same general trend can be observed for these results: The phoneme error rate falls from 1.3 to 0.83 when compared to models trained on *TIMIT*, with the weighted phoneme error rate decreasing from 0.93 to 0.64. Using triphones does not contribute to the result, and neither does the three-iteration bootstrapping process for training acoustic models. As in the *ACAP* results, not even the gender-specific models improve the result. This effect might occur because the range of pitch and expressions is much wider in singing than in speech, and therefore gender-specific models may not actually learn as much added helpful information. Other experiments indicated that gender-specific models also do not improve the results when using triphones or three alignment iterations as with the mixed-gender training data.

Going forward, monophonic models trained on mixed-gender data (i.e. *DampB*) appear to be the best and simplest solution. To gain insight into the role of the composition of the training data set, more experiments were conducted with the variants of *DampB* described in section 4.2.3. The results are shown in figure 5.8.

When using the smaller, more balanced version of *DampB* (*DampBB*), the results become somewhat worse, but not much, with a phoneme error rate of .85, and a

weighted phoneme error rate of 0.66. This is particularly interesting because this data set is only 4% the size of the bigger one and training is therefore much faster. With the smallest data set which is only half the size of *DampBB*, the change is similar: The Phoneme Error Rate rises to .89, the Weighted Phoneme Error Rate to .68. Since this data set has roughly the same amount of phonemes as *TIMIT*, this proves that the improvement is actually caused by the acoustic properties of the training data, rather than just the larger amount of data. (Of course, this factor also contributes). The reduced versions of *DampF* and *DampM* were tested as well, but once again, they do not provide an improvement over the mixed gender model.

5.3.4 Error sources

Of course, with an automatic alignment algorithm like this, errors cannot be avoided. To acquire a clearer picture of the reasons for the various misalignments, the audio data where they occurred was analyzed more closely. Some sources of error repeatedly stuck out:

Unclear enunciation Some singers pronounced words very unclearly, often focusing more on musical performance than on the lyrics.

Accents Some singers sung with an accent, either their natural one or imitating the one used by the original singer of the song.

Young children's voices Some recordings were performed by young children.

Background music Some singers had the original song with the original singing running in the background.

Speaking in breaks Some singers spoke in the musical breaks.

Problems in audio quality Some recordings had qualitative problems, especially loudness clipping.

For most of these issues, more robust phoneme recognizers would be helpful. For others, the algorithm could be adapted to be robust to extraneous recognized phonemes (particularly for the speaking problem). If possible, a thorough manual check of the data would be very helpful as well.

5.4 Conclusion

In this chapter, new approaches for phoneme recognition in singing were described. As a starting point, DNN models were trained on the *TIMIT* speech corpus. For verification, these models were evaluated on the test section of *TIMIT*, resulting in a Phoneme Error Rate of .4 and a Weighted Phoneme Error Rate of .3. The results on the *ACAP* singing data set were much worse: A Phoneme Error Rate of .1.06 and a Weighted Phoneme Error Rate of .8, demonstrating the difficulty of phoneme recognition in singing as opposed to speech. Similarly, the Phoneme Error Rate was 1.28 on the *DampTest* data sets, with a Weighted Phoneme Error Rate of .9. Generally, results on the *DampTest* sets are worse than on the *ACAP* test set, which presumably happens because the *DampTest* sets are performed by amateurs, whose enunciation is not as clear as that of the professional singers in the *ACAP* set and who may not always sing the correct lyrics, because the recording quality varies much more, and because the annotations may contain errors due to the automatic phoneme alignment (as opposed to the more realiable manual annotations of *ACAP*).

In order to make the models better adapted to singing, acoustic modifications were performed on the *TIMIT* training data - namely, pitch shifting and time stretching. This increases the Phoneme Error Rate on speech test data (*TIMITTest*, but improves them on sung data. Pitch shifting appears to have a stronger effect than time stretching, which might happen because this modification results in actual changed feature data, whereas time stretching mainly produces more frames with feature values similar or in between the existing ones. On the *ACAP* test data, the lowest Phoneme Error Rate is .95, and the lowest Weighted Phoneme Error Rate is .7 (with the pitch-shifted models). On *DampTest*, those values are decreased to .98 and .76 respectively.

The best-adapted models for recognizing phonemes in singing should be those trained on actual singing data. Unfortunately, there are no big, annotated singing data sets like those for speech. For this reason, the *DAMP* data set, which contains thousands of recordings of unaccompanied amateur singing, was selected, and the text lyrics were obtained from the internet. Then, various strategies for aligning the phonetic content of these lyrics to the audio were tested. In the simplest algorithm, HMM models trained on *TIMIT* were used for Viterbi alignment. This also turned out to be one of the most effective algorithms for the alignment process necessary for constructing this

new singing data set.

The resulting annotations, together with the singing audio data, were then used to train new DNN models for phoneme recognition. Using 6000 of these recordings of singers of both genders, the Phoneme Error Rate was lowered to .77 on the *ACAP* test set, and the Weighted Phoneme Error Rate was lowered to .59. On the *DampTest* sets, those values decreased to .83 and .64 respectively. Neither employing triphones instead of monophones nor training gender-specific models provided additional advantages. Iterating the alignment process decreased the result, possibly because errors in the original alignment become amplified over these iterations.

The influence of training set size was also investigated on the *DampTest* data. Using a phonetically balanced subset of *DAMP* that was just 4% of the size of the originally selected training set worsened the result by only 2 percent points (both in Phoneme Error Rate and Weighted Phoneme Error Rate). Using an even smaller data set that is roughly the size of *TIMIT* increased the Phoneme Error Rate only by a further 4 percent points (and the Weighted Phoneme Error Rate by 2), thereby proving that the better recognition results are caused by the sung content rather than just the size of the training data.

A manual error analysis was performed on cases where the phoneme recognition failed. Errors are frequently caused by linguistically or acoustically difficult segments in the test data, such as accents, children's voices, background music, or additional speaking.

Comparing these results to the state of the art is difficult because of the different testing data used. Considering raw numbers, it can be assumed that the best results are in the range of the best state of the art results, for example those by Mesaros et al. [32] (see section 3.2). In contrast to these approaches, no involved post-processing is employed. In the future, integrating such steps as singer adaptation and language modeling would in all probability serve to improve the approach even more.

6 Language identification

Singing language identification is the task of automatically determining the language of a sung recording. For this task, two algorithms were developed: One based on i-Vector extraction, and one based on the statistics of phoneme posteriorgrams. They will be described in detail in this section.
In both cases, the *NIST2003LRE* and *OGIMultilang* corpora were used for testing the algorithms on speech, and the *YTAcap* data set was used for singing (see chapter **??**).

6.1 Language identification in singing using i-vectors

As described in section **??**, i-Vector extraction is a feature dimension reduction algorithm that was originally developed for speaker recognition, but has since then been employed successfully for other tasks, including language identification. After the publication of this approach for i-Vector extraction for singing language identification, they also became used for other MIR tasks, such as artist recognition and similarity calculation [74] [75].

6.2 Proposed system

Figure 6.1 shows a rough overview over the i-vector classification system.

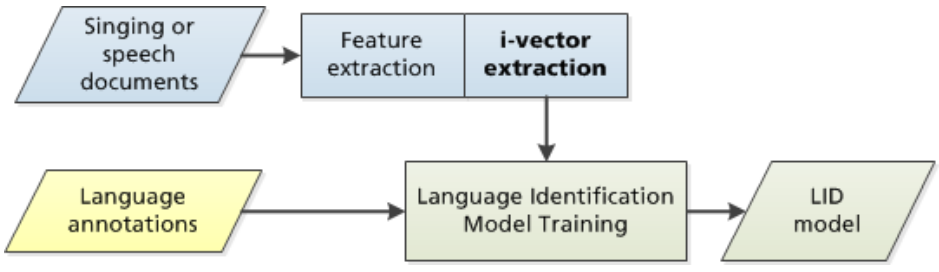


Figure 6.1: Overview of the process for language identification using i-vector extraction.

A number of features were extracted from each audio file. Table 6.1 shows an overview over the various configurations used in training.

Table 6.1: Feature configurations used in training.

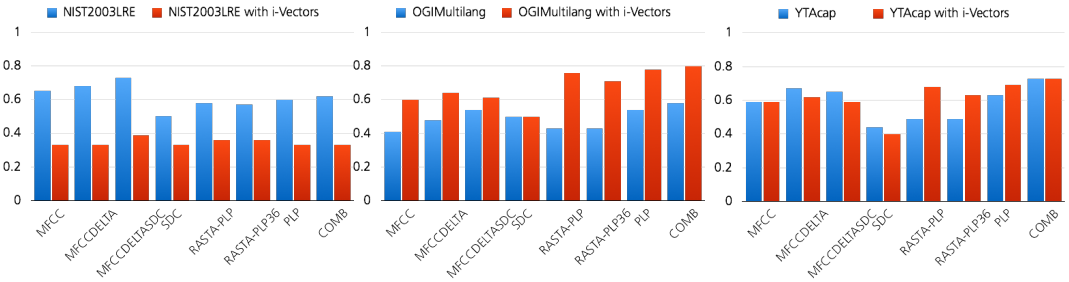
Name	Description	Dimensions
MFCC	MFCC, 20 coefficients	20
MFCCDELTA	MFCC, 20 coefficients, deltas and double deltas	60
MFCCDELTASDC	MFCCDELTA+SDC	117
SDC	SDC with configuration $7-1-3-7$	91
RASTA-PLP	PLP with RASTA processing, model order 13 with deltas and double deltas	39
RASTA-PLP36	PLP with RASTA processing, model order 36 with deltas and double deltas	96
PLP	PLP without RASTA processing, model order 13 deltas and double deltas	39
COMB	PLP+MFCCDELTA	99

For classification, Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) were tested. The MLPs were fixed at three layers, with the middle layer having a dimension of 256. Additional layers did not seem to improve the result. A larger middle layer only improved it slightly. The SVM parameters were determined using a grid-search. For each of those classifiers and data sets, all combinations of the features listed in table 6.1 were tested directly and with i-vector processing. All results were obtained using five-fold cross validation.

6.2.1 Experiments with known speakers

In the first experiment, MLP and SVM models were trained on randomly selected folds of the training data sets. This means that recordings by the same speaker are spread out between the training and test data sets. In theory, i-vectors could be particularly susceptible to capturing speaker characteristics instead of language characteristics, leading to evaluation results that may not be representative of results for unknown speakers. However, this effect is often ignored in literature [76], and an argument can be made that partially training models on speaker properties is realistic for some use cases (i.e. when many of the expected speakers for each language are already known).

The results for the MLP training on the *NIST2003LRE*, *OGIMultilang*, and *YTAcap* data sets are shown in figure 6.2.
As shown in figure 6.2a, the MLP did not produce good results on the *NIST2003LRE* database for any of the feature combinations. *NIST2003LRE* is the smallest of the data sets by a large margin. Since a relatively high-dimensional model was used, this is probably a case of overtraining. The i-vector processing step reduces the training



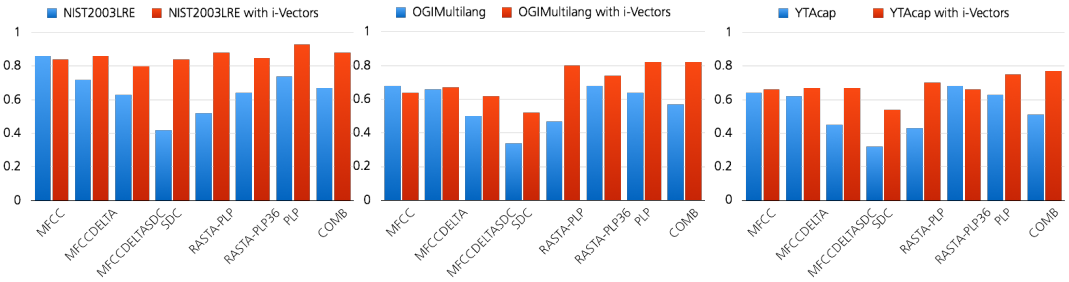
(a) NIST2003LRE (b) OGIMultilang (c) YTAcap

Figure 6.2: Results using MLP models on all three language identification data sets, with or without i-vector processing.

data even further, thus aggravating the problem.

The *OGIMultilang* data set contains roughly 4 times as much data as the *NIST2003LRE* set. With enough data, training an MLP classifier works a lot better. Without i-vector processing, this approach still only reach about 52% accuracy. i-Vector extraction improves the system massively. The best feature configurations are RASTA-PLP (82%), PLP (80%), and COMB (80%).

As with all other experiments, the task becomes harder when attempted on singing data. The results on the *YTAcap* data set are somewhat worse than those on *OGIMultilang*, even though they contain a similar amount of data. The best result without i-vector extraction is still obtained using the COMB feature configuration at 56%. Similar to the *OGIMultilang* experiment, i-vector extraction yields a large improvement. COMB remains the best configuration, now at 77%.



(a) NIST2003LRE (b) OGIMultilang (c) YTAcap

Figure 6.3: Results using SVM models on all three language identification data sets, with or without i-vector processing, with speakers shared between training and test sets.

Figure 6.3 shows the results for the SVM models trained on the same data sets. In general, these models appear to be able to capture the language boundaries better.

In contrast to the MLP experiment, SVMs produced good results on the *NIST2003LRE*

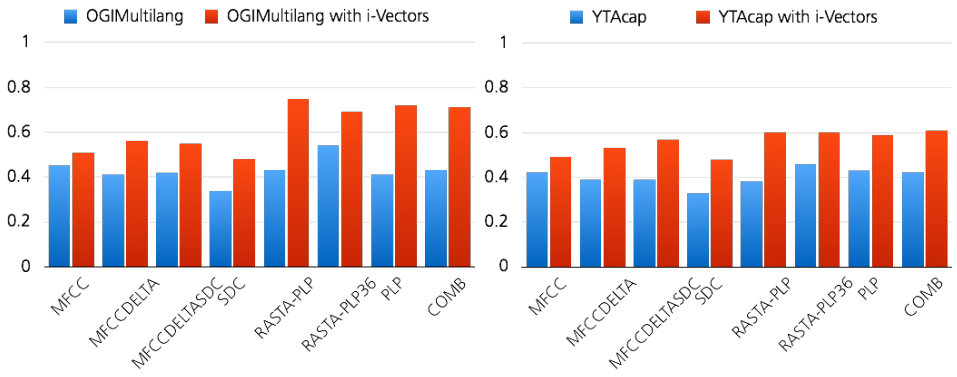
data set for all of the features. They seem to be able to discriminate very well on this small, clean data set. The best result with i-vector processing is 86% for MFCC features. When using i-vectors, 93% are achieved with PLP features. This may, in fact, be close to the upper bound for the classification here, which might be caused by annotation errors or ambiguous data.
The *OGIMultilang* corpus is bigger and more varied than the *NIST2003LRE* corpus, which makes it harder to classify. As shown, the high-dimensional pure features did not perform as well, with a maximum of 68% for MFCCs and RASTA-PLPs with 36 coefficients. Using i-vector extraction improved the result by a large margin. Feature-wise, PLPs without RASTA processing seem to work best at a result of 82%. MFCC and SDC features did not work quite as well, but did not hurt the result either when combined with PLPs (COMB result). It is interesting to see that the i-vector extraction decreased the results for MFCCs, the feature that worked best without it.
Similar to the *OGIMultilang* corpus, the *YTAcap* corpus provides very complex and varied data. The same effects occur on the direct feature training here, too: RASTA-PLPs with 36 coefficients provide the best results, but the accuracy is not very high at 68%. i-vector extraction once again serves to improve the result. The highest results when using i-vector extraction are 75% when using PLP without RASTA processing or 77% for the COMB configuration.

6.2.2 Experiments with unknown speakers

In order to find out what influence the speaker characteristics had on the result, the same experiments were then repeated with training and evaluation sets that strictly separated speakers. This experiment was not performed for the *NIST2003LRE* corpus because no speaker information is available for it. Since the SVM models performed much better in the previous experiment, only these models were tested.

The results are shown in figure 6.4. In general, all configurations performed worse, indicating that some of the characteristics learned by the models come from the speakers rather than the languages. Apart from this, the general trends for the features remain the same, and i-vector extraction still improves the over-all results.

On the *OGIMultilang* corpus, the best result is still obtained with RASTA-PLP features and i-vector processing, but the accuracy falls by around 8 percent points to 75%. On *YTAcap*, the effect is even worse: From an accuracy of 77% with the mixed



(a) OGIMultilang (b) YTAcap

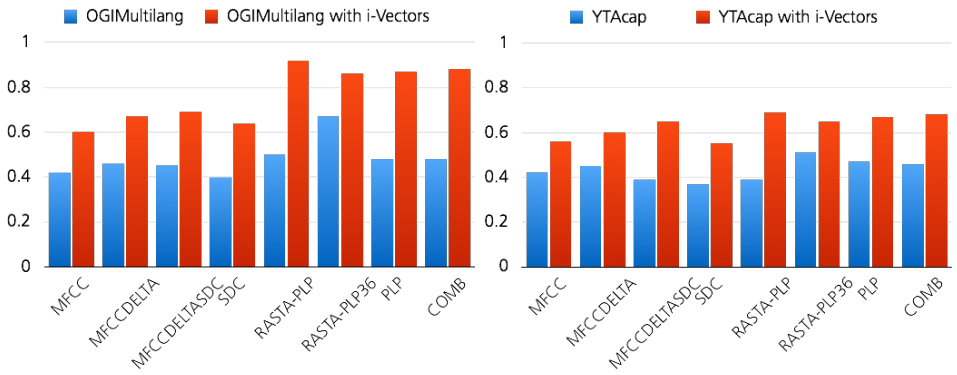
Figure 6.4: Results using SVM models on all three language identification data sets, with or without i-vector processing, with speakers separated between training and test sets.

condition, the result decreases to 61% for the separated condition. The reason for this is probably the wider signal variety in singing as opposed to speech; additionally, *YTAcap* also possesses a wider range of recording conditions than the controlled telephone conditions of *OGIMultilang*. Arguably, the solution for this effect would be the use of larger training data sets, which would be able to cover these acoustic and performance conditions better. Conversely, as the previous results show, the approach functions better when an application scenario can be limited to a range of known speakers, or at least recording conditions (as in the *NIST2003LRE* experiment).

6.2.3 Experiments with utterances combined by speakers

All previous experiments were performed on relatively short utterances of a few seconds in duration. In many application scenarios, much more audio data is available to make a decision about the language. In particular, songs are usually a few minutes in length, and in many cases, only one result per document (=song) is required.
For this reason, results for the *YTAcap* data set were taken from the previous experiment and summed up for each song (and therefore also for each singer). For the *OGIMultilang* corpus, results for all utterances by the same speaker were aggregated in the same fashion, resulting in similar durations of audio. (Again, this experiment was not performed with the *NIST2003LRE* corpus due to the lack of speaker information).

The results are shown in figure 6.5. Overall, aggregation of multiple utterances by the same speakers seems to balance out some of the speaker-specific effects seen in the previous experiment. Taking more acoustic information into account, the models are



(a) OGIMultilang (b) YTAcap

Figure 6.5: Document-wise results using SVM models, with or without i-vector processing.

able to determine the language with higher accuracy.
On the *OGIMultilang* corpus, the result is even better than on the condition with known speakers. The best result rises from 75% for short utterances to 92% for the aggregated documents (both with the RASTA-PLP feature). On the *YTAcap* data sets, the aggregated result is 69% (compared to 60% for line segments).

As suggested in the previous section, the approach produces practically usable results when the problem can be narrowed down, e.g. to known speakers or recording conditions. As this experiment shows, useful results can also be obtained when longer sequences are available for analysis.

6.3 LID in singing using phoneme recognition posteriors

Another developed approach is based upon phoneme statistics derived from phoneme posteriorgrams. To obtain representative statistics for model training, relatively long observations are necessary, but, as described in the previous section, this is the case for many applications, for example when considering song material (e.g. songs of 3-4 minutes in duration). On the other hand, phoneme posteriorgrams need to be calculated for a number of other tasks, such as keyword spotting or lyrics-to-audio alignment.

An overview of the approach is shown in figure 6.6. Posteriorgrams were generated on the test data sets *YTAcap* and *OGIMultilang* using the acoustic models trained on

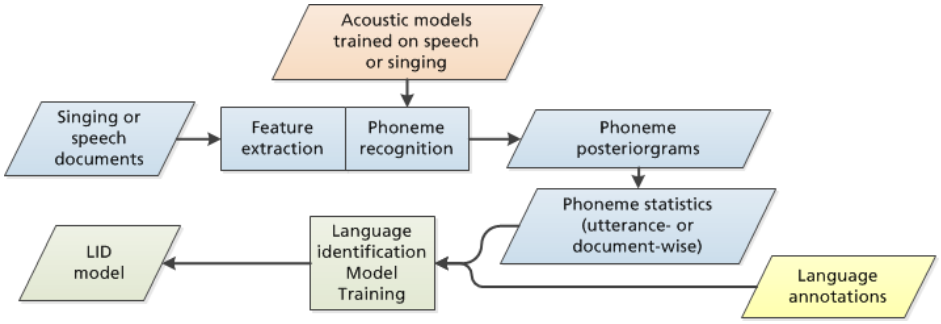


Figure 6.6: Overview of the process for language identification using phoneme statistics.

the *TIMIT* speech data set and on the *DAMP* singing data set as described in section 5.3. To facilitate the following language identification, phoneme statistics were then calculated in two different ways:

Utterance-wise statistics Means and variances of the phoneme likelihoods over each utterance were calculated (or, in the case of *YTAcap*, over each song segment). For further training, the resulting vectors for each speaker/song (=document) were used as a combined feature matrix. As a result, no overlap of speakers/songs was possible between the training and test sets.

Document-wise statistics Mean and variances of the phoneme likelihoods over whole songs or sets of utterances of a single speaker were calculated. This resulted in just two feature vectors per document (one for the means, one for the variances).

Naturally, relatively long recordings are necessary to produce salient statistics. For this reason, the aggregation by speaker/song was done in both cases rather than treating each utterance separately.

Then, Support Vector Machine (SVM) models were trained on the calculated statistics in both variants with the three languages as annotations. Unknown song/speaker documents could then be subjected to the whole process and classified by language. Again, all results in the were obtained using 5-fold cross-validation - i.e., SVMs were trained on 4/5 of each corpus, then the remaining 1/5 was classified with the model. This was done 5 times until each song/speaker document had been classified.

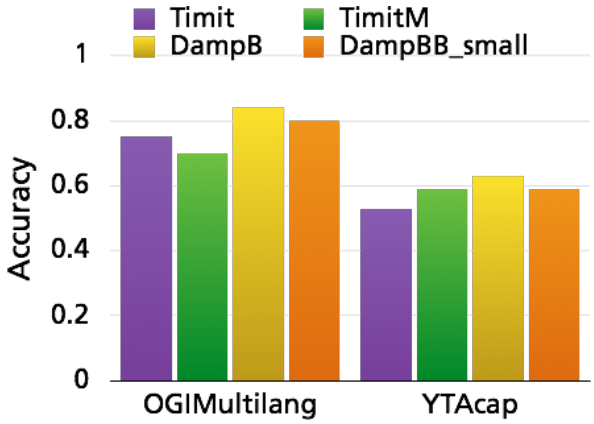


Figure 6.7: Results using document-wise phoneme statistics generated with various acoustic models.

6.3.1 Language identification using document-wise phoneme statistics

In the first experiments, SVM classifiers were trained on the document-wise phoneme statistics, and classification was also performed on a document-wise basis (i.e., only one mean and one variance vector per document). The results are shown in figure 6.7 in terms of accuracy (average retrieval when all documents are classified into exactly one language).

On the singing test set, results are worst when using acoustic models trained on *TIMIT* at just 53% accuracy, and become better when using the model trained on the *TIMIT* variant modified for singing or the small selection of the singing training set (59% each). The best result is achieved when the models are trained on the full singing data set at 63% accuracy.

Surprisingly, the results on the *OGIMultilang* corpus also improve from 75% with the *TIMIT* models to 84% using the *DampB* models. Since *TIMIT* is a very "clean" data set, training on the song corpus might provide some more phonetic variety, acting as a sort of data augmentation. This could be especially important in this context where phonemes are recognized in three different languages.

On both corpora, there is no noticeable bias of the confusion matrix - i.e., the confusions are spread out evenly. This is particularly interesting when considering that the acoustic models were trained on English speech or singing only.

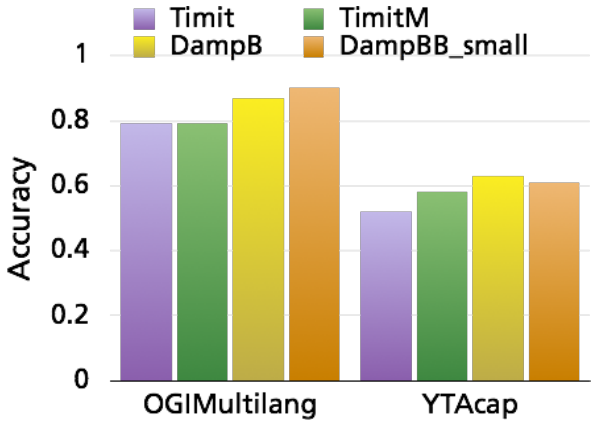


Figure 6.8: Results using utterance-wise phoneme statistics generated with various acoustic models.

6.3.2 Language identification using utterance-wise phoneme statistics

Next, language identification was performed with models trained on the statistics of each utterance contained in the document. The recognition process is still performed on the whole document. The results are reported in figure 6.8.
Phoneme statistics may not be as representative when computed on shorter inputs, but they may provide more information for the backend model training when utilized as a combined feature matrix for a longer document. The results on singing improve slightly to 63% with the acoustic model trained on the small singing corpus (*DampBB_small*) and decrease slightly for the *DampB* model (61%). However, on the speech corpus, the best result rises to 90%.

6.3.3 For comparison: Results for the i-vector approach

For comparison, models from the previous approach were also trained on the same time scales. I-vectors were calculated on the utterance- or the document-wise scale. This was done for PLP and MFCC features. The resulting i-vectors were then used to train SVMs in the same manner as in the previous experiments. (The difference here is that the models are already trained on the aggregated i-vectors, either with those for a whole document or with all i-vectors of the utterances constituting each document aggregated). The results are shown in figure 6.9.
The best result obtained on *YTAcap* corpus is 68% accuracy. This is only 5 percent points higher than the presented approach, which is much easier to implement. On the

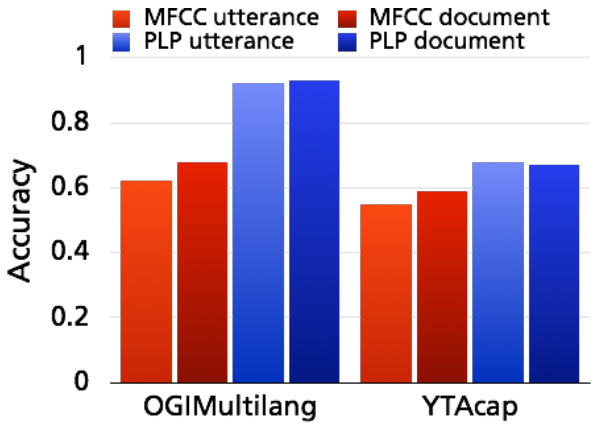


Figure 6.9: Results using utterance- and document-wise i-vectors calculated on PLP and MFCC features.

OGIMultilang data set, the difference is only 3 percent points (93%). Of course, the advantage of the i-vector approach is that it can also be performed on much shorter inputs.

6.4 Conclusion

In this section, two approaches to singing language identification were presented: One based on i-vector processing of audio features, and one based on phoneme statistics generated from posteriorgrams. In both cases, machine learning models were trained on the resulting data.

In the first approach, PLP, MFCC, and SDC features were extracted from audio data, and then run through an i-vector extractor. The generated i-vectors were then used as inputs for MLP and SVM training. The basic idea behind the i-vector approach is the removal of speaker- and channel-dependent components of the signal. This effectively reduces irrelevance to the language identification tasks and also reduces the amount of training data massively.
The smallest data set is the *NIST2003LRE* corpus. No feature configuration achieved good results when using the MLP backend. In this case, the small size of the corpus may lead to overtraining. I-vector processing only amplified this problem by reducing the amount of data even further. The SVM backend, however, produced good results of up to 93% for PLP features with i-vector extraction.

The *OGIMultilang* corpus is a much bigger speech corpus. Training without i-vector extraction did not work well for any feature configuration. The best accuracy for this scenario was 68%. Results of up to 83% were achieved with i-vector processing. There does not seem to be a large difference between SVM and MLP training, with SVM having just a slight advantage.
Language identification for singing was expected to be a harder task than for speech due to the factors described in section 3.1. The results on the *YTAcap* corpus turned out to be somewhat worse than those for the *OGIMultilang* corpus, which is of similar size. As on *OGIMultilang*, i-vector extraction improved the results. In this case, the accuracy improved from 63% to 73%.
The same experiment was performed again, this time with no speaker overlap between training and test sets. The results fell significantly, indicating speaker influence on the model training. In a third experiment, the results were aggregated into documents by each speaker, which again led to improved results. The best accuracy on *OGIMultilang* is 92%, while on *YTAcap*, it is 69%. Both experiments demonstrate that useful results can be obtained when limiting the task, e.g. by training on a set of known speakers or recording conditions, or by analyzing documents of longer durations. Alternatively, a wider range of speakers in the training data could lead to models that generalize better.
Overall, i-vector extraction reduces irrelevance in the training data and there leads to a more effective training. As additional benefits, the training process itself is much faster and less memory is used due to its data reduction properties. Most of the state-of-the-art approaches are based on PPRLM, which requires phoneme-wise annotations and a highly complex recognition system, using both acoustic and language models. In this respect, this system is easier to implement and merely requires language annotations.

The second presented method is a completely new language identification approach for singing. It is based on the output of various acoustic models, from which statistics were generated and SVM models were trained. In contrast to similar approaches for speech, no voice tokenization is performed. Since phoneme recognition on singing is not always reliable, the statistics are calculated directly on the phoneme posteriorgrams, although this does not take any temporal information into account. The acoustic models were trained only on English-language material (speech and singing); it would be interesting to test this with multi-language training data. Due to the statistics-based nature of the approach, it is not suited for language identification of very short audio recordings.

The accuracy of the result for singing is somewhat worse than the results obtained with the i-vector based approach. However, this new approach is much easier to implement and the feature vectors are shorter. For many applications, such posteriors need to be extracted anyway and can then efficiently be used for language identification when long observations are available. The best accuracy of 63% is obtained with acoustic models trained on the *DampB* singing corpus.
Interestingly, the best result on the *OGIMultilang* speech corpus is also obtained with these acoustic models (and is only 3 percent points below the one obtained with the i-vector approach). This possibly happens because the singing corpora provide a wider range of phoneme articulations. It would be interesting to try out these acoustic models for other phoneme recognition tasks on speech where robustness to varied pronunciations is a concern.

7 Sung keyword spotting experiments and results

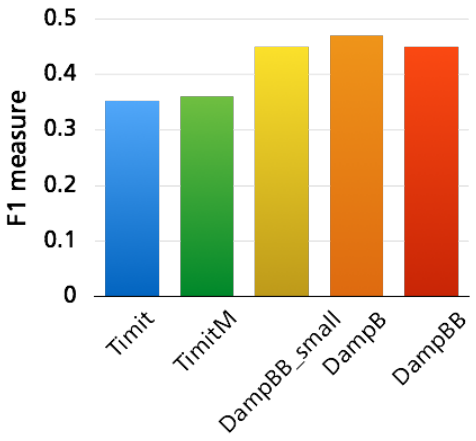
Keyword spotting is another task for which the new acoustic models described in chapter 5 were employed. A keyword-filler HMM algorithm was selected due to its independence on phoneme durations, which is a condition that cannot be fulfilled easily in singing. As described in section **??**, keyword-filler HMMs consist of two sub-HMMs: One to model the keyword and one to model everything else (=filler). The keyword HMM has a simple left-to-right topology with one state per keyword phoneme. The filler HMM is a fully connected loop of all phonemes. When the Viterbi path with the highest likelihood passes through the keyword HMM rather than the filler loop, the keyword is detected. Keyword detection was performed on whole songs, which is a realistic assumption for many practical applications. The *ACAP* and *DampTest* data sets were used for evaluation with the keyword set described in section **??**. Song-wise F_1 measures were calculated for evaluation.

7.1 Keyword spotting using keyword-filler HMMs

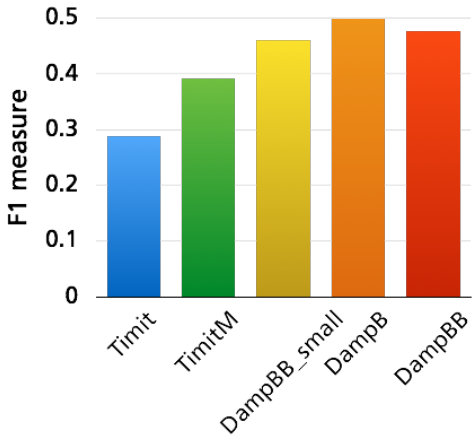
7.1.1 Comparison of acoustic models

Phoneme posteriorgrams were generated with the various acoustic models described in section 5.3. The results in terms of F_1 measure across the whole *DampTest* sets are shown in figure 7.1a. Figure **??** show the results of the same experiment on the small *ACAP* data set.

Across all keywords, a document-wise F_1 measure of 0.35 is obtained using the posteriorgrams generated with the *TIMIT* model on the *DampTest* data sets. This result is slightly higher for the *TimitM* models trained on "songified" speech, and rises to 0.45 using the model trained on the small *DampBB_small* singing data set. Surprisingly, the model trained on *DampBB* is only slightly better than the much



(a) F_1 measures for keyword spotting results on the *DampTest* data sets.



(b) Keyword spotting results on the *ACAP* data set.

Figure 7.1: F_1 measures for keyword spotting results using posteriorgrams generated with various acoustic models.

smaller one. Using the very big *DampB* training data set, the F_1 measure reaches 0.47.

On the hand-annotated *ACAP* test set, the difference is even more pronounced, rising from 0.29 for the *TIMIT* model to 0.5 for *DampB*. This might, again, be caused by the more accurate annotations or by the higher-quality singing. Additionally, the data set is much smaller with fewer occurrences of each keyword, which could emphasize both positive and negative tendencies in the detection.

7.1.2 Gender-specific acoustic models

Keyword spotting was also performed on the posteriorgrams generated with the gender-dependent models trained on *DampF* and *DampM* (also described in section 5.3. The results are shown in figure 7.2.

In contrast to the phoneme recognition results from Experiment C, the gender-dependent models perform slightly better for keyword spotting than the mixed one of the same size, and almost as good as the one trained on much more data (*DampB*). The F_1 measures for the female test set are 0.48 for the *DampB* model, 0.45 for the *DampBB* model, and 0.46 for the *DampFB* model. For the male test set, they are 0.46 and 0.45 for the first two, and 0.46 for the *DampMB* model.

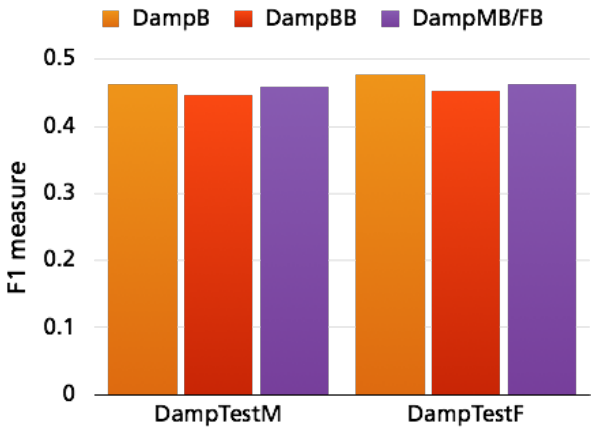


Figure 7.2: F_1 measures for keyword spotting results on the *DampTestM* and *DampTestF* data sets using mixed and gender-dependent models.

7.1.3 Individual analysis of keyword results

Figure 7.3 shows the individual F_1 measures for each keyword using the best model (*DampB*), ordered by their occurrence in the *DampTest* sets from high to low (i.e. number of songs which include the song). There appears to be a tendency for more frequent keywords to be detected more accurately. This happens because a high recall is often achievable, while the precision depends very much on the accuracy of the input posteriorgrams. The more frequent a keyword, the easier it also becomes to achieve a higher precision for it.

As shown in literature [**?**], the detection accuracy also depends on the length of the keyword: Keywords with more phonemes are usually easier to detect. This might explain the relative peak for "every", "little", and "always", in contrast to "eyes" or "world". Since keyword detection systems tend to perform better for longer words and most of the keywords only have 3 or 4 phonemes, this result is especially interesting.

One potential source of error are sequences of phonemes that overlap with keywords, but are not included in the calculation of the precision. Words spelled the same were included, but split phrases or other spellings were not (e.g. "away" as part of "castaway" would be counted, but "a way" would not be counted as "away"). This might artificially lower the results and could be an avenue for future improvement. Additionally, only one pronunciation for each keyword was provided, but there may be several possible.

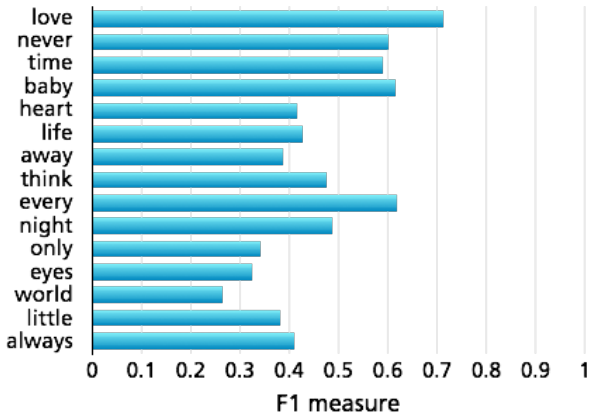


Figure 7.3: Individual F_1 measures for the results for each keyword, using the acoustic model trained on *DampB*.

7.2 Conclusion

In this chapter, an approach for keyword spotting using the new acoustic models trained on singing was described. This was done by extracting phoneme posteriorgrams generated with these new models from the audio and then running them through a keyword-filler approach to detect 15 keywords. The resulting F_1 measure rises from 0.35 for the models trained on speech (*TIMIT*) to 0.47 for the new models. This result is especially interesting because most of the keywords have few phonemes.Gender-dependent models perform slightly better than mixed-gender models of the same size. This approach was only tested with MFCC features. As preliminary experiments suggest [16], other features like TRAP or PLP may work better on singing. So-called log-mel filterbank features have also been used successfully with DNNs [**?**]. Another interesting factor is the size and configuration of the classifiers, of which only one was tested (after a small grid search to validate this choice). Other experiments also suggest that incorporating certain phoneme duration information into the recognition can improve the over-all results [**?**].

As in the phoneme recognition experiments, there is only a slight amount of improvement between the acoustic model trained on all 6000 songs of the *DAMP* data set and the one trained only on 4% of this data. It would be interesting to find the exact point at which additional training data does not further improve the models. On the evaluation side, a keyword spotting approach that allows for pronunciation variants or sub-words may produce better results. Language modeling might also help to alleviate some of the errors made during phoneme recognition.

These models have not yet been applied to singing with background music, which

would be interesting for practical applications. Since this would probably decrease the result when used on big, unlimited data sets, more specified systems would be more manageable, e.g. for specific music styles, sets of songs, keywords, or specialized applications. Searching for whole phrases instead of short keywords could also make the results better usable in practice.
As shown in [49] and [15] and in the next chapter, alignment of textual lyrics and singing already works well. A combined approach that also employs textual information could be very practical.

8 Lyrics Retrieval and Alignment

8.1 HMM-based lyrics-to-audio alignment

8.2 Posteriorgram-based retrieval and alignment

8.3 Phoneme-based retrieval and alignment

8.4 Application: Expletive detection

9 Conclusion

10 Future work

Bibliography

[1] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 57, no. 4, pp. 1738–52, Apr. 1990.

[2] H Hermansky, N Morgan, A Bayya, and P Kohn, "{RASTA-PLP} Speech Analysis," Tech. Rep. TR-91-069, ICSI, 1991.

[3] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, Jan. 1996.

[4] B Bielefeld, "Language identification using shifted delta cepstrum," in *Fourteenth annual speech research symposium*, Baltimore, MD, USA, 1994.

[5] P A Torres-Carrasquillo, E Singer, M A Kohler, R J Greene, D A Reynolds, and Jr. J. R. Deller, "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features," in *International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, USA, 2002, pp. 89–92.

[6] W M Campbell, J P Campbell, D A Reynolds, E Singer, and P A Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech and Language*, vol. 20, pp. 210–229, 2006.

[7] F Allen, E Ambikairajah, and J Epps, "Language identification using Warping and the Shifted Delta Cepstrum," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*, Shanghai, China, 2006, pp. 1–4.

[8] Najim Dehak, Patrick J Kenny, Reda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, may 2011.

[9] D. Martinez, O. Plchot, and L. Burget, "Language Recognition in iVectors Space," in *Interspeech*, Florence, Italy, August 2011, pp. 861–864.

[10] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," in *Digital Signal Processing*, 2000, p. 2000.

[11] C Charbuillet, D Tardieu, and G Peeters, "GMM Supervector for content based music similarity," *... Conference on Digital ...*, , no. 1, pp. 1–4, 2011.

[12] Y K Muthusamy, E Barnard, and R A Cole, "Reviewing automatic language identification," *IEEE Signal Procesing Magazine*, vol. 11, no. 4, pp. 33–41, oct 1994.

[13] E Singer, P A Torres-Carrasquillo, T P Gleason, W M Campbell, and D A Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification," in *Proceedings of Eurospeech*, Geneva, Switzerland, 2003, pp. 1345–1348.

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

[15] H. Fujihara and M. Goto, *Multimodal Music Processing*, chapter Lyrics-to-audio alignment and its applications, Dagstuhl Follow-Ups, 2012.

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

[17] Jan Schlüter, "Learning to Pinpoint Singing Voice from Weakly Labeled Examples," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, USA, 2016.

[18] Chong kai Wang, Ren-Yuan Lyu, and Yuang-Chin Chiang, "An automatic singing transcription system with multilingual singing lyric recognizer and robust melody tracker.," in *INTERSPEECH*. 2003, ISCA.

[19] T. Hosoya, M. Suzuki, A. Ito, and S. Makino, "Lyrics recognition from a singing voice based on finite state automaton for music information retrieval," *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 532–535, 2005.

[20] M.J.F. Gales and P.C. Woodland, "Mean and variance adaptation within the mllr framework," *Computer Speech & Language*, vol. 10, pp. 249–264, 1996.

[21] Matthias Gruhne, Konstantin Schmidt, and Christian Dittmar, "Phoneme recognition in popular music," *ISMIR*, 2007.

[22] Matthias Gruhne, Konstantin Schmidt, and Christian Dittmar, "Detecting phonemes within the singing of polyphonic music," *Proceedings of ICoMCS ...*, , no. December, pp. 60–63, 2007.

[23] Karin Dressler, "Sinusoidal Extraction using an efficient implementation of a multi-resolution {FFT}," in *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06)*, sep 2006, pp. 247–252.

[24] Hynek Hermansky, Nelson Morgan, Aruna Bayya, and Phil Kohn, "Rasta-plp speech analysis technique," in *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, Washington, DC, USA, 1992, ICASSP'92, pp. 121–124, IEEE Computer Society.

[25] Aki Härmä and Unto K. Laine, "A comparison of warped and conventional linear predictive coding," *IEEE Trans. Speech and Audio Processing*, vol. 9, pp. 579–588, 2001.

[26] Hiromasa Fujihara, Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno, *Singer identification based on accompaniment sound reduction and reliable frame selection*, pp. 329–336, 2005.

[27] 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.

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

[29] Gunnar Fant, "The source filter concept in voice production," *Quarterly Progress and Status Report*, vol. 22, no. 1, pp. 021–037, 1981.

[30] Annamaria Mesaros and Tuomas Virtanen, "Adaptation of a speech recognizer for singing voice," *European Signal Processing Conference*, , no. 1, pp. 1779–1783, 2009.

[31] A Mesaros and T Virtanen, "Recognition of phonemes and words in singing," *Acoustics Speech and Signal . . .*, pp. 1–4, 2010.

[32] Annamaria Mesaros and T Virtanen, "AUTOMATIC UNDERSTANDING OF LYRICS FROM SINGING," *Akustiikkapäivät*, pp. 1–6, 2011.

[33] Tuomas Virtanen, Annamaria Mesaros, and Matti Ryynänen, "Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music," in *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audition, SAPA 2008, Brisbane, Australia, September 21, 2008*, 2008, pp. 17–22.

[34] Annamaria Mesaros and Tuomas Virtanen, "Automatic Recognition of Lyrics in Singing," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, pp. 1–11, 2010.

[35] M McVicar, DPW Ellis, and M Goto, "LEVERAGING REPETITION FOR IMPROVED AUTOMATIC LYRIC TRANSCRIPTION IN POPULAR MUSIC," *mattmcvicar.com*, pp. 3141–3145, 2014.

[36] Jonathan G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," 1997, pp. 347–352.

[37] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka, "Rwc music database: Popular, classical, and jazz music databases," in *In Proc. 3rd International Conference on Music Information Retrieval*, 2002, pp. 287–288.

[38] 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)*, Copenhagen, Denmark, 2012, pp. 494–499.

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

[40] Ye Wang, Min-Yen Kan, Tin Lay Nwe, Arun Shenoy, and Jun Yin, "LyricAlly: automatic synchronization of acoustic musical signals and textual lyrics," *Proceedings of the 12th annual ACM international conference on Multimedia*, vol. 0, pp. 212–219, 2004.

[41] Min-Yen Kan, Ye Wang, Denny Iskandar, Tin Lay Nwe, and Arun Shenoy, "Lyrically: Automatic synchronization of textual lyrics to acoustic music signals," *IEEE Trans. Audio, Speech & Language Processing*, vol. 16, no. 2, pp. 338–349, 2008.

[42] Denny Iskandar, Ye Wang, Min-Yen Kan, and Haizhou Li, "Syllabic level automatic synchronization of music signals and text lyrics," in *Proceedings of the 14th ACM International Conference on Multimedia*, New York, NY, USA, 2006, MM '06, pp. 659–662, ACM.

[43] Hiromasa Fujihara, Masataka Goto, Jun Ogata, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno, "Automatic synchronization between lyrics and music CD recordings based on viterbi alignment of segregated vocal signals," in *Eigth IEEE International Symposium on Multimedia (ISM 2006), 11-13 December 2006, San Diego, CA, USA*, 2006, pp. 257–264.

[44] H. Fujihara and M. Goto, "Three techniques for improving automatic synchronization between music and lyrics: Fricative detection, filler model, and novel feature vectors for vocal activity detection," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, NV, USA, 2008, pp. 69–72.

[45] Matthias Mauch, Hiromasa Fujihara, and Masataka Goto, "Lyrics-to-audio alignment and phrase-level segmentation using incomplete internet-style chord annotations," in *Proceedings of the 7th Sound and Music Computing Conference (SMC 2010)*, 2010.

[46] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into hmm-based lyrics-to-audio alignment," *Trans. Audio, Speech and Lang. Proc.*, vol. 20, no. 1, pp. 200–210, Jan. 2012.

[47] Chi Hang Wong, Wai Man Szeto, and Kin Hong Wong, "Automatic lyrics alignment for cantonese popular music," *Multimedia Syst.*, vol. 12, no. 4-5, pp. 307–323, 2007.

[48] Kyogu Lee and Markus Cremer, "Segmentation-based lyrics-audio alignment using dynamic programming," in *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, 2008, pp. 395–400.

[49] A. Mesaros and T. Virtanen, "Automatic alignment of music audio and lyrics," in *DaFX-08*, Espoo, Finland, 2008.

[50] Rong Gong, Philippe Cuvillier, Nicolas Obin, and Arshia Cont, "Real-time audio-to-score alignment of singing voice based on melody and lyric information," in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*. 2015, pp. 3312–3316, ISCA.

[51] Georgi Dzhambazov, Ajay Srinivasamurthy, Sertan Sentürk, and Xavier Serra, "On the use of note onsets for improved lyrics-to-audio alignment in turkish makam music," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, Eds., 2016, pp. 716–722.

[52] Motoyuki Suzuki, Toru Hosoya, Akinori Ito, and Shozo Makino, "Music information retrieval from a singing voice based on verification of recognized hypotheses," in *Proceedings of the 7th International Conference on Music Information Retrieval*, Roger Dannenberg, Kjell Lemstrom, and Adam Tindale, Eds., Victoria, BC, Canada, Oct. 2006, University of Victoria, pp. 168–171, University of Victoria.

[53] Motoyuki Suzuki, Toru Hosoya, Akinori Ito, and Shozo Makino, "Music information retrieval from a singing voice using lyrics and melody information," *EURASIP J. Adv. Sig. Proc.*, vol. 2007, 2007.

[54] Chung-Che Wang, Jyh-shing Roger Jang, and Wennen Wang, "An Improved Query by Singing/Humming System Using Melody and Lyrics Information," *11th ISMIR*, , no. Ismir, pp. 45–50, 2010.

[55] A. Mesaros and T. Virtanen, "Recognition of phonemes and words in singing.," in *ICASSP*. 2010, pp. 2146–2149, IEEE.

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

[57] W.-H. Tsai and H.-M. Wang, "Towards Automatic Identification Of Singing Language In Popular Music Recordings," in *5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004, pp. 568–576.

[58] J Schwenninger, R Brueckner, D Willett, and M E Hennecke, "Language Identification in Vocal Music," in *7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 377–379.

[59] V Chandraskehar, M E Sargin, and D A Ross, "Automatic language identification in music videos with low level audio and visual features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 5724–5727.

[60] Hiromasa Fujihara, Masataka Goto, and Jun Ogata, "Hyperlinking lyrics: A method for creating hyperlinks between phrases in song lyrics," in *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, 2008, pp. 281–286.

[61] G. Dzhambazov, S. Sentürk, and X. Serra, "Searching lyrical phrases in a-capella turkish makam recordings," in *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR)*, Malaga, Spain, 2015.

[62] J. S. Garofolo et al., "TIMIT Acoustic-Phonetic Continuous Speech Corpus," Tech. Rep., Linguistic Data Consortium, Philadelphia, 1993.

[63] Victor Zue, Stephanie Seneff, and James Glass, "Speech database development at MIT: Timit and beyond," *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[64] A. Martin and M. Pryzbocki, "2003 NIST Language Recognition Evaluation," Tech. Rep., Linguistic Data Consortium, Philadelphia, 2006.

[65] J. C. Smith, *Correlation analyses of encoded music performance*, Ph.D. thesis, Stanford University, 2013.

[66] Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*, Cambridge University Press, 2006.

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

[68] 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*, 2000, pp. 29–32.

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

[70] R. M. Golden J. L. Flanagan, "Phase vocoder," *Bell System Technical Journal*, pp. 1493–1509, Nov. 1966.

[71] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

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

[73] D. Jurafsky and J. H. Martin, *Speech and language processing: An introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, 2009.

[74] Hamid Eghbal-zadeh, Bernhard Lehner, Markus Schedl, and Gerhard Widmer, "I-vectors for timbre-based music similarity and music artist classification," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, Meinard Müller and Frans Wiering, Eds., 2015, pp. 554–560.

[75] H. Eghbal-zadeh, M. Schedl, and G. Widmer, "Timbral modeling for music artist recognition using i-vectors," in *European Signal Processing Conference (EUSIPCO)*, Nice, France, 2015.

[76] Alvin Martin and Craig Greenberg, "The 2009 NIST Language Recognition Evaluation," *Proceedings of Odyssey*, , no. July, pp. 165–171, 2010.

[77] M Mehrabani and J H L Hansen, "Language identification for singing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 4408–4411.

[78] Anna M. Kruspe, "Improving singing language identification through i-vector extraction," in *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, 2014, pp. 227–233.

List of Figures

List of Tables

A Appendix

A lot of stuff that didn't fit into the main part ...

B Eigenständigkeitserklärung

Die vorliegende Arbeit habe ich selbstständig ohne Benutzung anderer als der angegebenen Quellen angefertigt.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Quellen entnommen wurden, sind als solche deutlich kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer oder anderer Prüfungen noch nicht vorgelegt worden.

Ilmenau, 17.12.2013

Sheldon Cooper

Thesis Summary

1. Scissors cuts paper, paper covers rock, rock crushes lizard, lizard poisons Spock, Spock smashes scissors, scissors decapitates lizard, lizard eats paper, paper disproves Spock, Spock vaporizes rock, and as it always has, rock crushes scissors.
2. I'm not insane, my mother had me tested!
3. All I need is a healthy ovum and I can grow my own Leonard Nimoy!

Thesen

1. These 1
2. These 2
3. These 3