NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
“DATA SCIENCE AND INFORMATION TECHNOLOGIES”

Masters Thesis

GREC: Multi-domain Speech Recognition for the Greek Language

Georgios G. Rouvalis

ATHENS

JANUARY 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
“ΕΠΙΣΤΗΜΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ”

Διπλωματική Εργασία

GREC: Πολυτομεακή αναγνώριση ομιλίας για την Ελληνική γλώσσα

Γεώργιος Γ. Ρούβαλης

ΑΘΗΝΑ

ΙΑΝΟΥΑΡΙΟΣ 2023

Masters Thesis

GREC: Multi-domain Speech Recognition for the Greek Language

Georgios G. Rouvalis
A.M.: DS2200006

SUPERVISOR: Vassilis Katsouros, Research Director, ATHENA Research and Innovation Center

EXAMINATION COMMITTEE:

Vassilis Katsouros, Research Director, ATHENA Research and Innovation Center

Athanasios Katsamanis, Principal Researcher, ATHENA Research and Innovation Center

Yannis Panagakis, Associate Professor, National and Kapodistrian University of Athens

JANUARY 2023

Διπλωματική Εργασία

GREC: Πολυτομεακή αναγνώριση ομιλίας για την Ελληνική γλώσσα

Γεώργιος Γ. Ρούβαλης
Α.Μ.: DS2200006

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Βασίλειος Κατσούρος, Διευθυντής Ερευνών, Ερευνητικό Κέντρο ”Αθηνά”

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Βασίλειος Κατσούρος, Διευθυντής Ερευνών, Ερευνητικό Κέντρο ”Αθηνά”

Αθανάσιος Κατσαμάνης, Κύριος Ερευνητής, Ερευνητικό Κέντρο ”Αθηνά”

Ιωάννης Παναγάκης, Αναπληρωτής Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΙΑΝΟΥΑΡΙΟΣ 2023

ABSTRACT

One of the leading challenges in Automatic Speech Recognition (ASR) is the development of robust systems that can perform well under multiple settings. In this work we construct and analyze GREC, a large, multi-domain corpus for automatic speech recognition for the Greek language. GREC is a collection of three available subcorpora over the domains of “news casts”, “crowd-sourced speech”, “audiobooks”, and a new corpus in the domain of “public speeches”. For the creation of the latter, HParl, we collect speech data from recordings of the official proceedings of the Hellenic Parliament, yielding, a dataset which consists of 120 hours of political speech segments. We describe our data collection, pre-processing and alignment setup, which are based on Kaldi toolkit. Furthermore, we perform extensive ablations on the recognition performance of Gaussian Mixture (GMM) - Hidden Markov (HMM) models and Deep Neural Network (DNN) - HMM models over the different domains. Finally, we integrate speaker diarization features to Kaldi-gRPC-Server, a modern, pythonic tool based on PyKaldi and gRPC for streamlined deployment of Kaldi based speech recognition.

SUBJECT AREA: Technology - Computer Science

KEYWORDS: Unsupervised domain adaptation, GREC, Automatic speech recognition, Multi-domain dataset

ΠΕΡΙΛΗΨΗ

Μία από τις κορυφαίες προκλήσεις στην Αυτόματη Αναγνώριση Ομιλίας είναι η ανάπτυξη ικανών συστημάτων που μπορούν να έχουν ισχυρή απόδοση μέσα από διαφορετικές συνθήκες ηχογράφησης. Στο παρόν έργο κατασκευάζουμε και αναλύουμε το GREC, μία μεγάλη πολυτομεακή συλλογή δεδομένων για αυτόματη αναγνώριση ομιλίας στην ελληνική γλώσσα. Το GREC αποτελείται από τρεις βάσεις δεδομένων στους θεματικούς τομείς των «εκπομπών ειδήσεων», «ομιλίας από δωρισμένες εγγραφές φωνής», «ηχητικών βιβλίων» και μιας νέας συλλογής δεδομένων στον τομέα των «πολιτικών ομιλιών». Για τη δημιουργία του τελευταίου, συγκεντρώνουμε δεδομένα ομιλίας από ηχογραφήσεις των επίσημων συνεδριάσεων της Βουλής των Ελλήνων, αποδίδοντας ένα σύνολο δεδομένων που αποτελείται από 120 ώρες ομιλίας πολιτικού περιεχομένου. Περιγράφουμε με λεπτομέρεια την καινούρια συλλογή δεδομένων, την προεπεξεργασία και την ευθυγράμμιση ομιλίας, τα οποία βασίζονται στο εργαλείο ανοιχτού λογισμικού Kaldi. Επιπλέον, αξιολογούμε την απόδοση των μοντέλων Gaussian Mixture (GMM) - Hidden Markov (HMM) και Deep Neural Network (DNN) - HMM όταν εφαρμόζονται σε δεδομένα από διαφορετικούς τομείς. Τέλος, προσθέτουμε τη δυνατότητα αυτόματης δεικτοδότησης ομιλητών στο Kaldi-gRPC-Server, ενός εργαλείου γραμμένο σε Python που βασίζεται στο PyKaldi και στο gRPC για βελτιωμένη ανάπτυξη μοντέλων αυτόματης αναγνώρισης ομιλίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνολογία - Επιστήμη των Υπολογιστών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μη-επιβλεπόμενη προσαρμογή πεδίου, GREC , Αυτόματη αναγνώριση ομιλίας, Πολυτομεακή συλλογή δεδομένων

ACKNOWLEDGEMENTS

This work was conducted in the Institute for Language and Speech Processing (ILSP), a leading R&D department of Athena Research Center in the area of Language Technologies in Greece and a centre of excellence for basic and applied research in the field.

I would like to acknowledge my professor Vassilis Katsouros, for the great opportunity he gave me to be involved in this project, for his valuable help and support, and the trust he showed me from the very start. Additionally, I want to express my appreciation to my thesis technical advisor, Georgios Paraskevopoulos, for his mentorship and availability he provided in order to discuss and brainstorm all the steps of the development phase, as well as, his extremely helpful insights throughout the whole process.

I would also like to acknowledge every other member of the ILSP team for their suggestions and guidance during every stage of this thesis along with creating a pleasant working environment that helped me stay positive and motivated at all times.

To my family

CONTENTS

LIST OF FIGURES

LIST OF TABLES

SOURCE CODE

1. INTRODUCTION

1.1 Motivation

Automatic Speech Recognition (or Speech-to-text) systems have seen wide development in the recent years and have changed the way users interact with technology. Still, developing real-world speech driven systems is challenging for under-represented languages, where transcribed speech data are not available at mass. One example of an under-represented language is Modern Greek, where the availability of all public ASR datasets amounts to less than 100 hours. While this amount of data, paired with language model adaptation techniques may be sufficient for the development of specialized, domain-constrained speech applications, it is not enough for building robust, general-purpose speech systems. Thus, efforts to develop resources for under-represented languages can have a significant impact.

Another challenge for speech recognition systems, is the development of models that can perform well in multiple real-world situations. On the acoustic model side, this means models should perform well for different speakers and accents, under varying recording conditions and devices. On the language model side, models should span large vocabularies for different application domains. Furthermore, human speech and language evolves over time, with new terms, idioms and colloquialisms. As an example, the Greek word for corona virus ("κορωνοϊός"), is a compound word that was not recognized by original, unadapted versions of our speech recognition system. Therefore, speech resources should evolve too and allow for iterative adaptation of systems to the vocabulary that people use in their daily lives.

In this work, we use and develop publicly available resources for Greek speech. First, we develop HParl (v1) from transcribed parliamentary proceedings. HParl is to our knowledge the largest public speech corpus to date for the Greek language. HParl is a sizeable 120 hour speech corpus consisting of segmented and aligned political speeches given by 387 politicians in the Greek parliament in the time period 2018-2022[1] that were performed in two large parliament chambers, allowing for modeling of reverberant speech. One benefit of using this public resource, is that the dataset can be kept always up to date with the evolving language and vocabulary that results from new developments in the world. The full data collection and preprocessing pipeline are described. For multi-domain evaluation, we merge HParl with three other datasets, namely, Logotypografia, Common Voice (cv9) and CSS10 [2], to create GREC, a standardised test-bed for Unsupervised Domain Adaptation (UDA) corpus of ASR models for Modern Greek, spanning the domains of news casts, crowd-sourced short speech, audiobooks, and public (political) speeches. We present our analysis for in-domain and out-of-domain performance of state-of-the-art Kaldi-based models.

[1] A subset of speeches in this time period were used for this work as a proof of concept due to computational limitations.

[2] We do not re-distribute these datasets, rather provide links to the original distributors.

1.2 Related Work

1.2.1 Domain Shift

Speech recognition models have matured to the point where they can enable commercial, real-world applications, e.g. voice assistants, dictation systems etc. However, the performance of ASR systems rapidly deteriorates when the test data domain differs significantly from the training data. Domain mismatches can be caused by differences in the recording conditions or shifts in the target vocabulary. These issues are extenuated in the case of low-resource languages, where diversity in the training data is limited due to poor availability of high-quality transcribed audio. Therefore, specialized domain adaptation approaches need to be employed when operating under domain-shift.

UDA methods are of special interest, as they do not rely on annotation of domain-specific data for supervised in-domain training. In the case of speech recognition the importance of UDA is extenuated, as the transcription and alignment process is especially expensive and time-consuming. Therefore, adaptation methods have been explored since the early days of ASR, at different levels of the system and different deployment settings [10]. Classical adaptation techniques focus on speaker adaptation through normalization [44] or feature-based approaches [32, 102]. Language Model (LM) adaptation has also been widely explored through cache based models and interpolation [157] for n-gram LMs, or through auxiliary features and fine-tuning for neural LMs [37, 38]. General adaptation techniques through audio can focus on data augmentation schemes [36, 40], pseudolabeling/teacher-student learning [39] or domain adversarial training [27]. Researchers have also begun to explore source-free adaptation of ASR models at test time [90].

Datasets concentrating on this significant obstacle have been created for research purposes such as handling reverberations [79], noisy speech [133], speech separation [21], distant speech recognition [132] and speech recorded in cars [57]. The CHiME challenge [6, 7, 19, 160], which published datasets including actual and simulated multi-channel speech data in noisy real-world settings as well as in multi-speaker scenarios, is a notable example of multi-domain handling. Due to the evident cost of obtaining labeled training data across all of these varied circumstances, unsupervised domain adaptation has emerged as a potential approach for adapting deep neural networks to distinct related domains [23, 27, 65].

With respect to supervised domain adaptation resources, the research community has made substantial efforts publishing speech corpora for ASR, with the majority of them in the literature related to the English language. Constructed on top of the LibriSpeech corpus, Libridapt [98] is a representation of the difficult real-world settings that ASR models must deal with. The dataset consist of 7200 hours of English speech captured on portable and embedded size microphones, covering 72 distinct domains. Mathur et al. emphasized the potential for domain adaptation algorithms to modify or align the representations of speech features across microphones when ASR models are used in settings where the microphones would not match. Additionally, through relatively increased WER in different accents evaluation they underlined the need to adapt a model trained on a small user group to a larger user group to draw more definitive results. The results related to domain shifts caused by ambient noise showed degradation of the performance of state-of-the-art ASR models, therefore making it a pertinent problem for domain adaptation research. A year after, Gigaspeech [18] was released as a complementary, evolving, multi-domain English speech recognition corpus. The corpus comprises of $40,000$ hours of audio in

total for semi-supervised or unsupervised training, as well as $10,000$ hours of high quality speech transcription for supervised ASR training. By covering multiple-sources i.e. audiobooks, podcasts and YouTube videos· multiple styles, for both read and spontaneous speech· multiple topics i.e. arts, science,sports, etc. the dataset is ideal for generalized ASR tasks. At the same time, The People's Speech [47], an open-source $30,000$-hour and growing supervised conversational English speech recognition dataset licensed for academic and commercial usage was published. The information was gathered by looking for suitably licensed audio files containing already-transcribed content online. They demonstrated that a model developed using this dataset has a 9.98% word error rate (WER) on the test-clean test set of Librispeech.

Multiple initiatives have been launched to create multi-domain data collections for languages other than English as well. The largest open source Mandarin speech corpus that includes a variety of domains to meet the needs of different speech recognition tasks at the moment is Wenetspeech [170]. The corpus consists of $10,000+$ hours of high-quality labeled speech, $2,400+$ hours of weakly labeled speech, and about $10,000$ hours unlabeled speech, with $22400+$ hours in total. In the same family of Sino-Tibetan languages, a Multi-domain Cantonese Corpus (MDCC) was recently published, which includes 73.6 hours of clean read speech combined with transcripts and was compiled from Cantonese audiobooks from Hong Kong. It covers a wide range of subjects in the areas of philosophy, politics, education, culture, lifestyle, and family.

1.2.2 Parliamentary Speech Data

Acquiring acoustic data for languages with limited resources is a crucial and difficult undertaking. In recent times, parliamentary data has become fine source to consider significant factors in the multi-domain field like audio capturing devices variation, different accents of speakers and environmental variables like reverberation. As a consequence, the specific type of data source can be traced in several works for different language studies.

The Althingi [58] corpus is the largest speech corpus currently available in Icelandic. By utilizing all available speech training data the researchers evaluated the performance of the dataset for different acoustic models and achieved WER of 14.76%. Based on this corpus, two years later Helgadóttir et al. introduced the first open source speech recognizer in use for Icelandic [59].

The Croatian language also earned its place on the map of the rapidly developing language technologies with the release of ParlaSpeech-HR [92]. The primary resource of the corpus is the parliamentary proceedings available through the ParlaMint corpus [33]. Results from experiments on the generated dataset revealed that there was a 4–5% difference between the spoken data and the parliamentary transcripts, which was also the word error rate of their best ASR system.

In [49] the research group produced BG-PARLAMA, the largest currently available corpus of Bulgarian speech suitable for training modern ASR systems. The ASR system trained on the BG-PARLAMA corpus achieved an error rate of around 7%.

The ParCzech 3.0 [83] is the latest speech corpus of Czech parliamentary data which preserves and formalizes as much metadata as possible (speakers and their gender, structure of the meetings and more) and adds also automatic annotation: morphological tags, syntactic structure and named entities.

The Finnish Parliament ASR corpus [161] corpus is the largest publicly available collection

of manually transcribed speech data for the Finnish language with over 3000 hours of speech, 449 speakers and rich demographic metadata.

FT SPEECH, also known as Folketing (FT) [80], is the latest speech corpus created from the recorded meetings of the Danish Parliament. The corpus contains over $1,800$ hours of transcribed speech by a total of 434 speakers. It is significantly larger in duration, vocabulary, and amount of spontaneous speech than the existing public speech corpora for Danish, which are largely limited to read-aloud and dictation data. To evaluate the quality of the corpus, Kirkeda et al. trained automatic speech recognition systems on the new resource and compared them to the systems trained on the Danish part of Språkbanken, the largest public ASR corpus for Danish to date. The baseline results showed that they achieved a 14.01 WER on the new corpus. A combination of FT SPEECH with in-domain language data provided comparable results to models trained specifically on Språkbanken, showing that FT SPEECH transfers well to this data set.

The European Parliament Plenary Session (EPPS) [51] corpus focuses on transcribing speeches given in the European Parliament. The word error rates in the experiments reported below were measured on the official 2006 development set, which consists of 3 hours of speech from 41 politicians. The acoustic models were trained on the official EPPS training data which consists of about 100 hours of transcribed speech from politicians and interpreters.

1.2.3 Transcriptions & Alignment

One key factor in speech recognition is obtaining accurately transcribed speech training data. Dealing with inadequate labelled training data is a lasting obstacle in speech recognition research. Due to the amount of data that is available and the absence of imperfect alignment between the two modalities, this is a technical challenge. To this end, utilizing approximate transcripts is definitely not a new topic. It has been demonstrated before how captioned multimedia speech can be transformed into a speech corpus.

Witbrock et al. in 1998 described a reliable unsupervised method for identifying precisely transcribed sections of television broadcasts, and showed how these segments could be used to train a recognition system [167]. Starting from acoustic models trained on the Wall Street Journal database, a single iteration of their training method reduced the word error rate on an independent broadcast television news test set from 62.2% to 59.5%. In 1999, Jang et al. followed a similar path extracting 131.4 hours of transcribed speech, improving the word error rate of their best speech recognition system (Sphinx-III) from 32.82% to 31.19% [71].

The transcripts of speech recordings must be in sync with the actual speech recordings in order to be used as resources. Performing forced alignment of text is crucial to the performance of speech recognition systems [108, 109]. In forced alignment, given a method to map graphemes to phonemes (normally a pronunciation lexicon) and a statistical model of how phones are realized, speech and its associated orthographic transcription are automatically aligned at the word and phone level. There are a number of well-known methodologies for carrying out this alignment based on different architectures, including using an existing ASR system to automatically transcribe the speech recordings, aligning the automatic transcripts with the human transcripts already in existence, and obtaining alignments between speech recordings and human transcripts as a result [52, 77, 81, 99, 136].

1.2.4 Greek Language Technology and Speech Corpora

The complexity of the Greek language brought on by its numerous inflectional rules is one of the key challenges in Greek automatic speech recognition. As a result, many efforts have been made in Greek language processing and modeling, but only a small number of works report extensive results in Large Vocabulary Continuous Speech Recognition (LVCSR) problems, which involve a number of other concerns with regard to feature extraction, acoustic modeling, and recognition methods.

In order to increase productivity in the field of journalism, Digalakis et al. created the first Greek dictation system, Logotypografos 1.0, and Greek voice corpus, Logotypografia Database, in 2003. Through the use of genomic HMMs with adjustable levels of Gaussian sharing across various HMM states [30], the research team was able to reach 19.27% word error rate. A slight but statistically significant enhancement of 0.28% in WER was made in [115] by using a maximum entropy language model that included n-gram and stem constraints. Detailed information about Logotypografia can also be found in section 5.2.2.

Instead of focusing on dictation, the work conducted in [135] provided corpora in the domain of broadcast news. After capturing numerous news shows broadcasted via the Greek satellite-television channel ERT, they evaluated several ASR systems and obtained approximately constant word error rates of about 38%. Along the same vein, Dimitriadis et al. proposed GRIDNEWS [31], a distributed system storing and retrieving broadcast News data recorded from the Greek television. The experiments on the Large-Vocabulary Speech Recognition system (Greek LV-ASR) was based on the statistical Hidden Markov Models framework, i.e. the Hidden Markov Model Toolkit (HTK) platform and managed to obtain a WER of 38.42%, similarly to the aforementioned work.

The need to address the obstacle of the unavailability of language resources was examined in [131] in a novel zero-resourced ASR approach to train acoustic models that only uses list of probable words from the target language. The Kullback-Leibler divergence based Hidden Markov model (KL-HMM), which has demonstrated to be particularly helpful to design ASR systems for new and under-resourced languages, served as the basis for this methodology [68,69,96,130]. For the evaluation of the suggested methodology Greek was used as the target zero-resource language using the SpeechDat(II) database [64]. Their research showed that the suggested method made it possible to create ASR systems in a configuration with limited resources and no supervision. The KL-HMM system resulted in word error rate of 43.0% by utilizing solely the word list from the target Greek language. Additionally, WER of 27.7% was accomplished via unsupervised adaption of KL-HMM parameters and trigraph modeling. By the same token, Jimseng et al. have demonstrated that a KL-HMM system can match or even surpass modern state-of-the-art speech recognition technologies for an under-resourced language even with a very limited amount of training data [67]. The performance of the KL-HMM system resulted in a 77% word accuracy with just five minutes of data and word labels. Therefore, they concluded that the KL-HMM framework is suitable for automatic speech recognition for languages with insufficient resources.

The use of ASR approaches in systems and applications is one of the first concerns that the scientific community has grappled with. After pointing out the absence of support for the Greek language in CMU Sphinx [85], Pantazoglou et al. proposed an approach to include the Greek language into the platform via a collection of tools [119]. Using a basic computing system, the language model, voice dictionary, and acoustic model for the generic Greek model were developed. Another research by Mporas et al. [110] assessed

the performance of a Greek speech recognizer in a smart-home setting. For two distinct types of microphones, the operating performance was examined under varied environmental circumstances. The SpeechDat(II)-FDB 5000 corpus served as the foundation for the acoustic modeling. Across all experiments, a task completion rate of 100% was noted, regardless of the variation in word error rates recorded for various conditions. In the robotic applications domain, the grapheme-to-phoneme (G2P) technique was utilized in [156] as a workaround for the lack of a Greek general model in Sphinx-4 [162] to perform automated word recognition. The G2P strategy, which avoids the often harsh acoustic model training, indicated signs of being a handy but rough technique for automatic word recognition in languages that Sphinx-4 does not yet support.

In the research area of medical applications, efforts have been made to incorporate Greek in speech recognition systems. Similar to many medical applications, aphasic systems lack speech data. Greek aphasic discourse corpus development has been attempted for the first time systematically using the GREECAD [159] to foster the multidisciplinary research of aphasia and to spur interest in the linguistic study of Greek aphasia. Chatzoudis et al. followed a different approach to this matter by utilizing the English data that was already available and employing language-agnostic linguistic characteristics. The research group was able to achieve zero-shot aphasia identification in low-resource languages like Greek and French with 81% and 83% detection accuracy respectively.

1.3 Contributions

Our key contributions are summarized as follows:

- We release HParl (v1), a modern and evolving 120h speech corpus for Greek.
- We build a multi-domain speech corpus for Greek and present extensive evaluations for in-domain and out-of-domain performance of Kaldi-based models.
- We incorporate speaker diarization capabilities to Kaldi-gRPC-Server, a modern, pythonic, gRPC-based software tool that allows for easy and performant deployment of Kaldi-based ASR and diarization models.

1.4 Outline of the Dissertation Presentation

The thesis is organized in the following way:

Chapter 2 presents a brief historical overview of automatic speech recognition and analyses several introductory concepts such as the traditional process of constructing ASR systems, the involving methods and algorithms, language modelling, acoustic modelling and Weighted Finite State Transducers (WFST).

Chapter 3 provides an in-depth view of Kaldi, a state-of-the-art automatic speech recognition toolkit.

Chapter 4 presents Kaldi gRPC Server, a modern alternative for deploying Speech Recognition models developed using Kaldi, demonstrates the key features and recommended usage while providing details on the contribution of this thesis, a speaker diarization component.

Chapter 5 describes the essence of the present work, a multi-domain speech recognition approach for the Modern Greek language. We briefly elaborate on the construction of HParl, a dataset comprised of proceedings of the Hellenic Parliament, while providing details on the processing pipeline in order to make it usable and extendable for the ASR research community. We merge HParl and three other datasets characterized by different speech scenarios into a larger single dataset named GREC.

Chapter 6 outlines the experiments of two separate ASR tasks that were conducted in order to evaluate the performance of various models on the GREC dataset and assess the results.

Finally, in **Chapter 7** we conclude with a discussion of this work and provide insights as to which future actions must be executed to extend and advance it.

2. BACKGROUND

2.1 History

In 1952, researchers at Bell Laboratories created one of the early initiatives that might be regarded as an ASR technology. This technology depicted in Fig. 1 was called “AUDREY”, taking a major leap over chores and random sounds as it was capable of actually recognising the distinct sound of a spoken digit by using only analog circuitry.

Figure 1: 1952 - AUDREY - Bell Labs. A six foot high rack of supporting analog-circuitry electronics is not shown.

A few years later in the 1960’s, IBM engineered a new technology called “Shoebox”, as depicted in Fig. 2, which could recognize 16 words spoken in English. Speaking through a microphone, which then turned sounds into electrical impulses, was how this technology was used. By then, it was clear that voice recognition technology was on the path to model human language. In the same decade, filter banks were combined with dynamic programming to produce the first practical recognizers, mostly for words spoken in isolation (i.e. with pause after each word), so as to simplify the task. In Russia, Professor Taras Vintsyuk proposed using dynamic programming techniques—commonly referred to as Dynamic Time Warping (DTW) to temporally align a pair of spoken utterances. By developing a 200-word recogniser, Velichko and Zagoruyko advanced the use of pattern recognition concepts in voice recognition [9] using Vintsyuk’s work [11].

Figure 2: 1961 — Shoebox - IBM video archives

Speech recognition made a number of significant improvements in the 1970s. This was mostly due to the US Department of Defense and Defense Advanced Research Projects Agency (DARPA). In the history of voice recognition, one of the biggest projects was the Speech Understanding Research (SUR) initiative. Carnegie Mellon's "Harpy" speech system [94] emerged from this program and was capable of understanding over $1,000$ words which is approximately the same as a three-year-old's range of vocabulary.

The use of specialized hardware created specifically for commercial small-vocabulary telephone applications advanced significantly throughout the 1970s as well. As an automated and effective way to represent speech, Linear Predictive Coding (LPC) became a prominent ASR representation. LPC is still the standard today in cellphone speech transmissions but for ASR purposed was replaced by the Mel-frequency Cepstral Coefficient (MFCC) approach in the next decade. This period of time also presented the creation of large widely available databases in several languages, allowing comparative testing and evaluation.

Advance	Date	Impact
Linear predictive coding	1969	Automatic, simple speech compression
Dynamic time warping	1970s	Reduces search while allowing temporal flexibility
Hidden Markov models	1975	Treat both temporal and spectral variation statistically
Mel-frequency cepstrum	1980	Improved auditory-based speech compression
Language models	1980s	Including language redundancy improves ASR accuracy
Neural networks	1980s	Excellent static nonlinear classifier
Kernel-based classifiers	1998	Better discriminative training
Dynamic Bayesian networks	1999	More general statistical networks

Figure 3: Major advances in ASR technology. Figure taken from O'Shaughnessy (2008) [116]

Speech recognition vocabulary increased from a few hundred to several thousand words in the 1980s. One of the breakthroughs came to light from a statistical method based on MFCCs known as the Hidden Markov Model, which has remained the dominant ASR methodology since then. In brief, instead of just using words and looking for sound patterns, the HMM estimated the probability of the unknown sounds actually being words.

That said, rather than attempting to get computers to resemble the way humans digest language, researchers began utilizing statistical models to enable artificial interpretation of speech patterns. The majority of the ASR technology used today is derived from this original model, despite the fact that the original technology was not very precise and efficient.

During the 1990s, speech recognition was significantly advanced in large part due to the personal computer. From isolated-word dictation systems to all-purpose continuous-speech systems, commercial applications have emerged. The use of ASR in software has become widespread since the mid-1990s. Medical reporting and legal dictation have been two dominant applications, as well as automation of services to the public over the telephone network. The vocal portal (VAL), a dial-in interactive speech recognition system (IVR), was introduced by Bell-South. Numerous phone tree systems that are still in use today were inspired by this design. This gave rise to highly expensive ASR technologies being sold during the 90's which eventually became more accessible and affordable during the technology boom in the 2000's.

By the year of 2001, speech recognition technology had achieved close to 80% accuracy. For most of the decade there were not many major signs of advancements until Google changed that with the release of Google Voice Search. The mobile nature of this application placed speech recognition into the hands of millions of people. It was also significant because the processing power could be offloaded to its data centers. Coupled with that, Google was collecting data from billions of searches which could help approxiamate speech recognition by predicting what a person is actually saying. At the time Google's English Voice Search System contained 230 billion words from user searches.

In 2011, Apple released Siri which was really similar to Google's Voice Search. The early part of this decade witnessed an explosion of other voice recognition apps as well. With Amazon's Alexa and Google Home consumers become more and more comfortable talking to voice assistants.

Today, some of the largest tech companies are competing to acquire the speech accuracy title. In 2016, IBM achieved a word error rate of 6.9%. In 2017, Microsoft dethroned IBM with a 5.9% score. Shortly afterwards, IBM improved their rate to 5.5%. However, it is Google that is holding the lowest rate at approximately 4.9%.

ASR technologies are constantly improving in terms of accuracy, speed, and cost. The need for humans to check the accuracy of these technologies is decreasing taking consideration of the data availability and the accessibility of ASR technology across all industries is spreading. The value of this technology is being constantly explored by universities, enterprises and many other organizations. What first started as a means to recognize numerical digits has now developed into a highly advanced system of recognizing hundreds of languages and accents in real-time.

2.2 Procedure

A typical ASR system is comprised of multiple components as seen in Fig. 41. The acoustic model creates representations of the sounds of the language, the lexicon describes how these sounds relate to construct words, and the language model demonstrates how these words are formed into sequences of words. The entire acoustic model is defined by taking the lexicon pronunciation HMM and adding to it the cepstral feature vectors computed from the raw audio file. Collectively, the acoustic model is utilized to measure $P(O|W)$, the likelihood of an audio observation given a word. When combined with a language model, the prior probability of a word, $P(W)$, can be estimated.

Utilizing these two models in conjunction, a decoding procedure can be executed using the Baum-Welch [127] and Viterbi [43] algorithms to compute the posterior probability $P(W|O)$ which provides the probability of any sequence of words given the observation of the raw audio file. By selecting the sequence that maximizes this probability, speech recognition is essentially performed. The following sections explain thoroughly the properties and behaviour of each individual component.

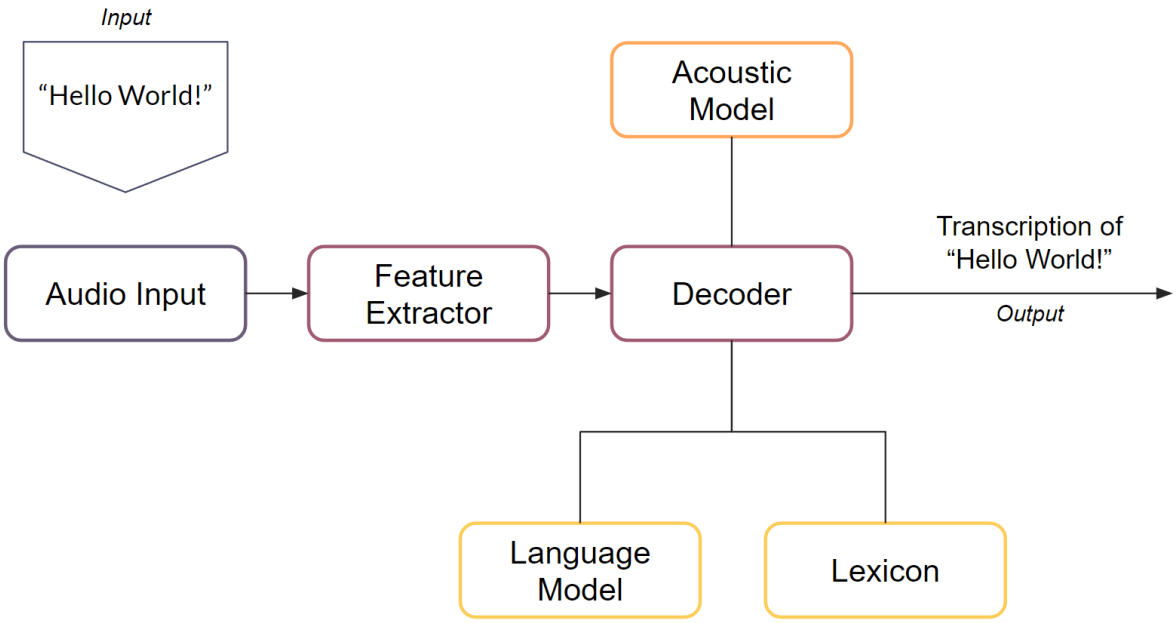

Figure 4: A traditional Automatic Speech Recognition pipeline

2.2.1 Feature Extraction

At the basis of automatic speech recognition lies the process of feature extraction which is the calculation of a series of feature vectors that results in a compact representation of the input speech signal. Feature extraction requires much attention since recognition performance relies heavily on the feature extraction phase.

This is usually performed in three steps. The first one is known as the speech analysis or the acoustic front-end, which analyzes the speech stream spectrally and temporally and produces raw characteristics describing the envelope of the power spectrum of short speech intervals. The second stage compiles an extended feature vector composed of static and dynamic features. These expanded feature vectors are then converted in the last stage into more compact and reliable vectors, which are subsequently provided to the recognizer.

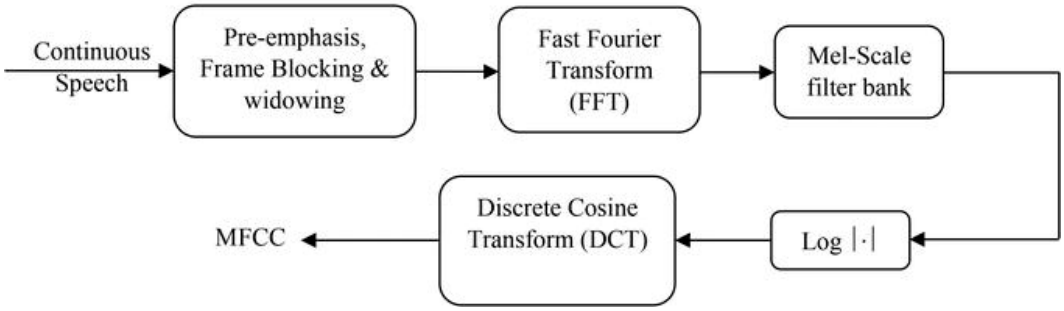

Figure 5: Mel-frequency Cepstral Coefficients feature extraction process.

MFCC [139, 168], LPC [9, 26], Linear Prediction Cepstral Coefficients (LPCC), Line Spectral Frequencies (LSF), Perceptual Linear Prediction (PLP) [60] and Discrete Wavelet Transform (DWT) [113] are a few of the feature extraction methods that are employed to extract data from speech signals in order to recognize and identify speech. These techniques are often used in speech recognition systems for a number of purposes since they have long been proved effective. Table 1 shows a comparison between their properties. The following subsections describe some of the most commonly used methods for feature extraction in speech recognition.

Table 1: Comparison between the feature extraction techniques [3].

Method	Filter	Filter Shape	Target Model	Computation Speed	Coefficient Type	Noise Resistance	Sensitivity to quantization/noise	Reliability	Frequency Captured
MFCC	Mel	Triangular	Human Auditory System	High	Cepstral	Medium	Medium	High	Low
LPC	Linear Prediction	Linear	Human Vocal Tract	High	Autocorrelation Coefficient	High	High	High	Low
LPCC	Linear Prediction	Linear	Human Vocal Tract	Medium	Spectral	High	High	Medium	Low&Medium
LSF	Linear Prediction	Linear	Human Vocal Tract	Medium	Spectral	High	High	Medium	Low&Medium
DWT	Lowpass/Highpass	-	-	High	Wavelets	Medium	Medium	Medium	Low&High
PLP	Bark	Trapezoidal	Human Auditory System	Medium	Cepstral&Autocorrelation	Medium	Medium	Medium	Low&Medium

2.2.1.1 Mel-Frequency Cepstral Coefficient (MFCC)

The most prevalent and dominant method used to extract spectral features is calculating Mel-Frequency Cepstral Coefficients. MFCCs are one of the most popular feature extraction techniques used in speech recognition based on frequency domain using the Mel scale which is based on the human ear scale. MFCCs being considered as frequency domain features are much more accurate than time domain features. Mel-Frequency Cepstral Coefficients is a representation of the real cepstral of a windowed short-time signal derived from the Fast Fourier Transform (FFT) of that signal. The difference from the real cepstral is that a nonlinear frequency scale is used, which approximates the behaviour of the auditory system. Additionally, these coefficients are robust and reliable to variations according to speakers and recording conditions. MFCC is an audio feature extraction technique which extracts parameters from the speech similar to ones that are used by humans for hearing speech, while at the same time, de-emphasizes all other information. The speech signal is first divided into time frames consisting of an arbitrary number of samples. In most systems overlapping of the frames is used to smooth transition from frame to frame. Each time frame is then windowed with Hamming window to eliminate discontinuities at the edges. The filter coefficients $w(n)$ of a Hamming window of length n are computed according to the formula:

$$W(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{2.1}$$

After the windowing, FFT is calculated for each frame to extract frequency components of a signal in the time domain. FFT is used to speed up the processing. The logarithmic Mel-Scaled filter bank is applied to the Fourier transformed frame. This scale is approximately linear up to 1kHz, and logarithmic at greater frequencies. The relation between frequency of speech and Mel scale can be established as:

$$\text{Mel Scaled Frequency} = [2595log(1 + f(Hz)/700] \tag{2.2}$$

MFCCs use Mel-scale filter bank where the higher frequency filters have greater bandwidth than the lower frequency filters, but their temporal resolutions are the same. The last step is to calculate Discrete Cosine Transformation (DCT) of the outputs from the filter bank. DCT ranges coefficients according to significance, whereby the $0th$ coefficient is excluded since it is unreliable. DCT is defined by the following, where is a constant dependent on N:

$$X_k = \alpha \sum_{i=0}^{N-1} x_i \cos\left\{\frac{(2i+1)\pi k}{2N}\right\} \tag{2.3}$$

The result of the last step is called Mel Frequency Cepstrum Coefficient. The set of coefficients is called acoustic vectors. Therefore, each input utterance is transformed into a sequence of acoustic vectors. The MFCC extraction process can be seen in Fig. 5.

2.2.1.2 Linear Predictive Coding (LPC)

One of the most effective speech analysis techniques is Linear Predictive Coding, which may be used to efficiently encode speech at low bit rates. The LPC technique's fundamental aspect is that a voice sample may be effectively represented as a linear combination of prior speech samples. LPC is a frame based analysis of the speech signal which is performed to provide observation vectors of speech. The input speech signal digitized spectrally flatten speech signal is put through a low order digital system to make it less susceptible to finite precision effects later in the signal processing.

To compute LPC features, initially the speech signal is blocked into frames of N samples. The output of the pre-emphasizer network is related to the input to the network. After frame blocking, the next step is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. Typical window is the Hamming window. The next step is to auto correlate each frame of windowed signal Where the highest auto-correlation value is the order of the LPC analysis. The next processing step is the LPC analysis, which converts each frame of auto-correlations into LPC parameter set by using Durbin's method [16]. LPC cepstral coefficients, is a very important LPC parameter set, which can be derived directly from the LPC coefficient set. A block diagram of LPC feature extraction process is depicted in Fig. 6.

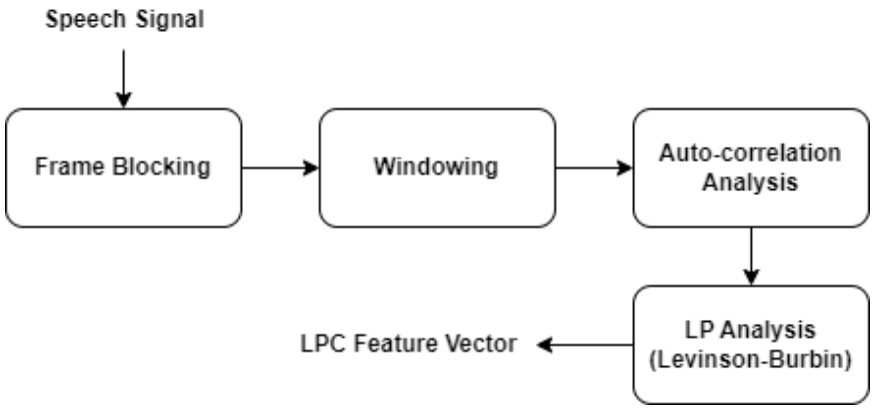

Figure 6: LPC speech analysis pipeline

2.2.1.3 Perceptual Linear Prediction (PLP)

The Perceptual Linear Prediction model was developed by Hermansky in 1990 [61]. The purpose of PLP is modelling accurately the human speech based on the field of psychophysics of human hearing [158]. PLP analysis, like LPC analysis, is based on the short-term spectrum of speech. Contrary to pure linear predictive analysis of speech, perceptual linear prediction alters the short-term spectrum of the speech by a number of psychophysically based transformations.

The Bark-spaced filter-bank of 18 filters is used by the PLP parameters to span the frequency range $[0, 5000]\ Hz$. The PLP coefficients are calculated exactly as follows:

- The N-point DFT is applied to the input signal $x(n)$ in the discrete time domain.
- The critical-band power spectrum is computed through discrete convolution of the power spectrum with the piecewise approximation of the critical-band curve, where the Bark warped frequency is obtained through the Hertz-to-Bark conversion.
- Equal loudness pre-emphasis is applied on the down-sampled.
- Intensity-loudness compression is performed.
- The output is linearly interpolated to give interpolated auditory spectrum.
- An inverse DFT is performed to obtain the equivalent auto-correlation function.
- The PLP coefficients are computed after auto-regressive modeling and conversion of the auto-regressive coefficients to cepstral coefficients.

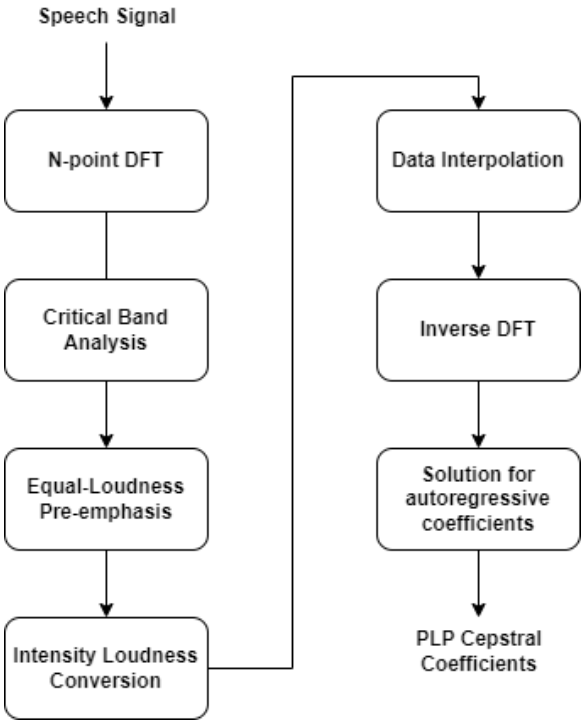

Figure 7: PLP speech analysis pipeline

2.2.2 Acoustic Model

The acoustic model (AM) is the main component of an ASR system which holds most of the computational load and performance of the system. The acoustic model's primary responsibility is to identify which sound, or phoneme, from the phone set is being uttered in each audio frame. Its creation involves the use of audio recordings of speech and their text scripts, splitting them into small segments or frames and then compiling them into a statistical representation of sounds that describe words.

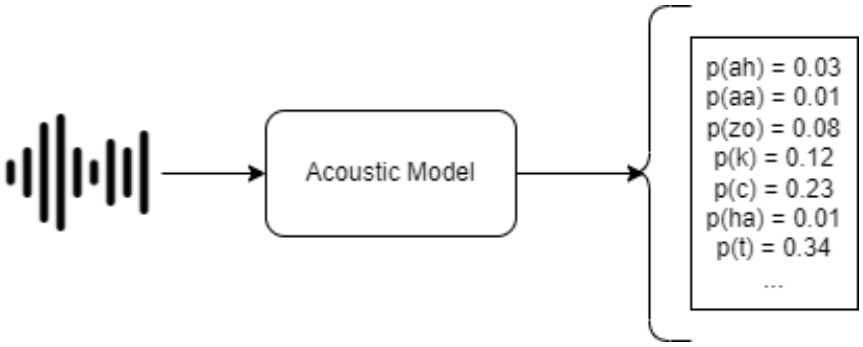

Figure 8: Probabilities of spoken phonemes in short audio frames generated by an AM

Deep neural networks trained on thousands of hours of transcribed audio data are a popular choice for acoustic models. The key to making sure the acoustic model performs effectively for various acoustic properties is having the appropriate data for training and testing. The training data must represent the features of the acoustic model, which models factors like accent, gender, age, microphone, and background noise.

Speaking style is a different, less evident component that affects the acoustic model. Compared to when speaking casually with a human, people are more likely to speak clearly if they are aware that they are speaking to a computer. The acoustic model includes this difference in enunciation as additional feature.

2.2.3 Language Model

A language model (LM) is a probabilistic statistical model that estimates the likelihood that a particular series of words will appear in a phrase depending on the words that came before it. [76]. It is usually an n-gram model or a neural network trained on millions of words of text data.

A training collection of sample sentences is often the input for a language model, and the result is a probability distribution across word sequences. Predicting the next word of a sequence of words can be achieved by using the last one word (unigram), last two words (bigram), last three words (trigram) or last n words (n-gram). Each conversation's topic greatly influences the words and phrases used, but it may also be impacted by other aspects such as age, gender, formality, speaking style, and others. The language model training data should reflect the kinds of words and phrases users will say to the final system.

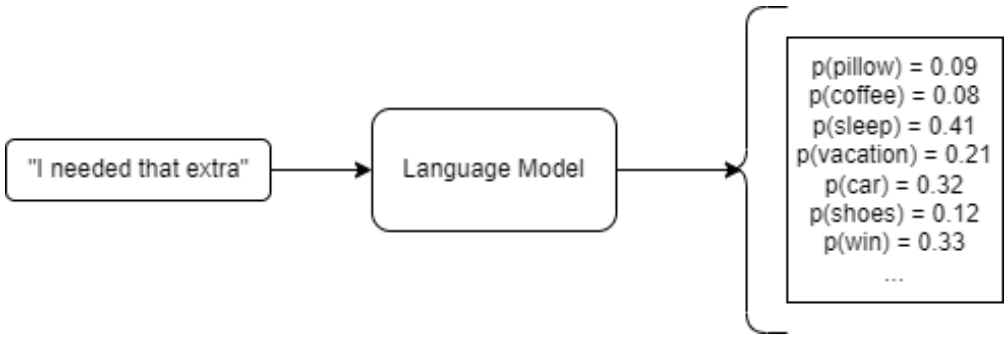

Figure 9: Probabilities of upcoming words generated by a language model

2.2.4 Lexicon

A key part of any automatic speech recognition system is the lexicon. The definition of lexicon can be altered for the reason that it can mean different things depending on the context. In its most standard form, a lexicon is a dictionary of words with their pronunciations broken down into phonemes, i.e. units of word pronunciation.

One of the most popular resources for creating a lexicon is the Carnegie Mellon University (CMU) Pronouncing Dictionary [166], often abbreviated as CMUdict. This resource contains four files: a set of 39 phone (phoneme) symbols as defined in ARPABET [82], a file containing the assignment of these symbols to their type, such as “vowel”, “fricative”, “stop”, or “aspirate”, a file giving pronunciations of various punctuation marks, and finally the core dictionary giving pronunciations of standard English words, deconstructed by phone.

Table 2: Example mappings from words to their pronunciations in the ARPABET phoneme set

Word	Pronunciation
mine	m ay n
sun	s ah n
freeze	f r iy z
nominate	n aa m ah n ah t
waist	w ey s t
control	k ah n t r ow l
reinforce	r iy ih n f ao r s

Another way that the word lexicon is used is to refer to the Finite State Transducer (FST) which is generated by the lexicon preparation and sometimes referred to as “L.FST”. A finite state automaton that combines two sets of symbols is known as a finite state transducer. In the case of the lexicon, such a transducer maps the word symbols to their respective pronunciations. More details can read in section 2.6.4.

2.3 Methods & Algorithms

2.3.1 Hidden Markov Models

Hidden Markov models, named after the Russian mathematician Andrey Andreyevich Markov who developed Markov chains and laid the groundwork for the relevant statistical theory, were introduced and advanced in the early 1970s by groups at Carnegie Mellon University and IBM who introduced the use of discrete density HMMs [72, 114], and, later on, the use of continuous density HMMs at Bell Labs [74, 75, 88]. Since then, they have served as the foundation for numerous continuous speech recognition systems [45].

HMMs are statistical models that can extract hidden information from sequential symbols that can be observed.The goal of an HMM is to extract the hidden parameters from the observable parameters, where the system being modelled is assumed to be a Markov process with unknown parameters. A robust HMM accurately models the real world source of the observed real data and is capable of simulating the source. A lot of machine learning techniques that are mainly based on HMMs and have been traditionally used in signal processing and speech recognition, are now an essential part of optical character recognition and computational biology while they have become a fundamental tool in bioinformatics due to their robust statistical foundation, remarkable simplicity and elasticity [34, 89].

Four factors are associated with an HMM:

- H - Set of hidden states
- V - Set of visible states
- a_{ij} – Transition probabilities corresponding to the visible states
- b_{jk} – Emission probability of the visible state from the hidden states

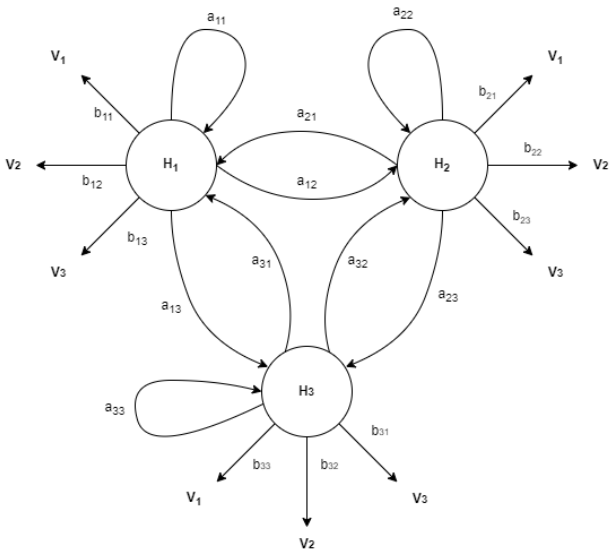

Figure 10: Probabilistic parameters of a Hidden Markov model H — Hidden states V — Possible observations a — State transition probabilities b — Output probabilities

In Fig. 10, H_i are the hidden states and V_j are the visible states. In any hidden state the model may emit any one of the visible states. The machine has the option of moving between states or staying put in one state. For example, a_{33} is the transition probability of H_3 remaining in the same state. Generally, the state transition is shown as:

$H_i(t-1) -> H_j(t)$

Again b_{11} is the emission probability of the hidden state H_1 if it emits visible state V_1. We can say that b_{jk} is the state emission probability of the visible state V_k if the machine is in state H_j and can be written as

$P(V_k \mid H_j) = b_{jk}$

Essentially, having analyzed the above, it is concluded that at any state the machine emits a state transition probability to any other state:

$$\sum_{j} a_{ij} = 1; \quad \forall i \tag{2.4}$$

and at any state the machine emits a state emission probability to any other visible state:

$$\sum_{k} b_{jk} = 1; \quad \forall i \tag{2.5}$$

The Hidden Markov model in figure 11 represents the process of predicting whether someone will be found to be walking, driving, or sleeping on a particular day depending upon whether the day is rainy or sunny.

The properties of this Hidden Markov model example are the following:

- states = ('Rainy', 'Sunny')
- observations = ('Walk', 'Drive', 'Sleep')
- start probability = {'Rainy': , 'Sunny':}
- transition probability = {
  'Rainy': {'Rainy': 0.5,'Sunny': 0.5},
  'Sunny': {'Rainy': 0.4,'Sunny': 0.6},
  }
- emission probability = {
  'Rainy': {'Walk': 0.2,'Drive': 0.3, 'Sleep': 0.5},
  'Sunny': {'Walk': 0.5,'Drive': 0.4, 'Sleep': 0.1},
  }

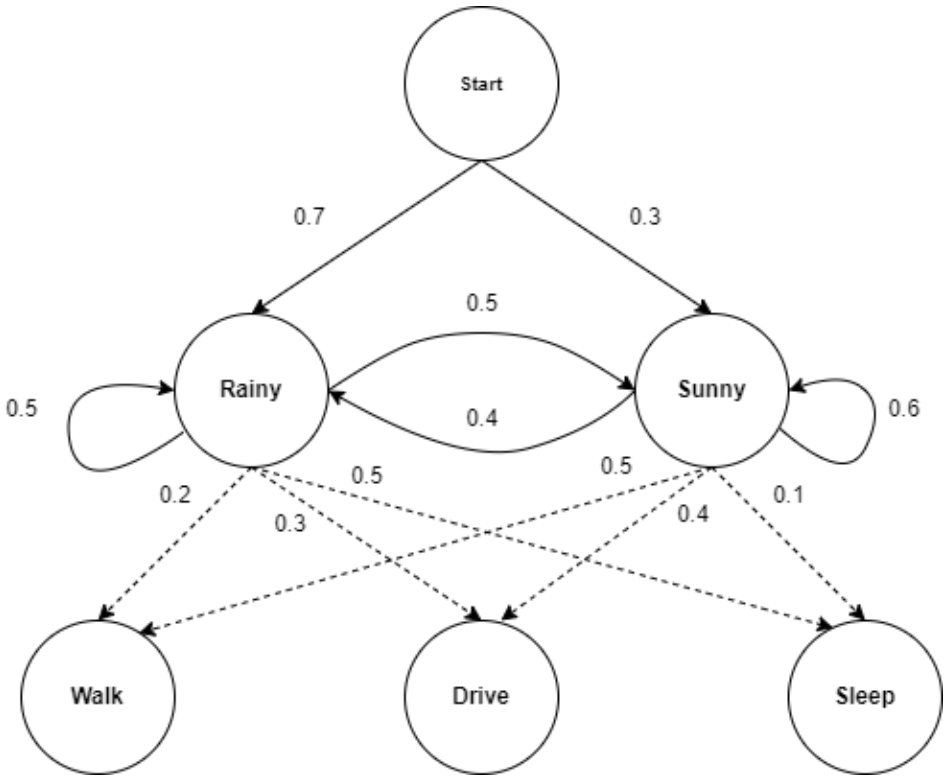

Figure 11: A Hidden Markov model example

An initial probability is represented by start probability on the top node. Two hidden states follow, in this case rainy and sunny. These states are hidden because what is observed as the process output is whether the person is walking, driving, or sleeping. The series of observations is walk, drive and sleep. Transition probability represents the transition of one state (rainy or sunny) to another state given the current state. Emission probability represents the probability of observing the output, walk, drive and sleep given the states, rainy or sunny.

Three basic problems can be solved with Hidden Markov Models:

- Given the Hidden Markov Model λ and a sequence of observations O, finding the probability of an observation $P(O \mid \lambda)$. This is called the **Evaluation Problem** and is implemented using **Forward** algorithm as explained in section 2.3.1.2.
- Given the Hidden Markov Model λ and an observation sequence O, finding the most likely state sequence $(s_1, s_2 ... s_n)$. This is called a **Decoding Problem** and is implemented using the **Viterbi** algorithm as explained in section 2.3.1.3.
- Finding an observation sequence $(O_1, O_2 ... O_n)$ and Hidden Markov Model λ that maximizes the probability of O. This is called a **Learning Problem** or **Optimization Problem** and is implemented using the **Baum-Welch** algorithm as explained in section 2.3.1.4.

2.3.1.1 HMM in Speech Recognition

The foundation of a number of effective methods for acoustic modeling in speech recognition systems is the Hidden Markov model. Considering a simple word recognizer, the purpose of this system would be to find the most probable sequence of phonemes given a sequence of speech signal segments. The most likely word for a speech signal may be determined using HMMs since a group of phonemes can be mapped to a word. Observations in the domain of speech recognition are the segments of spoken speech signal. Hidden states are the sequence of phonemes that are meant to be recognized.

As each word in the corpus must have its own HMM, a phoneme-based word HMM model cannot scale for identifying continuous speech. For successful recognition of continuous speech it is required to utilize contextual triphone based sub-word HMMs. In the case of a phoneme based HMM for the word ‘dog’ would have /d/ /ao/ and /g/ as states as seen in Fig. 12. In this approach, it is required to create a HMM for every word in the corpus and train it to with the utterances of the word to enhance the model accuracy. The accuracy of speech recognition is, undeniably, largely shaped by how effectively the HMMs were trained.

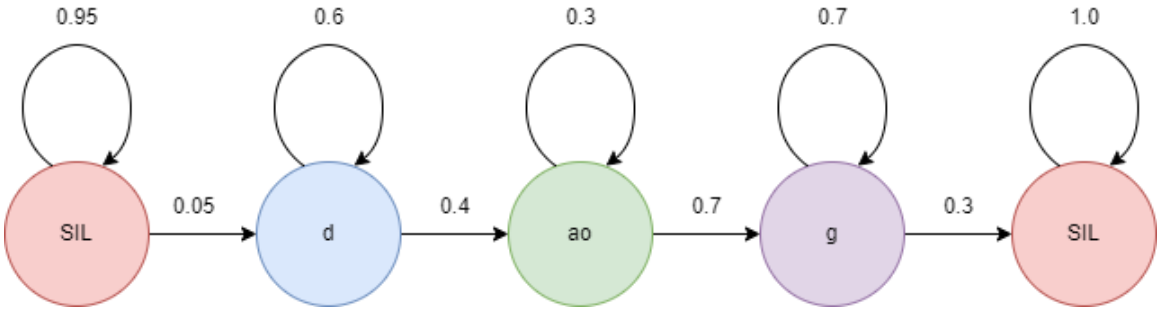

Figure 12: Phoneme based Hidden Markov model

2.3.1.2 Forward Algorithm & Evaluation Problem

The evaluation problem is defined as finding $P(O \mid \lambda)$, given the observation sequence $O = O_1, O_2, O_3, ..., O_T$. The most straight forward way to find the solution is enumerating every possible state sequence of length T. Consider one such state sequence $Q = q_1, q_2, q_3, ..., q_T$ such that q_1 produces O_1 with some probability, q_2 produces O_2 with some probability and so on. so using chain rule.

$$P(O \mid \lambda) = \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q1}(O_1)\, a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{2.6}$$

but the order of chain rule is N^T, since at every $t = 1, 2, ..., T$, there are N possible states which can be reached. This is clearly and inefficient algorithm, to over come this forward algorithm is used.

Forward Algorithm: Forward algorithm is a dynamic algorithm which uses forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(O_1, O_2, ..., O_i, q_t = S_i \mid \lambda) \tag{2.7}$$

i.e., the probability of partial observation sequence, $O_1, O_2 ... O_t$ and state S_i at time t given the model λ, as we are taking partial observation in to account, we can solve $\alpha_t(i)$ inductively as:

Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \tag{2.8}$$

Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N \tag{2.9}$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.10}$$

since $\alpha_t(i)$ is the probability of partial observation sequence O_1 to O_t and the state at time t is S_i, then the product $\alpha_i(t)\alpha_{ij}$ is the probability of joint event that the observation O_1 to O_t are observed and state S_j is reached at time $t+1$ via state S_i at time t. Summing over all N possible state $S_i(1 \leq i \leq N)$, at time t, results in the probability of S_j at time $t+1$. By multiplying this quantity with $b_j(O_t+1)$, we can find $\alpha_{t+1}(j)$. This computation is performed for all state $j(1 \leq j \leq N)$ for a given t and it is iterated through $t = 1, 2, 3, ..., T-1$. Finally, the required $P(O \mid \lambda)$ is sum of the terminal forward variables $\alpha_T(i)$, this is true because:

$$\alpha_T(i) = P(O_1, O_2, ..., O_T, q_T = S_i|\lambda) \tag{2.11}$$

2.3.1.3 Viterbi Algorithm & Decoding

For the decoding problem, there are various way in which optimum solution can be calculated. The difficulty lies in the definition of optimum state sequence. One way to find optimum state sequence is to choose the states q_t which are individually most likely. But this has serious flaw in sense that if two states i and j are selected such that $a_{ij} = 0$, then despite of most likely state at time t and $t + 1$, it is not valid state sequence. Therefore, deciding optimum criteria is crucial.

There is a way to locate the best state sequence by using dynamic programming, such as the Viterbi algorithm [43]. The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states called the Viterbi path. The latter results in a sequence of observed events, especially in the context of Markov information sources and Hidden Markov models.

Viterbi Algorithm: In order to find single best state sequence, $Q = q_1, q_2.q_3, ..., q_t$, (which produces given observation sequence) for a given observation sequence $O = o_1, o_2, o_3, ..., o_t$, we define a quantity:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P\left[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t \mid \lambda\right] \tag{2.12}$$

i.e., $\delta_t(i)$ is the best score along a single path, at time t, which accounts for the first t observations and ends in state S_i, by induction:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j\left(O_{t+1}\right) \tag{2.13}$$

In order to find the state sequence we need to keep track of the states which maximize the above equation. We do this with the help of an array $\psi_t(j)$ for each t and stat j. Once the final state is reached, the corresponding state sequence can be found out using backtracking. The Viterbi algorithm is similar in implementation to the Forward algorithm, except for the backtracking part. The major difference is maximization of the previous state in place of summing procedure in forward calculation.

2.3.1.4 Baum-Welch Algorithm & Learning

The most challenging of all problems is to adjust the model parameter (A, B, λ) to maximize the probability of the observation sequence given the model. Given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. However, there are some heuristic procedures which attempt to find local optimization over global optimization.

Baum-Welch Algorithm: Also known as the forward-backward algorithm, the Baum-Welch algorithm is a dynamic programming approach and a special case of the expectation-maximization (EM) algorithm computing more than the Viterbi algorithm [100]: it computes the probability of being in any given state at any given moment, summed over all conceivable pathways that pass through that state at the given time. It does not take into account just one path rather than all possible paths.

In order to define a procedure for re-estimation of HMM parameter, the following definition is introduced:

$$\xi_t(i,j) = P\left(q_t = S_i, q_{t+1} = S_j \mid O, \lambda\right) \tag{2.14}$$

i.e.,the probability of being in state S_i at time t, and state S_j at time $t+1$ given the model and observation sequence:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j\left(O_{t+1}\right) \beta_{t+1}(j)}{P(O \mid \lambda)} \tag{2.15}$$

where $\beta_t(j)$ = backward variable , i.e., probability of the partial observation sequence $t+1$ to end (T), given state S_i at time t and model. Now we define $\gamma_t(i)$ as the probability of being in state S_i at time t, given the observation sequence and model; thus we can associate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over j, such that:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.16}$$

If we sum $\gamma_t(i)$ over time we are presented with a quantity which can be interpreted as expected number of times state S_i is visited, or expected number of transition made from

S_i. Similarly summation of $\xi_t(i,j)$ over time can be interpreted as the expected number of transition made from state S_i to state S_j. That is:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions form } S_i \text{ to } S_j$$

The aforementioned formula provides a mechanism for re-estimating an HMM parameter.

$$\bar{\pi} = \text{Expected number of times in state } S_i \text{ at time } (\text{t} = 1) = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transition from state } S_i \text{ to } S_j}{\text{expected number of transition form state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$\frac{\sum_{\substack{t=1 \\ s.t \\ O_t=V_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

2.3.2 Gaussian Mixture Models

A probabilistic model called a Gaussian mixture model assumes that all of the data points were produced by combining a limited number of Gaussian distributions with unknown parameters. Mixture models may be seen as a generalization of k-means clustering to include details on the covariance structure of the data as well as the locations of the latent Gaussian centers. The iterative Expectation-Maximization technique or Maximum A Posteriori (MAP) estimation from a well trained prior model are employed to estimate GMM parameters from training data.

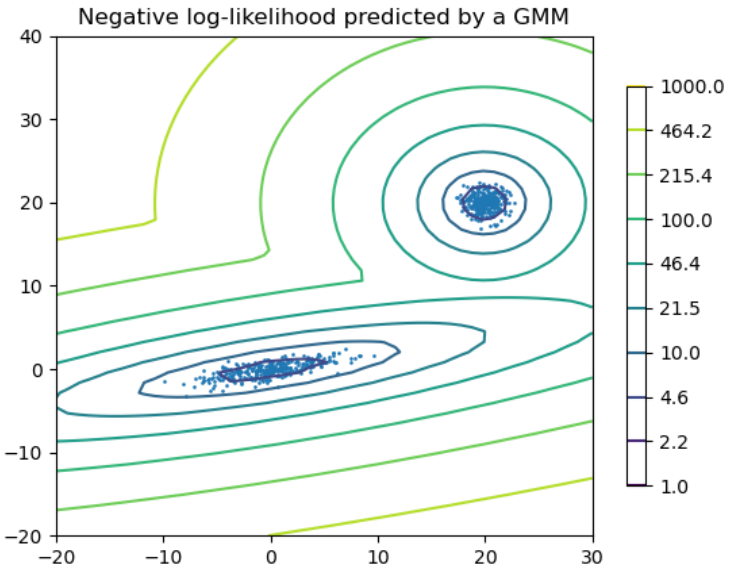

Figure 13: Two-component Gaussian mixture model: data points, and equi-probability surfaces of the model.

In speech recognition scenarios the acoustic events are usually modeled by Gaussian probability density functions (PDFs), described by the mean vector and the covariance matrix. However, unimodel PDF with only one mean and covariance are unsuitable to model all variations of a single event in speech signals. As a result, the complex structure of the density probability is modelled using a mixture of single densities. For a D-dimensional feature vector denoted as x_t, the mixture density for speaker λ is defined as weighted sum of M component Gaussian densities as given by the following:

$$P\left(x_t \mid \lambda\right) = \sum_{i=1}^{M} w_i P_i\left(x_t\right) \tag{2.17}$$

where w_i are the weights and $P_i(x_t)$ are the component densities. Each component density is a D-variate Gaussian function of the form:

$$P_i\left(x_t\right) = \frac{1}{\left(2\pi\right)^{D/2}\left|\Sigma_i\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\left[\left(x_t-\mu_i\right)'\Sigma_i^{-1}\left(x_t-\mu_i\right)\right]} \tag{2.18}$$

where μ_i is a mean vector and Σ_i covariance matrix for $i_t h$ component. The mixture weights have to satisfy the constraint:

$$\sum_{i=1}^{M} w_i = 1 \tag{2.19}$$

The complete Gaussian mixture density is parameterized by the mean vector, the covariance matrix and the mixture weight from all component densities. These parameters are collectively represented by:

$$\lambda = \left\{w_i, \mu_i, \Sigma_i\right\}; \quad i = 1, 2, \ldots . M \tag{2.20}$$

2.3.2.1 Parameters Estimation

With the use of Maximum Likelihood (ML) estimation, the parameters of a GMM model may be obtained. The main objective of the ML estimation is to derive the optimum model parameters that can maximize the likelihood of GMM. The likelihood value is, however, a highly non-linear function in the model parameters and direct maximization is not feasible. Instead, maximization is done through iterative procedures. The iterative EM algorithm is the most frequently used of the several methods created to maximize the likelihood value [24].

The EM algorithm begins with an initial model and tends to estimate a new model such as that the likelihood of the model increasing with each iteration. The process is repeated until a specific convergence threshold is reached or a specified number of iterations have been completed. In the following iteration, this new model is regarded as the original model. A summary of the various steps followed in the EM algorithm are described below.

1. Initialization: In this step an initial estimate of the parameters is obtained. The performance of the EM algorithm depends on this initialization. Generally, the Linde–Buzo–Gray algorithm (LBG) [1] or K-means algorithm [93] is employed to initialize the GMM parameters.

2. Likelihood computation (the E step): In each iteration the posterior probabilities for the i_th mixture is computed as:

$$\Pr\left(i \mid x_t\right) = \frac{w_i P_i\left(x_t\right)}{\sum_{j=1}^{M} w_j P_j\left(x_t\right)} \tag{2.21}$$

3. Parameter update (the M step): Having the posterior probabilities, the model parameters are updated according to the following expressions:

Mixture weight update:

$$\overline{w_i} = \frac{\sum_{i=1}^{T} \Pr\left(i \mid x_t\right)}{T}. \tag{2.22}$$

Mean vector update:

$$\overline{\mu_i} = \frac{\sum_{i=1}^{T} \Pr\left(i \mid x_t\right) x_t}{\sum_{i=1}^{T} \Pr\left(i \mid x_t\right)} \tag{2.23}$$

Covariance matrix update:

$$\bar{\sigma}_i^2 = \frac{\sum_{i=1}^{T} \Pr(i \mid x_t) |x_t - \overline{\mu_i}|^2}{\sum_{i=1}^{T} \Pr(i \mid x_t)} \tag{2.24}$$

The MAP estimation method can be used to estimate GMM parameters in addition to the EM methodology [111]. For instance, speaker identification systems employ MAP estimation to create a speaker model by modifying a Universal Background Model (UBM) [134]. It is also employed in other pattern recognition applications where a previous, generic model is modified using a small amount of labeled training data.

The MAP estimation technique includes two stages, just like the EM algorithm. Similar to the “Expectation” step of the EM method, the first step involves computing estimates of the training data’s sufficient statistics for each mixture in the prior model. These ”new” sufficient statistic estimations are then blended for adaptation with the ”old” sufficient statistics from the previous mixture parameters using a data-dependent mixing coefficient, in contrast to the second phase of the EM process. In line with the design of the data-dependent mixing coefficient, mixtures with a greater amount of new data rely more on the new sufficient statistics for final parameter estimation than mixtures with a lower amount of new data do.

2.4 Language Modeling

In general, the acoustic component of speech recognition systems generates a series of phonemes or phonetic segments that can be a set of hypotheses, which correspond to text that the system has recognized. The set of words contained in the lexicon is compared to the sequence of symbols produced by the acoustic component during recognition in order to determine the best set of words that will make up the system's final output. It is crucial to introduce rules at this stage that can specify linguistic constraints in the language and allow for a decrease in the number of potentially viable phoneme sequences. This is achieved with the usage of a language model. A language model is made up of two major parts: the vocabulary, which is a collection of words that the system can identify, and the grammar, which is a set of rules that underline how the vocabulary's items may be grouped together and used to form sentences.

There are generally two fundamental categories of language models [66]. The first is built on predefined, frequently manually written rules, widely known as formal grammars which can label a word sequence as either correct or wrong. They can be employed in scenarios when we anticipate a small number of distinct word combinations and a short vocabulary. Since it is impractical to incorporate all potential rules into a formal grammar, this kind of language models cannot be employed in large vocabulary continuous speech recognition.

The second category of language models, known as stochastic models or statistically based models, calculate the likelihood that a given word sequence will occur in the modeled language. The most popular word-based n-gram models are utilized [73] or models based on them [172]. The a priori probability of a word sequence $w_1, ..., w_k$ can be calculated using chain rule as:

$$P(w_1, ..., w_k) = P(w_1)P(w_2 \mid w_1)...P(w_k \mid w_1, ..., w_{k-1}) \tag{2.25}$$

The core purpose of the n-gram language model is to estimate the a posteriori probability of a word w_k given an arbitrary word sequence $w_1, ..., w_{k-1}$, namely $P(w_k \mid w_1, ..., w_{k-1})$. This approach, however, is not applicable because a large number of histories can not be seen in the training corpus. In order to simplify the modeling process when the n-gram model is used as the language model, the following assumption must be made:

$$P(w_k \mid w_1, ..., w_{k-1}) \approx P(w_k \mid w_{k-n+1}, ..., w_{k-1}) \tag{2.26}$$

This is sometimes referred to as the Markov assumption and states that the likelihood of the next word depends exclusively on the $n-1$ most recent words.

Usually, the probabilities of the n-gram model are approximated using Maximum Likelihood Estimation (MLE):

$$P\left(w_k \mid w_{k-n+1}, \ldots, w_{k-1}\right) = \frac{C\left(w_{k-n+1}, \ldots, w_{k-1}, w_k\right)}{C\left(w_{k-n+1}, \ldots, w_{k-1}\right)} \tag{2.27}$$

where $C(W)$ indicates the number of occurrences of word sequence W in the training corpus.

Depending on the activities for which the language model will be utilized, different types of texts should be used to train it. For instance, if the goal is to build a language model to be utilized in a task of recognizing children's speech, a lot of texts that are linked to children's literature must be employed. Using more generalized data, such as those found in articles or television subtitles, a more generic language model can be essentially built. When there isn't sufficient amount of related texts, these generic models can also be utilized to create task-adapted models by interpolating a generic language model with a smaller model created from a particular text source.

2.5 Acoustic Modeling

2.5.1 Gaussian Mixture - Hidden Markov Models (GMM - HMM)

In classical acoustic modeling, given an input X, for example MFCC features derived from a raw audio waveform, the goal is to predict:

$$W^* =_W P(W|X) \tag{2.28}$$

, the probabilistically optimal sequence of words obtained by the model having seen X. This can be modelled as:

$$W^* =_W P(W|X) \tag{2.29}$$

By applying Bayes' Rule, this produces:

$$W^* =_W P(W|X) = P(X|W) * P(W)/P(X) = argmax_W P(X|W) * P(W) \tag{2.30}$$

In the last step, it is possible to remove $P(X)$ from the denominator because this probability is independent of the variable W, which is being maximized with respect to.

Now, in this scenario, we typically model $P(X \mid W)$ using the acoustic model and $P(W)$ given the language model. This is because the probability of W is independent of the acoustic features X and instead has to do with a word's probability in the natural use of language independent of the audio being presented.

As already explained in section 2.3.1, a Hidden Markov model is composed of a collection of states, a transition matrix that indicates the likelihood of changing between any two states, a set of observations, and a set of emission probabilities that indicate the likelihood that an observation will emerge from a certain state. In HMMs for speech recognition, the observations are the acoustic feature vectors provided by MFCCs or similar, and the states are phonemes, or units of sound utilized in word articulation. The majority of HMM models for speech are left-to-right, i.e. they place strong priors on transition probabilities such that certain phonemes can only occur after others and that the HMM cannot backtrack to previous states. This type of constrained HMM is called a Bakis network and is constrained due to how rigorously constructed human speech is. Such a network also allows self

loops, allowing a phoneme to repeat in case its utterance duration is greater than the 25ms window width.

Estimating the probabilities of observing one of the observed acoustic MFCC vectors is required as the next step in building the HMM model. Since we don't have access to these probabilities, we often make the assumption that they follow a multivariate Gaussian distribution

$$\frac{1}{2\pi}^{D/2} |\Sigma|^{1/2} \exp\left(-\frac{1}{2}(o_t - \mu_j)^T \Sigma_j^{-1}(o_t - \mu_j)\right)$$

where o_t is the observation vector at time t, μ_j is the mean vector of the observations and D is the dimensionality of the observations. Σ is the diagonal covariance matrix with each entry along the diagonal corresponding to the covariance of a single observation vector dimension. It should be noted how μ_j and Σ are both readily computable from the observed data.

The aforementioned multivariate Gaussian distributions are combined to represent the observation likelihoods in the GMM. The Baum-Welch technique, a variant of the Expectation-Maximization algorithm, is utilized to jointly train this mixture model and the HMM [128]. An example of a GMM-HMM system is depicted in Fig. 14.

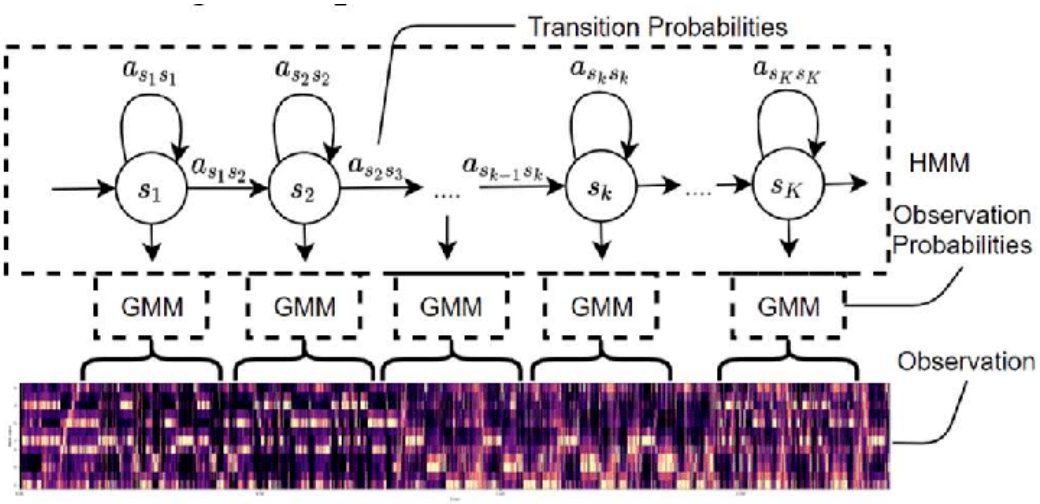

Figure 14: GMM-HMM mixture model

2.5.2 Deep Learning Hybrid Models (DNN - HMM)

2.5.2.1 Overview

The newest trend in voice recognition is Deep Neural Networks (DNNs). Statistical data-driven techniques have facilitated improvements in automated speech recognition. Deep neural networks have played a significant role in acoustic modeling replacing GMM in the HMM-GMM acoustic model during the past few years, propelling advancement [62].

Some of the biggest industry pioneers, such as Google [70] and Microsoft [25], are starting to deploy DNNs in their production systems, and numerous papers have been published in this field. Deep Belief Networks (DBN) [103] and Restricted Boltzmann Machines (RBM) [104] were common models for the DNN in the early 2010s. These days, however, sequence based models such as Long short-term memory models (LSTMs) [63], Bidirectional-LSTMs [54], or Gated Recurrent Unit models (GRU) [20] are most frequently used.

Fig. 15 shows a common approach of a deep neural network speech recognition pipeline containing the following components: A spectrogram generator that translates raw audio signals into spectrograms, a neural acoustic model that receives the spectrograms as input and outputs a matrix of probabilities over characters over time, a decoder (optionally coupled with a n-gram language model) which produces possible sentences from the probability matrix, and last but not least, a punctuation and capitalization model that formats the generated text for easier human interaction.

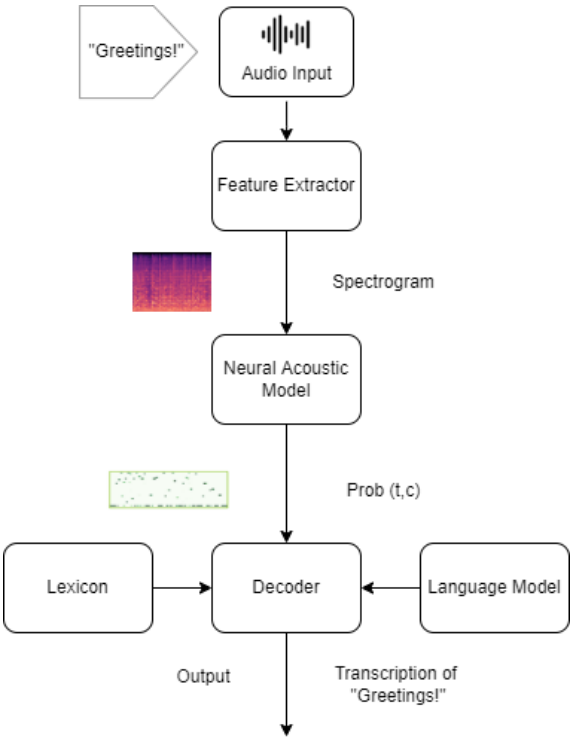

Figure 15: Example of a deep learning speech recognition pipeline

2.5.2.2 Low-Resource DNN ASR

The restrictions brought on by a lack of resources are clear, despite the fact that DNN-based acoustic models have significantly outperformed traditional ASR systems [2, 54]. Large quantities of transcribed speech data are needed for the successful training of effective DNN models, but this data are still difficult to come by and sometimes expensive for many domains and languages. One way to overcome the obstacle of low training data, could be the augmentation of the training set. Artificial generation of training data in the domain of speech recognition was originally considered in [13, 14] and proved to be effective when only limited size speech data were available for training. These methods, however, were only concerned with accommodating GMM-HMM models and depended on excessive a priori knowledge of the underrepresented target speech domain. The study in [50] aimed at closing the gap between DNN–HMM and GMM–HMM in situations when the available data is adequate for effective GMM training but at the same time too limited to extract meaningful DNN models. A detailed survey in automatic speech recognition for under-resourced languages can be read in [12] and for deep learning individually in [140].

2.5.2.3 Comparison of GMM-HMM and DNN-HMM

One of the most significant developments in speech recognition technology over the past ten years is the deep neural network's success with LVSR. Up until the early 2010s, only HMM-GMM systems in a variety of configurations were employed in speech recognition. However, by leveraging the rapid increase of computational power in modern computer systems, when presented with large vocabulary continuous speech recognition tasks, the DNN hybrid approach significantly outperformed the traditional acoustic models utilizing continuous density HMMs based on Gaussian mixture models [22, 141, 142]. As a consequence of the DNNs being great at discriminative learning, particularly when trained on sufficiently sized datasets, they have demonstrated much improved accuracy for modeling the observation probabilities [62] than traditional GMMs which have been a convenient method of modeling HMM state distributions for a long time [91, 129, 149].

In [117], the research team investigated DNN-HMM architecture for several large vocabulary speech recognition tasks. Their findings showed that DNN can consistently achieve about $25 - 30\%$ relative error reduction over the best discriminatively trained GMMs even with up to 700 hours of training data in some ASR tasks. They also carried out a number of experiments to determine where the DNN's all-time high gain originates. According to the research, DNN's feature vectors, which are concatenated from a number of consecutive speech frames within a large context window, are nearly entirely responsible for the gain of the DNN. During that period, Deng et al. [25] and other's work also demonstrated that deep learning is a powerful technology; e.g. on the Switchboard ASR task the word error rate had reduced sharply from 23% in the GMM-HMM system as state-of-the-art to as low as 13% at the time [78, 152]. Regarding other automatic speech recognition tasks, in [144], the researchers presented a pronunciation verification method for integration into an automated speech therapy tool for children with Childhood Apraxia of Speech (CAS) while comparing the performance of conventional GMM-HMMs and hybrid DNN-HMMs. For the GMM-HMM, they achieved a minimum PER of 37% when tested with normal speech and 43% with disordered speech. When using the DNN-HMM, the PER decreased to 21% and 36% for normal and disordered speech respectively. They also reported that the DNN-HMM performed better than the GMM-HMM with disordered speech, due to its ability to train better with limited size training data. These days, in addition to hybrid ap-

proaches, end-to-end ASR systems are employed in a variety of techniques [53, 165, 171]. Nevertheless, hybrid models continue to be among the finest systems with the greatest performance, despite several advances in end-to-end models.

Typically, an HMM and a DNN model are coupled to form a DNN-HMM system, or more accurately, a hybrid model. In the DNN-HMM framework, an estimation based on DNN is used in place of a GMM-based probability approximation for each state of the HMM. While the DNN-HMM utilizes a feature vector consisting of a concatenation of many consecutive frames as its input, the standard GMM-HMM utilizes a frame-based feature vector as its input to the GMM. In addition, the DNN models are normally trained directly on the Mel filter bank features directly rather than the MFCCs, offering as advantages reduced pre-processing time and richer feature representations. All other processes, including the HMM-based state transition model, pronunciation model, and language model, are left unchanged except from the conditional probability estimation for each state. Consequently, the decoding process has no significant modifications. Fig. 16 depicts the paradigm transition from HMM-GMM to HMM-DNN that has occurred in the ASR area in recent years. The traditional GMM-HMM acoustic modeling framework is displayed on the left side.

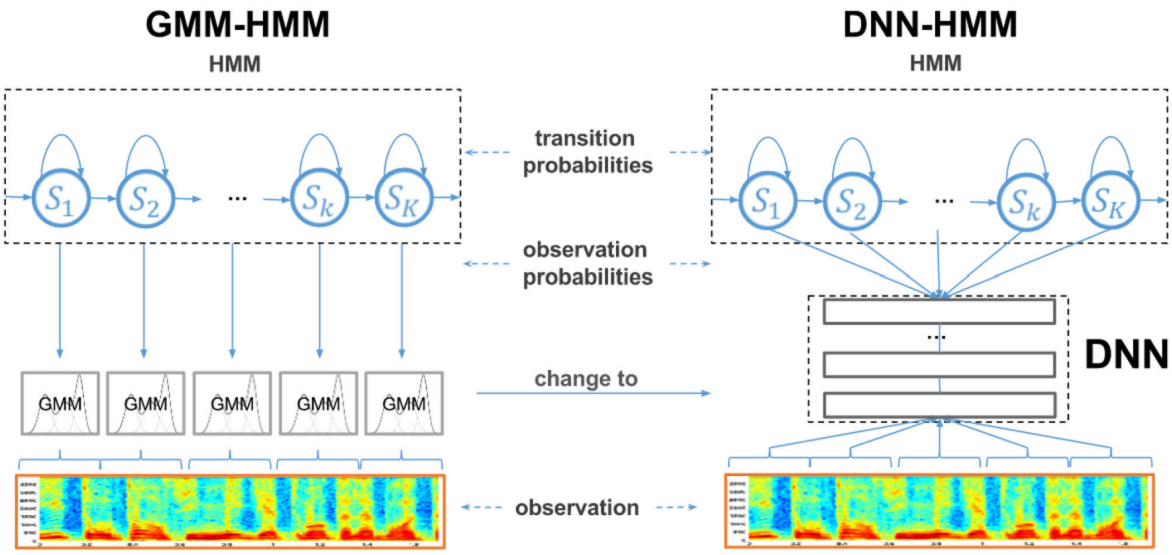

Figure 16: GMM-HMM vs DNN-HMM acoustic models

2.6 Weighted Finite State Transducers

2.6.1 Weighted Acceptors

Weighted finite state automata (or weighted acceptors) consist of a set of states, an initial state, a set of final states (with final weights), and a set of transitions between states. Each transition has a source state, a destination state, a label and a weight. Such automata are called weighted finite state acceptors (WFSA), since they accept or recognize each string that can be read along a path from the start state to a final state. Each accepted string is assigned a weight, namely the accumulated weights along accepting paths for that string, including final weights. An acceptor as a whole represents a set of strings, namely those that it accepts. As a weighted acceptor, it also associates to each accepted string the accumulated weights of their accepting paths.

2.6.2 Weighted Transducers

Weighted finite-state transducers (WFSTs) generalize WFSAs by replacing the single transition label with a pair of an input label and an output label, plus a weight on each of its transitions. While a weighted transducer associates symbol sequences and weights, a WFST can represent a relationship between two levels of representation, for instance between phones and words or between HMMs and context-independent phones. More precisely, a transducer specifies a binary relation between strings: two strings are in the relation when there is a path from an initial to a final state in the transducer that has the first string as the sequence of input labels along the path, and the second string as the sequence of output labels along the path. In general, this is a relation rather than a function since the same input string might be transduced to different strings along two distinct paths. A weighted transducer puts weights on transitions in addition to the input and output symbols. Weights may encode probabilities, durations, penalties, or any other quantity that accumulates along paths to compute the overall weight of mapping an input sequence to an output sequence. Weighted transducers are therefore a natural choice to represent the probabilistic finite-state models prevalent in speech processing.

WFSTs provide a consistent and natural representation for HMM models, context dependency, pronunciation dictionaries, grammars, and alternative recognition outputs. Generalized finite-state operations combine these representations quickly and effectively. A weight pushing method distributes the weights throughout the routes of a weighted transducer, whereas weighted determinization and minimization algorithms manage efficiently their time and space needs as explained in the following section. For instance, WFSTs are extensively utilized in Kaldi, a popular toolkit for automated speech recognition, however they are often employed for inference or to estimate the parameters of shallow models. More details about WFSTs in Kaldi can be found in section 3.7.

2.6.3 Weighted Transducer Algorithms

2.6.3.1 Composition

As the name implies, this term describes the process of joining two WFSTs to create a single WFST. For instance, a word-level grammar and a pronunciation lexicon can be composed to create a phone-to-word level transducer whose word sequences are bound to the grammar.

Composition is performed by applying three stages:

- By integrating the original states of the previous WFSTs into pairs, the initial states of the new WFST are created.
- Final states are paired together in a similar manner.
- We add an edge from the source pair to the destination pair for each pair of edges where the o-label of the first WFST is the i-label of the second. The sum rules are applied to the edge weight.

This is an example of composition:

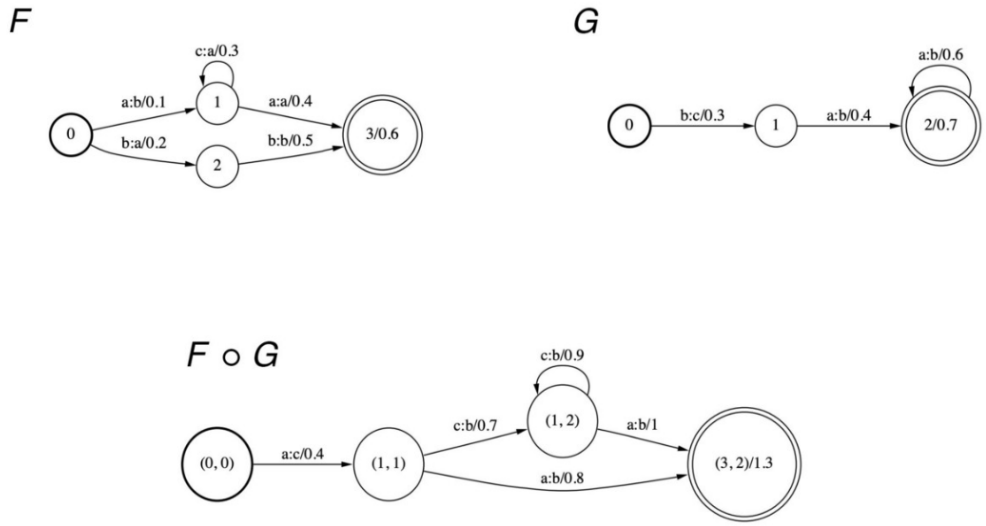

Figure 17: Example of a WFST composition

2.6.3.2 Determinization

The process of determination removes multiple paths, leaving just one path for each input sequence. It dramatically reduces the overall the amount of time and space needed to decode an input sequence. This is particularly important in ASR due to pronunciation lexicon redundancy in large vocabulary tasks. However, not all WFST can be determinized: a weighted transducer is *deterministic* or *sequential* if and only if each of its states has at most one transition with any given input label. By such a formulation, a deterministic WFST removes all redundancy and greatly reduces the complexity of the underlying grammar. There are several techniques to determinize a WFST. Fig. 18 shows an example of the determinization operation.

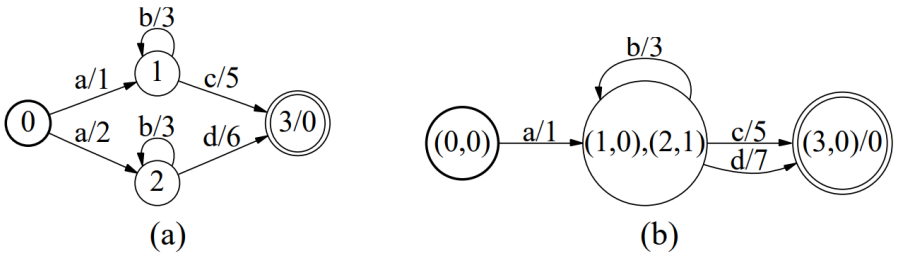

Figure 18: Example of determinization of weighted automata. (b) weighted automaton obtained by determinization of (a).

2.6.3.3 Minimization

Although minimization is not as essential as determinization, it is still a helpful optimization technique. It refers to minimizing the number of states and transitions in a deterministic WFST.

Minimization involves two steps:

- All weights are pushed towards the start state as seen in Fig. 19 (**Weight Pushing**).
- Combining those states which have identical paths to any final state. In the example of Fig. 19, states 1 and 2 are equivalent after weight pushing, so they are merged into one state.

Fig. 20 depicts the complete pipeline of a WFST reduction, which involves all three operations.

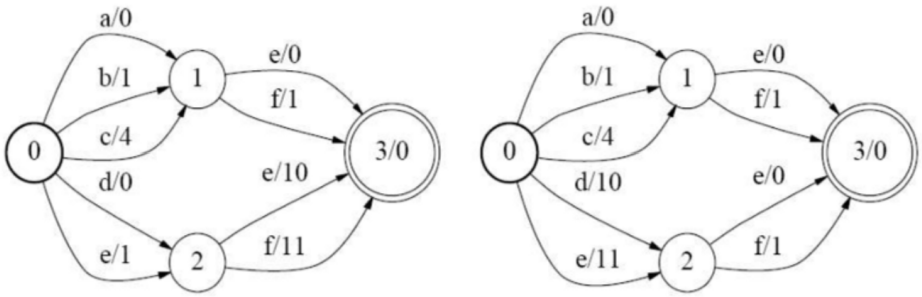

Figure 19: Example of a WFST minimization

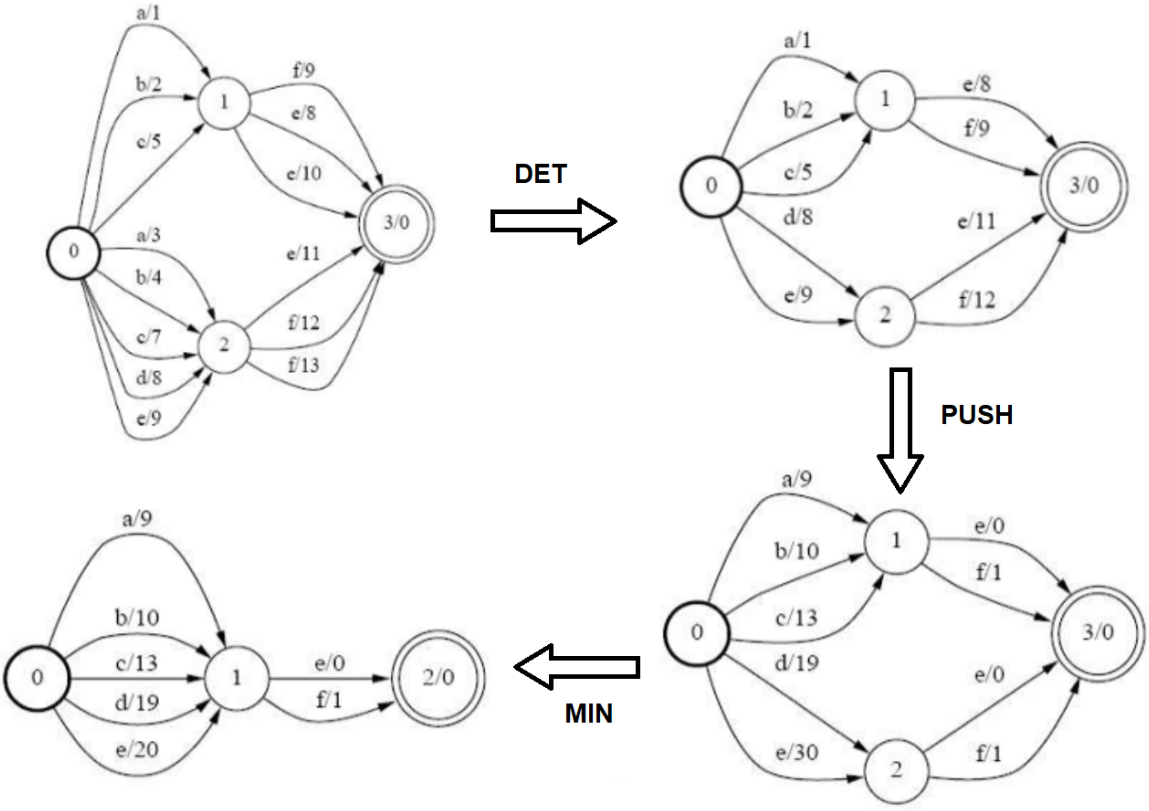

Figure 20: Example of a WFST reduction

2.6.4 WFSTs in Speech Recognition (HCLG Graph)

When it comes to simulating HMMs and resolving state machine issues, Weighted Finite-State Transducer excels. It offers a natural representation for HMM phonetic models, context-sensitive phones, pronunciation lexicons, and grammars in the context of ASR. Three stages are involved in decoding speech utterances: decoding graph generation, decode and lattice generation and language model rescoring.

2.6.4.1 Decoding Graph Generation

The decoding graph (HCLG) using either a GMM-HMM or a DNN-HMM architecture is modeled as a finite-state transducer formed by composing an FST representation of the HMM, phonetic context-dependency, pronunciation dictionary and trigram language model. The Finite State Transducer framework and its implementation, OpenFST, provide well studied graph operations [105] which can be effectively used for acoustic modelling. Using the FST framework the speech decoding task is expressed as a beam search in a graph, which is well studied problem. The FST graphs used for AM model training and speech decoding can be constructed as sequence of standardized OpenFST operations .In general, the composed transducer $H \circ C \circ L \circ G$ represents the entire pipeline of speech recognition. Each of the components can individually be improved, so that the entire ASR system gets improved.

$$HCLG = H \circ C \circ L \circ G \tag{2.31}$$

The symbol $\circ$ represents an associative binary operation of composition on FSTs. Several WFSTs are composed in sequence for use in speech recognition. We briefly explain the functionality of these transducers from Equation 2.31:

- **Grammar (G)** is an acceptor that encodes the grammar or language model.
- **Lexicon (L)** represents the lexicon and provides information about the likelihood of phones without context.
- **Context-dependent phonetics (C)** represents the relationship between context-dependent phones on input and phones on output.
- **HMM Structure (H)** contains the HMM definitions, that take as input id number of probability density functions and return context-dependent phones.

Despite the fact that this is the standard recipe there are a lot of details underneath, it is necessary that the output is determinized and minimized, and in order for HCLG to be determinizable disambiguation symbols must be inserted. It is essential, as well, that the HCLG is stochastic as much as possible. In the conventional recipes, this is achieved (if at all) with the ”push-weights” operation. The approach to ensuring stochasticity is different, and is based on ensuring that no graph creation step removes stochasticity;

In Table 3 the input-output of each transducer is presented.

Table 3: HCLG graph components I/O

	Transducer	Input	Output
G	Word-level Grammar	Words	Words
L	Pronunciation Lexicon	Phones	Words
C	Context-Dependency	Context-Dependent Phones	Phones
H	HMM	HMM States	Context-Dependent Phones

Following one liner illustrates how Kaldi decoding graph is created using standard FST operations.

$$\mathbf{H} \circ \mathbf{C} \circ \mathbf{L} \circ \mathbf{G} = min(det(\mathbf{H} \circ min(det(\mathbf{C} \circ min(det(\mathbf{L} \circ \mathbf{G})))))) \tag{2.32}$$

Fig. 21 displays the full HCLG graph decoding pipeline.

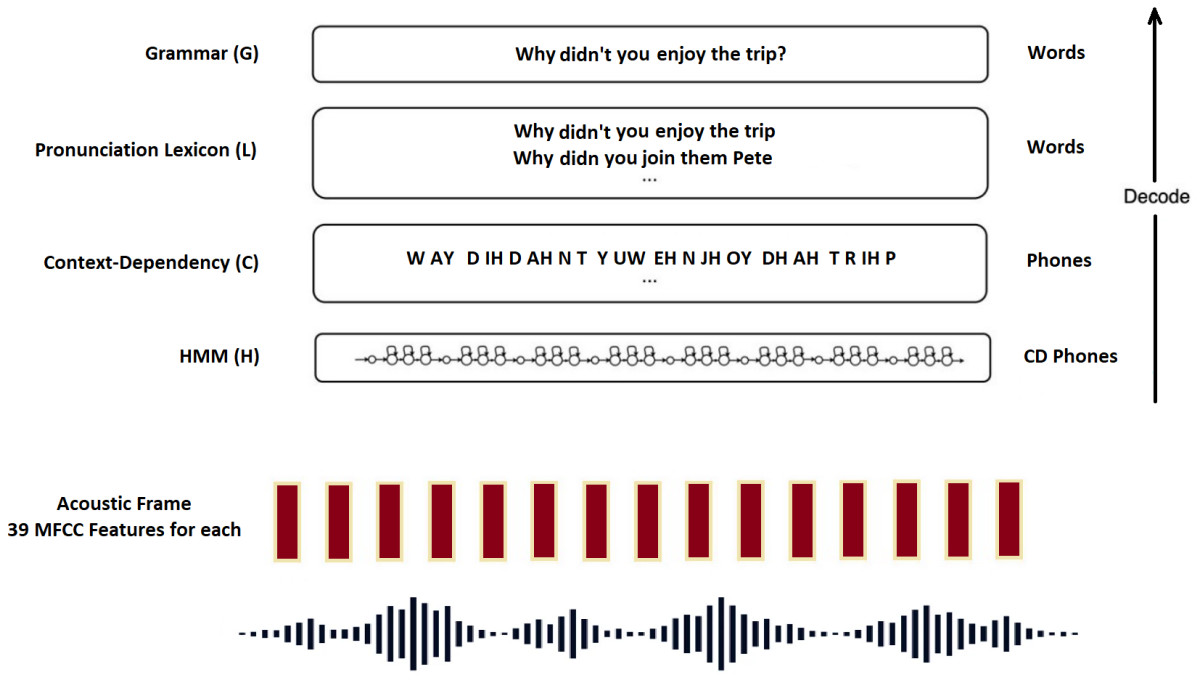

Figure 21: The HCLG graph decoding pipeline

2.6.4.2 Decoding & Lattice Generation

The decoder computes the most likely word sequence for a certain audio input. In many circumstances it is required to generate more than just the best scoring outcome, but all similar word sequences. A word lattice is a useful tool for creating and compactly encoding many sentence variations (or word graph). Weighted finite-state transducer is a practical tool for representing the word lattice. Every route through this FST is an alternate weighted sentence. An N-best list can be generated efficiently as the N best paths in the lattice [107].

The HCLG decoding graph contains a set of context dependent states with weighted arcs between the individual states. For each state transition, the input symbols correspond to context-dependent HMM states and the output symbols are words. To decode and generate the lattice, each arc of HCLG is traversed for each input feature vector and state-level arcs are created for the acoustic and graph costs. Then beam width pruning is applied in a window of frames to keep the most likely result with the corresponding graph and acoustic costs.

2.6.4.3 Language Model Rescoring

For rescoring, the lattice that contains the entire surviving path is rescored by applying a language model scaling factor over a range. This rescoring method over the lattice filters the non-sense word sequences that have the lowest probability scores in the language model. The HMM transition probabilities, HMM emission probabilities, and language model probabilities are encoded in the graph via the weights. The optimal path is found by searching this graph using the Viterbi algorithm. It represents the most likely word sequence given the acoustic features extracted from the input audio. Mohri et al. in [106] provided an in-depth analysis of speech recognition with Weighted Finite Transducers.

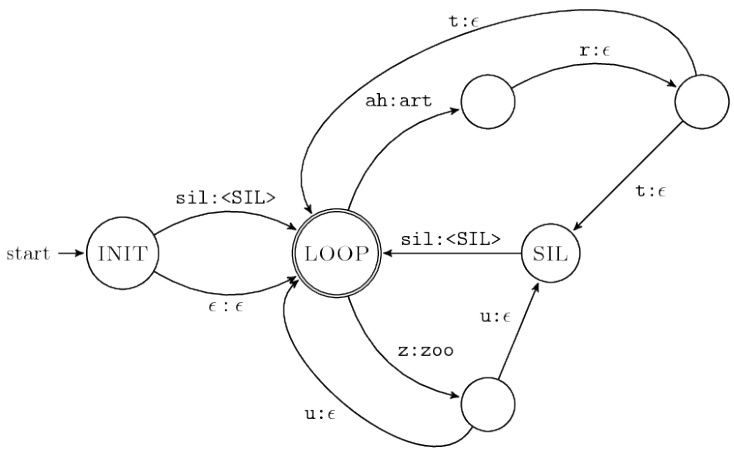

Figure 22: An example of the phonetic dictionary FST with two words

3. KALDI

With practically every technique currently utilized in ASR systems, Kaldi [125] is a cutting-edge open source toolkit for voice recognition developed in C++ and designed for use by speech recognition researchers.

Multiple open-source toolkits are available for automated speech recognition researchers to choose from when developing a recognition system. These include the RWTH ASR toolkit [95, 138] (written in C++) , Julius [86] (written in C), Sphinx-4 [163] (written in Java), and HTK [169] . The necessity for a finite-state transducer based framework, for extensive linear algebra support, and a non-restrictive license led to the development of the Kaldi toolkit. Kaldi provides recipes for training acoustic models on widely used speech corpora such as the Wall Street Journal Corpus [121], TIMIT [48], and several other. These recipes can also serve as a template for training acoustic models on manual speech data.

3.1 Overview

The toolkit is dependent on two external libraries, both of which are free to use: one is OpenFst [4] for the finite-state framework, and the other is numerical algebra libraries. For the latter, the standard Basic Linear Algebra Subroutines (BLAS) and Linear Algebra PACKage (LAPACK) routines are utilized. It is possible to divide the library modules into two separate groups, each of which depends solely on one of the external libraries. The DecodableInterface serves as the link between these two sections.

For the purpose of creating and operating a speech recognizer, command-line tools developed in C++ are used to offer access to the library's features. Each tool only accepts a minimal number of command line inputs and has very specific functionality: for example, there are separate executables for accumulating statistics, summing accumulators, and updating a GMM-based acoustic model using maximum likelihood estimation. Additionally, since every tool can read from and write to pipelines, chaining together various tools is made simple.The toolkit is designed in a manner that incorporating a new feature typically entails adding new code and command-line tools rather than updating old ones in order to prevent a gradual degradation of the program quality over time. Fig. 23 depicts a schematic overview of the Kaldi toolbox.

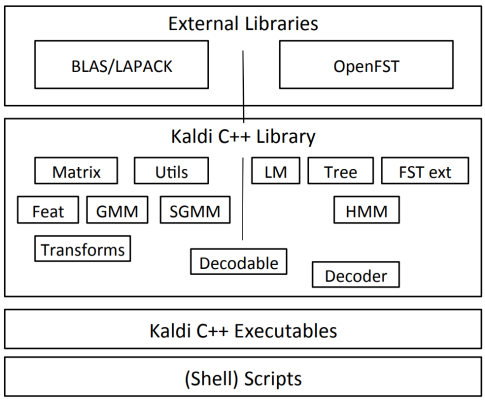

Figure 23: The different components of the Kaldi toolkit

3.2 Pipeline

A typical automatic speech recognition pipeline using Kaldi follows a similar sequence of steps to other ASR systems. The pipeline typically involves the following stages:

- In the speech pre-processing stage, the raw audio signal is cleaned and filtered to remove noise and other interference. This step helps improve the overall accuracy of the ASR system.
- In the feature extraction stage, the cleaned audio signal is processed to extract relevant features that can be used to represent the speech. Kaldi includes a range of algorithms for feature extraction, including methods for extracting Mel-frequency cepstral coefficients and other spectral and temporal features.
- In the acoustic modeling stage, the extracted features are used to train a model that can map the acoustic characteristics of the speech to the corresponding words or phrases. Kaldi includes a range of algorithms for acoustic modeling, including hidden Markov models and deep neural networks.
- In the language modeling stage, a separate model is trained to predict the likelihood of a sequence of words or phrases given the context. Kaldi includes a range of algorithms for language modeling, including n-gram language models and recurrent neural networks (RNNs).

Once these models have been trained, the ASR system can process a new audio signal, extract relevant features, and use the trained models to convert the speech into text. This process typically involves using the acoustic model to generate a set of potential word or phrase sequences, and then using the language model to score and rank these sequences to determine the most likely transcription. Fig. 24 depicts the Kaldi pipeline.

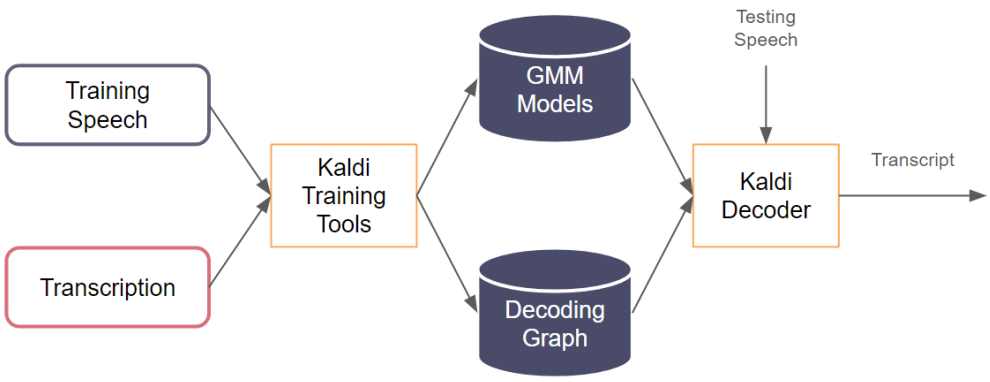

Figure 24: The Kaldi pipeline

3.3 Feature extraction

The feature extraction and waveform-reading code seeks to produce standard MFCC and PLP features, providing reasonable defaults while also making the most often used choices available for customization (for example, the number of mel bins, minimum and maximum frequency cutoffs, etc.). Support is provided for the majority of widely employed feature extraction techniques, including Vocal Tract Length Normalization (VTLN), cepstral mean and variance normalization (CMVN), Linear Discriminant Analysis (LDA), Semi-tied Covariance/Maximum Likelihood Linear Transform (STC/MLLT), Heteroscedastic Linear Discriminant Analysis (HLDA), and others.

3.4 Acoustic Modelling

The primary function of Kaldi is to handle established models, such as diagonal Gaussian Mixture Models and Subspace Gaussian Mixture Models (SGMMs), while also being easily expandable to new types of models. The toolbox supports both feature-space adaptation using feature-space MLLR (fMLLR), also known as constrained MLLR [46], and model-space adaptation using Maximum Likelihood Linear Regression (MLLR) [87]. Additionally, each context-independent phone's HMM structure may be specifically specified in Kaldi. The topology format supports non-emitting states and enables the user to specify how the PDFs will be tied in various HMM states in advance.

3.5 Phonetic Decision Trees

The phonetic decision tree code makes it efficient for arbitrary context sizes (i.e. enumerating contexts were avoided), and also makes it general enough to support a wide range of approaches. The standard approach is to have a decision tree that asks queries about, for example, the left and right phones in each HMM-state of each monophone. The decision-tree roots in this system may be shared between phones and between phone states, and queries can be posed about any phone in the context window as well as the HMM state. Language-based knowledge may be used to provide phonetic questions, but in kaldi recipes, the questions are created automatically using a tree-clustering of the phones. An expanded phone set may be used to ask questions about things like phonetic stress (if marked in the dictionary) and word start/end information; in this case the decision-tree roots are shared among the different versions of the same phone.

3.6 Language Modeling

It is theoretically feasible to employ any language model that can be described as an FST because Kaldi is built on an FST-based framework. Tools for converting LMs in the common ARPA format to FSTs are included in the package. The IRST Language Modeling (IRSTLM) toolbox [41] is utilized in a number of recipes for tasks like LM pruning. Users can utilize the IRSTLM toolkit or a more feature-rich toolkit like SRILM [150] to create LMs from raw text.

3.7 Decoding Graphs

Weighted Finite State Transducers are used in all training and decoding techniques (WFSTs). The input symbols on the decoding graph in the standard recipe [105] correspond to context-dependent states; in the Kaldi toolbox, these symbols are numeric and are referred to as pdf- ids. This approach would have a number of issues, including the inability to determinize the FSTs and a lack of information from the Viterbi route through an FST to determine the phone sequence or train the transition probabilities because various phones are permitted to share the same pdf-ids. The ”transition-id,” which encodes the pdf-id, the phone it is a member of, and the arc (transition) inside the topology specification for that phone, is added to the input of the FSTs in order to address these issues. The ”transition-ids” and the ”transition-probability parameters” in the model map to each other exactly: the decision was made to make transitions as precise as possible without enlarging the decoding graph.

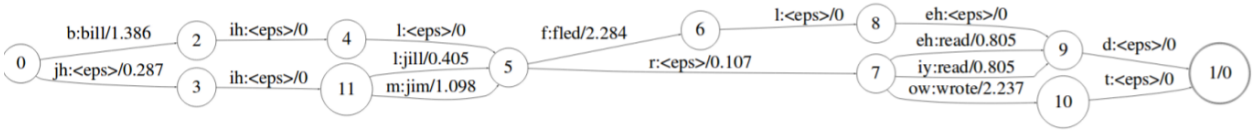

Figure 25: A simple representation of a WFST taken from “Springer Handbook on Speech Processing and Speech Communication”. Each connection is labeled: Input:Output/Weighted likelihood

The technique used to create the decoding-graph is based on the recipe described in [105]; there are a few slight differences, though. One significant one concerns the manner in which the procedure known as ”weight-pushing,” which is intended to ensure that the FST is stochastic, is handled. ”Stochastic” denotes that, for each state, the weights in the FST add up to one in the correct meaning (like a properly normalized HMM). Weight pushing may not succeed or may result in undesirable pruning behavior if the FST representing the grammar or language model (G) is not stochastic, as is the case with backoff language models. The Kaldi approach seeks to prevent weight-pushing completely while ensuring that each step of graph construction appropriately ”preserves stochasticity”. This essentially indicates that the inability to sum to one will never be worse than it was in G.

3.8 Decoders

There are several decoders available, ranging from basic to highly tuned. More will be added in the future to deal with issues like on-the-fly language generation, model rescoring, and lattice generation. A C++ class that implements the fundamental decoding technique is referred to as a ”decoder”. It is not necessary for the decoders to use a certain kind of acoustic model: they require an object satisfying a very simple interface with a function that provides some kind of acoustic model score for a particular (input-symbol

and frame). The programs used for command-line decoding are all relatively straightforward, do just one pass of decoding, and are each tailored to work with a certain decoder and acoustic model type. Multi-pass decoding is implemented at the script level.

4. KALDI GRPC SERVER

4.1 Overview

Kaldi gRPC Server[1] was created as a modern alternative for deploying Speech Recognition models and was developed using Kaldi. In addition to providing highly accurate offline speech recognition, the server is a useful component to include in websites, chat-bots and telephony as well as a backend for streaming speech recognition over the internet. The following sections provide further details on the capabilities of the tool, while section 4.5 analyzes the contribution of this thesis to Kaldi-gRPC-Server, the speaker diarization component.

4.2 Features

Currently, Kaldi gRPC Server features a standardized API based on a modified version of Jarvis proto files, which mimics the Google speech API. This allows for easy switching between Gloud speech recognizers and custom models developed with Kaldi. The tool is fully implemented in Python and PyKaldi bindings [15] are utilized to interface with Kaldi programmatically as explained in section 4.4.1. This facilitates a clean, customizable and extendable implementation.

It also features a fully bidirectional streaming using HTTP/2 (gRPC). Binary speech segments are streamed to the server and partial hypotheses are streamed back to the client with ease. Long speech can be transcribed arbitrarily and DNN-HMM models are supported out of the box. On top of that, Recurrent Neural Network Language Modeling (RNNLM) lattice rescoring is supported. Clients for other languages can be easily generated using the proto files too.

There is an option to build for all intents and purposes a binary file image using the kaldi bindings through singularity containers [84] to allow seamless integration with any scientific computational resources. In short, singularity containers build a fakeroot filesystem into a single, executable file.

Fig. 26 depicts an example of automatic speech recognition with a model deployment using Kaldi-gRPC-Server. The target audio clip was a short microphone recording of a sentence in English. Additional models for several ASR tasks with greater accuracy can be traced in Kaldi official site[2].

[1] https://github.com/georgepar/kaldi-grpc-server

[2] https://kaldi-asr.org/models.html

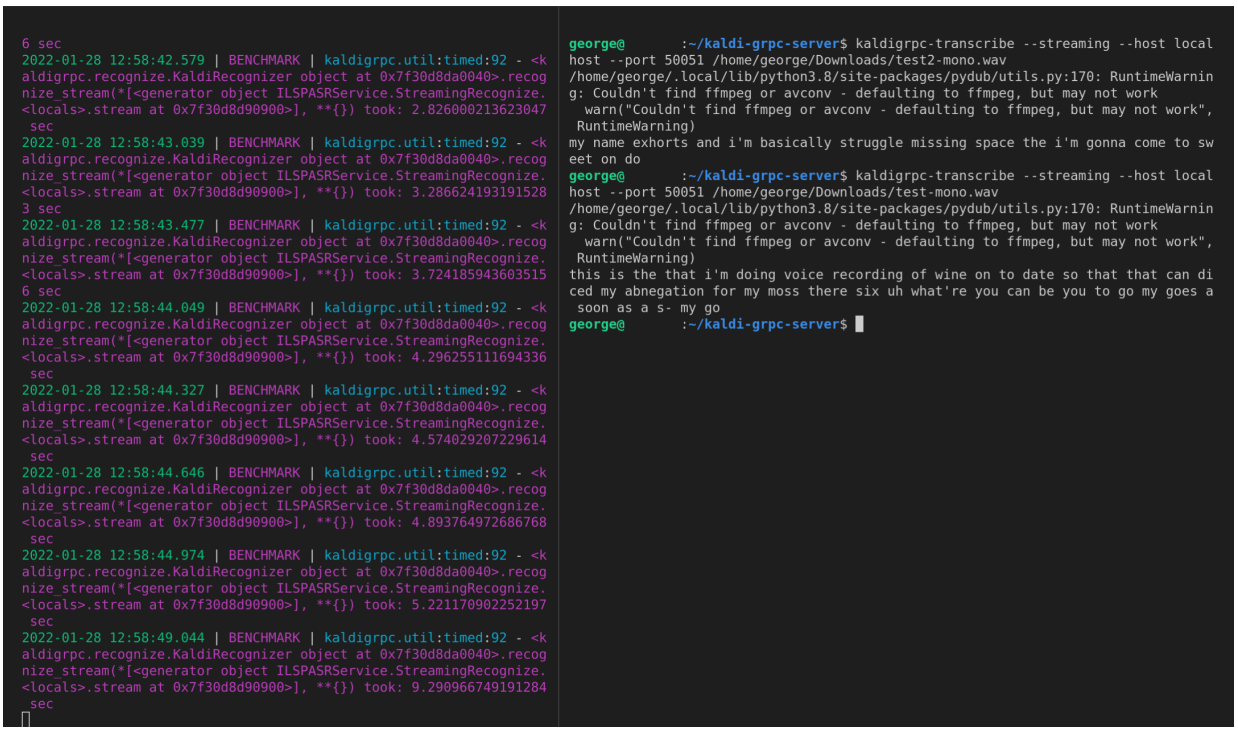

Figure 26: Example of ASR by deploying the Chime6 model.

4.3 Kaldi Model Structure

The following structure for the deployed model is recommended:

```
model
├── conf
│   ├── ivector_extractor.conf
│   ├── mfcc.conf
│   ├── online_cmvn.conf
│   ├── online.conf
│   └── splice.conf
├── final.mdl
├── global_cmvn.stats
├── HCLG.fst
├── ivector_extractor
│   ├── final.dubm
│   ├── final.ie
│   ├── final.mat
│   ├── global_cmvn.stats
│   ├── online_cmvn.conf
│   ├── online_cmvn_iextractor
│   └── splice_opts
└── words.txt
```

Figure 27: Kaldi model structure

The key files / directories are:

- **conf**: Configuration files that are used to train the model
- **final.mdl**: The acoustic model
- **HCLG.fst**: The composed HCLG graph (output of mkgraph.sh)
- **global_cmvn.stats**: Mean and standard deviation used for CMVN normalization
- **words.txt**: Vocabulary file, mapping words to integers
- **ivector_extractor**: Model trained to extract ivector features (used for tdnn / chain models)

4.4 Wrapping Kaldi

Kaldi gRPC Server is fully written in Python, utilizing PyKaldi bindings as a way to interface with the Kaldi framework. Users typically interact with Kaldi either by running its highly modular and composable command-line programs manually inside a UNIX shell or by writing scripts that run these programs. Any functionality that is not exposed by one of the myriad command-line Kaldi programs can be accessed via the C++ application programming interface (API). While this interaction scheme is highly effective, it does not fully address the needs of researchers and developers who would like to use Kaldi in programming languages other than C++.

4.4.1 PyKaldi

PyKaldi [15] aims to bridge the gap between Kaldi and all the nice things Python has to offer. It is more than a collection of bindings into Kaldi libraries. It is a scripting layer providing first class support for essential Kaldi and OpenFst types in Python. PyKaldi vector and matrix types are tightly integrated with NumPy. They can be seamlessly converted to NumPy arrays and vice versa without copying the underlying memory buffers. PyKaldi FST types, including Kaldi style lattices, are first class citizens in Python. The API for the user facing FST types and operations is almost entirely defined in Python mimicking the API exposed by pywrapfst, the official Python wrapper for OpenFst.

PyKaldi harnesses the power of CLIF to wrap Kaldi and OpenFst C++ libraries using simple API descriptions. The CPython extension modules generated by CLIF can be imported in Python to interact with Kaldi and OpenFst. While CLIF is great for exposing existing C++ API in Python, the wrappers do not always expose a ”Pythonic” API that is easy to use from Python. PyKaldi addresses this by extending the raw CLIF wrappers in Python (and sometimes in C++) to provide a more ”Pythonic” API. Below figure illustrates where PyKaldi fits in the Kaldi ecosystem.

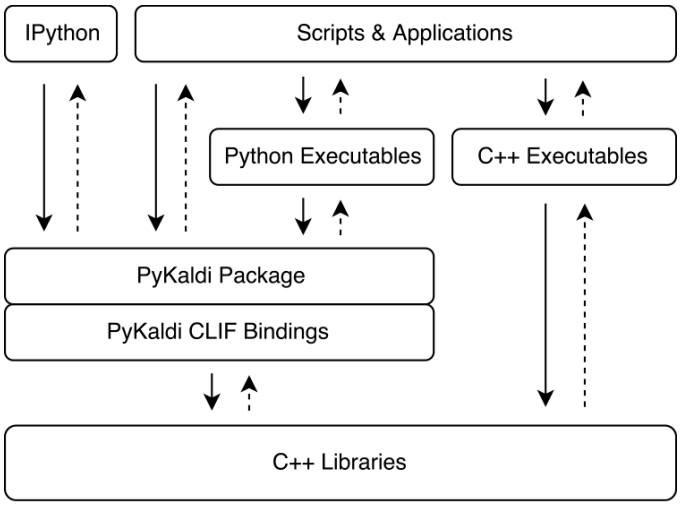

Figure 28: Extended Kaldi software architecture

4.5 Speaker Diarization Component

ASR is not the only option provided in Kaldi gRPC Server; the most recent addition to the framework's features arsenal and part of this thesis is the speaker diarization component. Speaker Diarization is also a powerful tool. By identifying and labeling speakers, one can analyze each speaker's behaviors, scan for patterns and trends among individual speakers, make predictions and so forth. Some typical examples of speaker diarization applications:

- A podcast service can utilize speaker labels to identify the host and guest, providing readable transcriptions to the audience.
- In order to identify patterns that can enhance communication, a call center might examine agent-customer interactions as well as consumer requests or complaints.
- A medicine platform might associate doctors to patients to provide an accurate transcript, attach a readable transcript to patient records, or place the transcript into a medical record system.

A typical speaker diarization pipeline typically involves the following steps:

- In the pre-processing stage, the raw audio or video data is cleaned and filtered to remove noise and other interference. This step helps improve the overall accuracy of the speaker diarization system.
- In the segmentation stage, the audio or video data is divided into smaller segments, typically based on the presence of speech. This allows the system to analyze the data in smaller, more manageable chunks.
- In the speaker modeling stage, a model is trained to represent the characteristics of each individual speaker in the data. This model may be based on features such as the speaker's voice, accent, or language.
- In the clustering stage, the speaker models are used to group similar speakers together and label each segment of the data with the corresponding speaker identity.

Once these models and algorithms have been trained, the speaker diarization system can process a new audio or video recording and use the trained models to identify and label the different speakers. This process typically involves analyzing the data in segments, using the trained speaker models to identify the speaker in each segment, and then clustering similar speakers together to label the data.

For the development of the new component, we used PyKaldi to generate raw bindings into Kaldi C++ API and extend those bindings in Python to provide a better user experience. Speaker diarization tasks can be performed by deploying the corresponding diarization models.

4.5.1 Pipeline Creation

As in every ASR project, a file containing short audio segments is produced through segmentation. In this case, our baseline system uses a simple energy-based Voice Activity Detection (VAD). Using data on the average log-energy in a specified window centered on the current frame, the energy VAD determines whether a frame is speech or non-speech.

```
def extract_segments(wav_rspecifier, segments_rxfilename, wav_wspecifier,
    min_segment_length=0.1, max_overshoot=0.5):
    print("Extracting segments from wav file...")

    reader = RandomAccessWaveReader(wav_rspecifier)
    writer = WaveWriter(wav_wspecifier)
    istream = util.io.Input(segments_rxfilename, binary=False)

    num_lines = 0
    num_success = 0

    while 1:
        line = istream.readline()
        if not line:
            break

        num_lines += 1
        split_line = re.split(" |\t|\r", line.strip())

        if len(split_line) != 4 and len(split_line) != 5:
            print("Warning: Invalid line in segments file: " + str(line))
            continue

        segment = split_line[0]
        recording = split_line[1]
        start_str = split_line[2]
        end_str = split_line[3]

        # Parse the start and end times as float values. Segment is ignored if
        # any of end times is malformed.
        start, end = None, None

        start = float(start_str)
        if start is None:
            print(
                "Warning: Invalid line in segments file [bad start]: " + str(
                    line))
            continue

        end = float(end_str)
        if end is None:
            print(
                "Warning: Invalid line in segments file [bad end]: " + str(
                    line))
            continue

        # Start time must be non-negative and not greater than the end time,
        # except if the end time is -1.
        if (start < 0 or (end != -1.0 and end <= 0) or ((start >= end) and (
            end > 0))):
            print(
```

```
                "Warning: Invalid line in segments file [empty or invalid
                    segment]: " + str(line))
            continue

        channel = -1  # -1 means channel is unspecified.
        # If the line has 5 elements, then the 5th element is the channel
            number.
        if len(split_line) == 5:
            channel = int(split_line[4])
            if not channel or channel < 0:
                print(
                    "Warning: Invalid line in segments file [bad channel]: " +
                        str(line))
                continue

        # Check whether the recording ID is in wav.scp; if not, skip the
            segment.
        if not reader.has_key(recording):
            print("Warning: Could not find recording " +
                  str(recording) + ", skipping segment " + str(segment))
            continue

        wave = reader.value(recording)
        wave_data = wave.data()
        samp_freq = wave.samp_freq  # Sampling fequency.
        num_samp = wave_data.num_cols,  # Number of samples in recording.
        num_chan = wave_data.num_rows  # Number of channels in recording.
        file_length = num_samp[0] / samp_freq  # In seconds.

        # Start must be within the wave data, otherwise skip the segment.
        if start < 0 or start > file_length:
            print("Warning: Segment start is out of file data range [0, " +
                str(
                file_length) + "s]; skipping segment '" + str(line) + "'")
            continue

        # End must be less than the file length adjusted for possible
            overshoot;
        # otherwise skip the segment. end == -1 passes the check.
        if end > file_length + max_overshoot:
            print("Warning: Segment end is too far out of file data range [0,"
                + str(
                file_length) + "s]; skipping segment '" + str(line) + "'")
            continue

        # Otherwise ensure the end is not beyond the end of data, and default
        # end == -1 to the end of file data.
        if end < 0 or end > file_length:
            end = file_length

        # Skip if segment size is less than the minimum allowed.
        if end - start < min_segment_length:
            print("Warning: Segment " + str(segment) +
                  " too short, skipping it.")
            continue


        # Check that the channel is specified in the segments file for a multi
            -
        # channel file, and that the channel actually exists in the wave data.
```

```
        if channel == -1:
            if num_chan == 1:
                channel = 0
            else:
                print("Error: Your data has multiple channels. You must
                    specify the channel in the segments file. "
                      "Skipping segment " + str(segment))
        else:
            if channel >= num_chan:
                print("Warning: Invalid channel " + str(channel) + " >= " +
                      str(num_chan) + ". Skipping segment " + str(segment))
                continue

        # Convert endpoints of the segment to sample numbers. Note that the
        # conversion requires a proper rounding.
        start_samp = int(start * samp_freq + 0.5)
        end_samp = int(end * samp_freq + 0.5)

        if end_samp > num_samp[0]:
            end_samp = num_samp[0]

        # Get the range of data from the orignial wave_data matrix.
        segment_matrix = SubMatrix(
            wave_data, channel, 1, start_samp, end_samp - start_samp)
        segment_wave = WaveData.from_data(samp_freq, segment_matrix)
        # Write the range in wave format.
        writer.write(segment, segment_wave)
        num_success += 1

    print("Log: Successfully processed " + str(num_success) +
          " lines out of " + str(num_lines) + " in the segments file. ")
    return 0
```

SOURCE CODE 4.1: Segments extraction

Afterwards, feature extraction for the audio is initiated that will later be the inputs for the x-vectors extractor which essentially creates the x-vectors.

```
# Compute mfcc features
feats_rspecifier = (
    "ark:compute-mfcc-feats --config=/kaldigrpc/model/conf/mfcc.conf scp:/
       kaldigrpc/diarizer_model/data/wav2.scp ark:- |"
)

with MatrixWriter("ark,scp:../diarizer_model/data/feats.ark,../diarizer_model/
   data/feats.scp") as writer:
    for key, feats in SequentialMatrixReader(feats_rspecifier):
        writer[key] = feats
```

SOURCE CODE 4.2: Feature extraction

The next phase will be to extract the x-vectors for the data [147, 148].

```
def extract_vectors(config, apply_cmn=True, window = None):
    print("\nStarting x-vector extraction...")
    feats_rspecifier = "ark:../diarizer_model/data/feats.ark"

    # Stage 0: Set up new xvector features if APPLY CMVN is set
    if apply_cmn:
        rspecifier = (
            "ark:compute-mfcc-feats --config=/kaldigrpc/model/conf/mfcc.conf
               scp:/kaldigrpc/diarizer_model/data/wav2.scp ark:-"
            " | apply-cmvn-sliding --norm-vars=false --cmn-window={} --center=
               true ark:- ark:-"
            " | select-voiced-frames ark:- scp,s,cs:/kaldigrpc/diarizer_model/
               data/vad.scp ark:- |"
        ).format(window)

        with MatrixWriter("ark,scp:../diarizer_model/data/xvectors_cvmn_feats.
           ark, ../diarizer_model/data/xvectors_cvmn_feats.scp") as writer:
            for key, feats in SequentialMatrixReader(rspecifier):
                writer[key] = feats

            feats_rspecifier = "ark:../diarizer_model/data/xvectors_cvmn_feats
               .ark"

    # Stage 1: Extract xvectors from nnet
    NnetXvectorCompute(
        config, feats_rspecifier, "ark,scp:../diarizer_model/data/xvectors.ark
           ,
        ../diarizer_model/data/xvectors.scp")

    # Stage 2: Computing mean of x-vectors
    iVectorMean("ark:/kaldigrpc/diarizer_model/data/xvectors.ark",
                "/kaldigrpc/diarizer_model/data/mean.vec")

    # Stage 3: Computing whitening transform
    if config.pca_dim is None:
        config.pca_dim = -1
    print("\nComputing whitening transform...")
    estPca("ark:/kaldigrpc/diarizer_model/data/xvectors.ark", "/kaldigrpc/
       diarizer_model/data/transform.mat", read_vectors=True, normalize_mean=
       False, normalize_variance=True, dim=config.pca_dim, binary=False)
    print("X-vector extraction complete!")
```

SOURCE CODE 4.3: X-Vector extraction

The scoring of the pair-wise similarity between the x-vectors and the Probabilistic Linear Discriminant Analysis (PLDA) backend [126] is the next stage.

```
def score_plda(config, target_energy=0.1):
    print("\nInitializing PLDA scoring...")

    # Stage 0: Copy a table of iVectors but subtracts the global mean
    ivector_subtract_global_mean(
        "ark:/kaldigrpc/diarizer_model/data/xvectors.ark",
        "/kaldigrpc/diarizer_model/data/mean.vec", "ark:/kaldigrpc/
           diarizer_model/data/subtracted_mean_xvectors.ark")

    # Stage 1: Apply a linear or affine transform to individual vectors
```

```
    transform_vec(
        "/kaldigrpc/diarizer_model/data/transform.mat", "ark:/kaldigrpc/
            diarizer_model/data/subtracted_mean_xvectors.ark",
        "ark:/kaldigrpc/diarizer_model/data/
            transformed_subtracted_mean_xvectors.ark")

    # Stage 2: Normalize length of iVectors to equal sqrt(feature-dimension)
    ivector_normalize_length(
        "ark:/kaldigrpc/diarizer_model/data/
            transformed_subtracted_mean_xvectors.ark",
        "ark:/kaldigrpc/diarizer_model/data/plda_xvectors.ark")

    # Stage 3: Perform PLDA scoring for speaker diarization
    ivector_plda_scoring_dense(
        config.model_cfg.plda,
        "/kaldigrpc/diarizer_model/data/spk2utt",
        "ark:/kaldigrpc/diarizer_model/data/plda_xvectors.ark",
        "ark,scp:/kaldigrpc/diarizer_model/data/scores.ark,/kaldigrpc/
            diarizer_model/data/scores.scp",
        target_energy
    )
```

SOURCE CODE 4.4: PLDA scoring

The last part will be to cluster the PLDA scores that are created using agglomerative hierarchical clustering [55]. Since the number of speakers that each utterance has is unknown, the clustering is run in an unsupervised way and and the threshold can be tuned manually in the script.

```
def apply_clustering(config, rttm_channel=1):
    print("Performing clustering using PLDA scores...")

    os.system(
        "awk '{print $1, '4'}' /kaldigrpc/diarizer_model/data/wav.scp > /
            kaldigrpc/diarizer_model/data/reco2num_spk")

    # Stage 0: Perform agglomerative clustering
    agglomerative_cluster(
        "scp:/kaldigrpc/diarizer_model/data/scores.scp",
        "/kaldigrpc/diarizer_model/data/spk2utt",
        "ark,t:/kaldigrpc/diarizer_model/data/labels",
        "ark,t:/kaldigrpc/diarizer_model/data/reco2num_spk", threshold = 0.4
    )

    # Stage 1: Compute RTTM
    create_rttm(
        "/kaldigrpc/diarizer_model/data/filtered_segments",
        "/kaldigrpc/diarizer_model/data/labels",
        "/kaldigrpc/diarizer_model/data/rttm",
        rttm_channel
    )
```

SOURCE CODE 4.5: Clustering using PLDA scores

After clustering an output file named rttm is generated. The file contents and format look like this:

```
SPEAKER rec1 0 56.200 16.400 <NA> <NA> 1 <NA> <NA>
SPEAKER rec1 0 73.050 5.830 <NA> <NA> 1 <NA> <NA>
SPEAKER rec2 0 79.230 4.270 <NA> <NA> 2 <NA> <NA>
SPEAKER rec1 0 83.760 8.625 <NA> <NA> 1 <NA> <NA>
SPEAKER rec1 0 92.385 4.525 <NA> <NA> 1 <NA> <NA>
SPEAKER rec2 0 97.230 6.230 <NA> <NA> 2 <NA> <NA>
SPEAKER rec2 0 103.820 0.850 <NA> <NA> 2 <NA> <NA>
```

The 2nd column is the recording-id from the wav.scp file, the 4th column is the starting timestamp of the current segment, the 5th column is the duration of the current segment and the 8th column is the ID of the speaker in the specific segment.

The diarization process is practically complete. Now, we may attempt to utilize speech recognition techniques to determine what each speaker said or use speaker verification techniques to confirm whether we are familiar with any of the speakers. A block diagram of the pipeline of the speaker diarization analysis can be seen in detail in Fig. 29. More details on the coding part of the pipeline can be found in the corresponding repository[3].

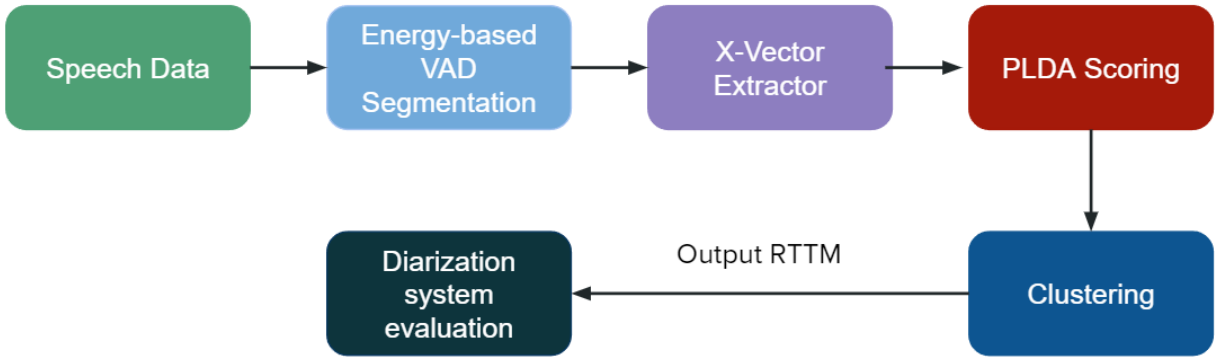

Figure 29: The process of speaker diarization in Kaldi-gRPC-Server

[3]https://github.com/georgepar/kaldi-grpc-server

4.6 Future Work

A number of upcoming features are listed below:

- Add support for mixed kaldi and PyTorch acoustic / language models
- Add full support for pause detection (interim results)
- Add load balancer / benchmarks
- Utilize streamlined conversion scripts from exp folder to model tarball
- Provide support for all Speech API configuration options

5. THE GREC CORPUS

The GREC corpus contains 204 hours of Greek speech. Audio is segmented into individual utterances and each utterance is paired with its corresponding transcription. Table 4 summarizes the included subcorpora, as well as the train, development and test splits. The dataset is constructed with three core principles in mind:

1. **Data Volume**: We collect the largest publicly available speech recognition corpus for the Greek language, able to scale to hundreds of hours of transcribed audio.
2. **Temporal Relevance**: Language changes over time. We aim at an up-to-date corpus that encompasses the latest terms and topics that appear in daily speech.
3. **Multi-Domain Evaluation:** Single domain evaluation can lead to misleading estimations of the expected performance for ASR models. For example, state-of-the-art ASR models [35] achieve under 5% word error rate on Librispeech [118] test sets, but this is an over-estimation of system performance in the field. This is extenuated when considering different acoustic conditions or terminology. We consider multi-domain evaluation essential when developing and deploying real-world ASR models.

To satisfy the first two points, we collect data from a public, continuously updated resource, i.e. the Hellenic Parliament Proceedings, where recordings of the parliamentary sessions are regularly uploaded. The benefit of using this resource is the straight-forward collection of a continuously growing, multi-speaker corpus of transcribed audio that is always up-to-date, as the parliamentary discussions revolve around current affairs. We refer to this corpus as HParl. For the multi-domain evaluation, we merge HParl with three other corpora that have different acoustic and language characteristics. We refer to the merged, multi-domain corpus as GREC. In the next section, we will describe the collection and curation process of HParl, and present the relevant statistics for the challenge.

Table 4: The GREC corpus. We can see the duration of each split in `hours : minutes : seconds` format, as well as the number of speakers for each of the sub-corpora.

Dataset	Domain	Speakers	Train	Dev	Test	Total Duration
HParl	Public (political) speech	387	95:23:43	12:25:19	11:58:39	119:47:41
CV9	Dictated speech	325	10:45:18	01:43:29	01:45:15	14:14:02
Logotypografia	News casts	125	48:43:01	08:31:10	08:22:27	65:36:38
CSS10	Audiobooks	1	02:12:48	00:23:37	01:15:43	03:52:08
All	-	838	157:04:50	23:03:35	23:22:04	203:30:29

5.1 Collection and Curation of HParl

Modern technological advances allow for more direct government transparency, through the commodification of storage and internet speeds. Consequently, the records of plenary sessions of the Hellenic Parliament are made publicly available for direct access through a webpage[1]. The available video recordings date back to 2015. For each plenary session, a video recording is uploaded, along with a full transcription that is recorded verbatim, and in real time by the parliament secretaries. During the creation of HParl, we build a web-crawler that can traverse and download the video recordings, along with the transcriptions from the official website. The collection process is parallelized over multiple threads, and parameterized by a range of dates and, optionally, a target corpus size in GB or in hours. For this version of HParl, we collect the plenary sessions from four separate date ranges, as described in Table 5. The majority of the collected sessions are from 2019, but we also include sessions from 2018 and 2022 to facilitate coverage of several topics and incorporate the latest mediaspeak in Greek. The individual core components of the HParl curation pipeline as seen in Fig. 30 are the audio pre-processing, text pre-processing, alignment, post-processing and dataset splitting.

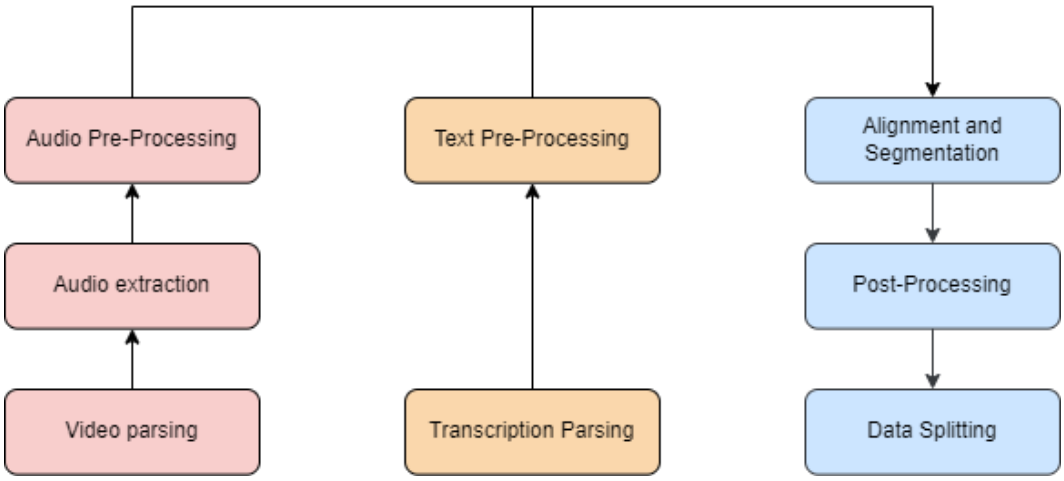

Figure 30: The structure of the parliamentary data pipeline

Table 5: Plenary sessions included in HParl. The Hours column refers to the raw (unsegmented) hours of collected audio.

Start date	End date	#Sessions	Hours
15-02-2022	01-03-2022	10	55
18-01-2019	01-02-2019	10	52
28-03-2019	10-05-2019	20	108
10-12-2018	21-12-2018	10	88

[1] https://www.hellenicparliament.gr/en/

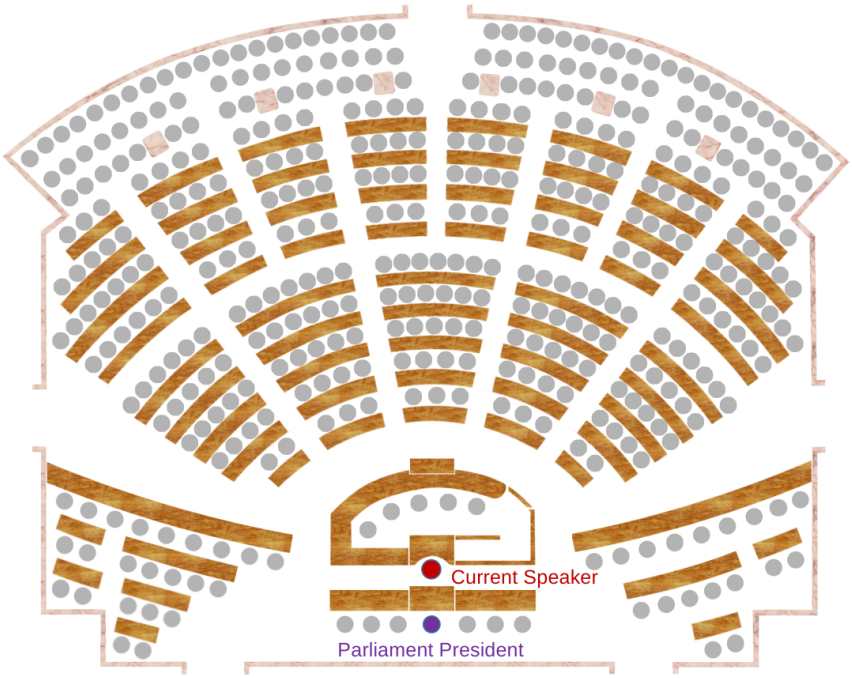

Figure 31: Overview of the Hellenic Parliament Chamber. The chamber has an amphitheatrical shape and can accomodate approximately $400 - 450$ people. The positions of the key speakers, i.e. current speaker and the parliament president are annotated in the image.

5.1.1 Audio Pre-processing

Fig. 31 shows an overview of the Hellenic Parliament Chamber. Plenary sessions take place mainly in this room, or in the secondary House Chamber that has similar setup but is smaller in size. Because of the room and microphone characteristics, the captured audio in the video streams contains reverberation, due to sound reflections. We employ a light preprocessing pipeline, by passing the input video streams through FFmpeg, and converting them to monophonic, lossless audio format at $16,000$ Hz sampling rate. The resulting audio is not passed through any de-reverberation or speech enhancement software. The resulting audio files have a minimum, average and maximum duration of 6 minutes, 6 hours and 16 hours respectively.

5.1.2 Text Pre-processing

The text files contain full, word-by-word transcription of the speeches and questions asked by members of the audience, as well as extra annotations made by the parliament secretaries. Some annotations are relevant, i.e. the speaker name, while others are plain text descriptions of events happening during the session and need to be filtered out (e.g. “The session is interrupted for a 15 minute break”). We use a rule-based system, based on regular expressions, that filters the unnecessary information, keeping only the transcriptions and the speaker names. The speaker labels are created by transliterating their names and roles from Greek to Greeklish using the “All Greek to Me!” tool [17]. Text is lower-cased and normalized to remove multiple whitespaces. The result is a text file containing the raw transcriptions, and a mapping from speaker labels to their respective text parts.

5.1.3 Alignment and Segmentation

The primary challenge of exploiting the plenary sessions for ASR purposes is the length of the plenary recordings, as their durations vary from 6 minutes to 16 hours in length. However, data samples used to train ASR are generally less than 30 seconds long. Computational challenges have limited the length of training utterances for HMM-GMM models [101], and continue to do so in the contemporary neural network models. Therefore, we need to segment the sessions into smaller pieces that are more suitable for ASR training. A second challenge is posed by the presence of mismatches between audio and transcripts. Parliamentary proceedings do not fully capture everything that is said during the parliamentary sessions, and do not account for speech disfluencies.

In order to obtain smaller, clean segments, that are suitable for ASR training we follow the segmentation procedure proposed by [97]. Initially the raw recordings are segmented into 30 second segments and the transcriptions are split into smaller segments of approximately 1000 words called *documents*. Each segment is decoded using a seed acoustic model trained on the Logotypografia corpus [29] and a 4-gram biased LM trained on the corresponding transcription of each recording. The best path transcript of each segment is obtained and paired with the best matching *document* via TF-IDF similarity. Finally each hypothesis is aligned with the transcription using Smith-Waterman alignment [146] to select the best matching sub-sequence of words. The above method yields a list of text utterances, with their corresponding start and end times in the source audio files. Fig. 32 shows a block diagram of a basic HMM based segmentation/alignment pipeline.

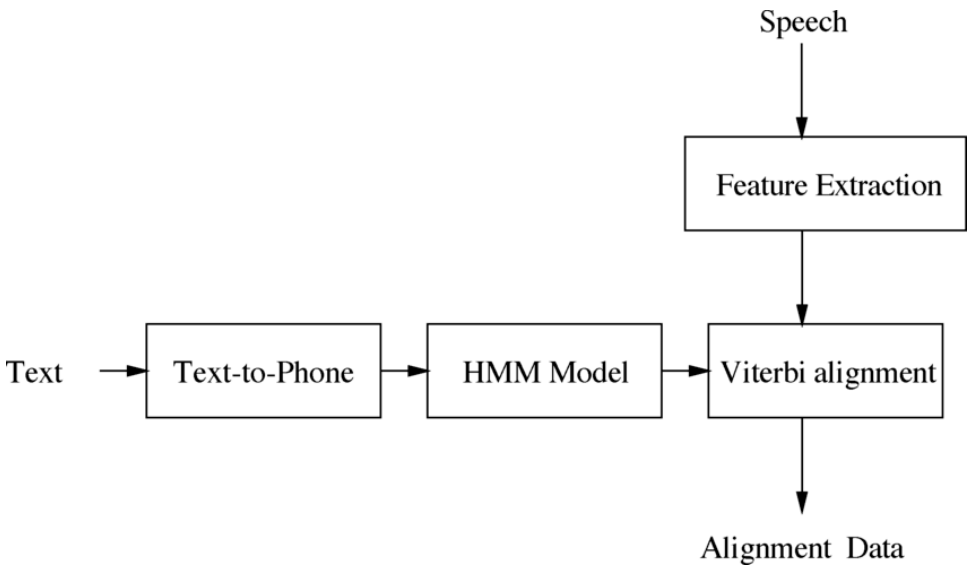

Figure 32: Block diagram of an HMM based segmentation/alignment system

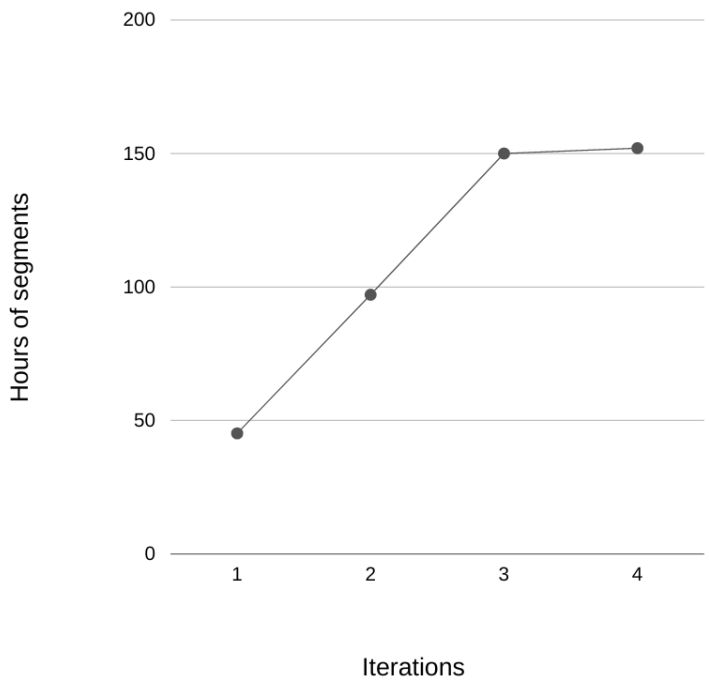

Figure 33: The rate of increase of segmented hours with respect to the number of iterations

When this course of actions finishes, the output segments are used to train an acoustic model. Following this, the acoustic model is reused to re-segment the original data and produce more segments and at the same time more accurate ones. The entire process, as depicted in Fig. 34, is refined through multiple iterations until the amount of total segments converges. During this procedure, 4 repetitions took place, yielding 120 hours of useable segmented utterances out of the original 303 hours of raw audio, or a ratio of 39.6%. Fig. 33 shows the rate in which the number of segmented hours increased relatively to the iterations applied.

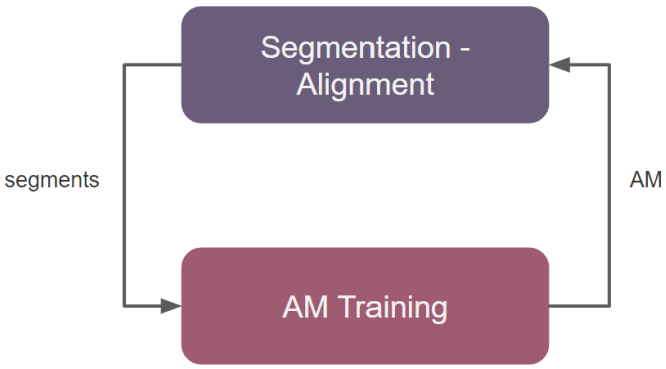

Figure 34: The re-segmentation process of the HParl dataset

5.1.4 Post-processing

After the segments are extracted, we filter out extremely short segments (less than 2 words). Moreover, the iterative alignment algorithm may replace some intermediate words with a `<spoken-noise>` tag. When this tag is inserted, we match the surrounding text with the raw transcriptions and re-insert the missing words. Furthermore, we match each segment to its corresponding speaker label. Segments without a speaker label are discarded.

An additional characteristic of the acoustic data is the gender information. By assigning gender labels to the corresponding speaker labels, it is feasible to train gender classifiers or enhance the accuracy of a speaker-dependent ASR system. In this work, speakers are associated to their gender based on name suffixes, using a simple rule: Speaker names which end in a(α), h(η), w(ω) or is(ις) are classified as female, while the rest as male. We format the segments, speaker and gender mappings in the standard folder structure used by the Kaldi speech recognition toolkit [125].

5.1.5 Data Expansion

One advantage of utilizing this publicly accessible resource is its potential to stay current with the evolving language and vocabulary brought on by advancements in the world. A continuous data collection pipeline can be established, leveraging the HParl corpus as new proceedings are uploaded, providing a streamlined expansion of the parliamentary data. The Hellenic Parliament website houses approximately ten years of speech data, which if fully utilized, would result in a corpus of approximately $10,000$ hours of speech. Further expansion is feasible, as nearly 17 plenary sessions, averaging 3.5 hours each, take place every month. At this rate, the dataset can grow by $1,000$ hours annually. For this study, a subset of the available speeches from this time period was used due to computational constraints.

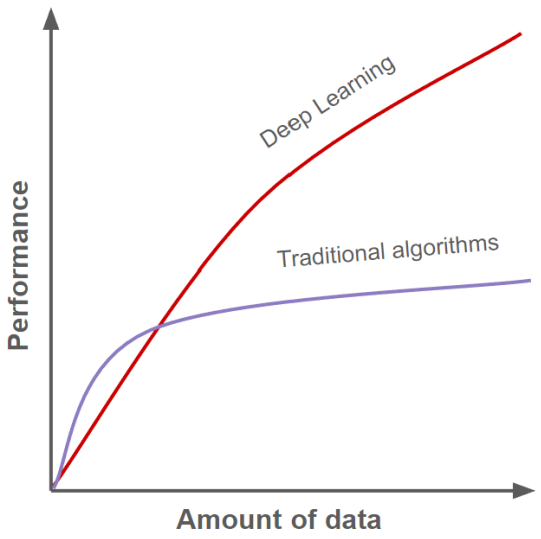

Figure 35: Deep learning vs Traditional ML

5.1.6 Data Splitting

Speaker-independent (SID) systems are desirable in many applications where speaker-specific data do not exist. For this reason, a separate SID development set, comprised of 2 speakers and spanning 1 hour, as well as an evaluation set comprised of 4 speakers and spanning 2 hours were isolated for the purpose of evaluating an ASR system on speakers not present in any of the training data. The remained data composed the Speaker-Dependent (SD) dataset which was split into train, development and evaluation sets, yet, distributing respectively male and female speakers. Table 6 shows the data splitting of HParl into train, development and test sets, following a ratio of 80%-10%-10% respectively. The development set contains 12 hours of segmented speech. Similarly, the test set contains 12 hours of segmented speech. The rest 95 hours of segmented speech are assigned to the training set. The splitting was performed under no consideration of the time frame of the plenary sessions. In a future version, the dataset will be splitted evenly across the time periods to facilitate a more proper evaluation. Overlapping speech was calculated as well, spanning 1 hour.

Table 6: Time duration of HParl training components across the 3 different time periods and overlap calculation.

Duration type	Train	Dev	Test
Total duration	95:23:43	12:25:19	11:58:39
Total duration 2018	35:00:28	00:00:00	00:00:00
Total duration 2019	50:51:36	00:00:00	10:40:32
Total duration 2022	08:36:11	12:16:51	01:11:36
Total duration without overlap	94:28:15	12:16:51	11:52:09
Overlap	00:55:28	00:08:27	00:06:30

5.2 Including corpora from different domains

Automated speech recognition systems still have difficulty dealing with variance between the training and test domains. There is a critical need for cross-domain training data to address the problem of domain shift. A small number of efforts have been made in processing and modelling the Greek language in domain adaptation scenarios, thus, it is necessary to conduct further research and propel advancement of the existing technology. The GREC dataset is our approach in combining data from different domains and speech scenarios and incorporate them to the core dataset containing the parliamentary data as a means of generating a multi-domain corpus in Modern Greek. To achieve this, we merge our brand new HParl corpus with Logotypografia, Common Voice (version 9) and the audiobook "A Tale Without a Name" from CSS10. Table 4 shows the different components of GREC and their corresponding duration. The next sections describe each individual corpus that formulates GREC.

5.2.1 Common Voice - Short Speech

Common Voice is a well-known initiative to enable the general public have open access to voice recognition technology [5]. It is a corpus of speech data read by users on the Common Voice website[2], and based upon text from a variety of public domain sources including user submitted blog posts, old books, movies, and other public speech corpora. The primary purpose of it is to enable the training and testing of automatic speech recognition systems, but can be utilized for other purposes as well.

The data collection is performed with the use of a web app or a mobile app. Contributors are presented with a prompt of maximum 15 words and are asked to read it. The prompts are taken from public domain sources, i.e. books, wikipedia, user submitted prompts and other public corpora. A rating system is built into the platform, where contributors can upvote or downvote submitted `<audio,transcript>` pairs. A pair is considered valid, if it receives two upvotes. Speaker independent train, development and test splits are provided.

The dataset is open to the research community, released under a permissive Creative Commons license (CC0). As of November 2019 there were a total of 38 languages collecting data, but the most recent release includes 100 languages. To date, more than $50,000$ people have taken part, contributing to $2,500$ hours of audio. In terms of the number of hours and languages, this is the largest audio corpus available in the public domain for speech recognition. For the purpose of showcasing Common Voice, Ardila et al. utilized Mozilla's DeepSpeech Speech-to-Text toolkit [56] for speech recognition experiments. By applying transfer learning from a source English model, they found an average Character Error Rate improvement of 5.99 ± 5.48 for twelve individual languages (Slovenian, German, Italian, Turkish, French, Welsh, Chuvash, Catalan, Irish, Breton, Tatar, and Kabyle). For most of these languages, these were the first ever published results on end-to-end Automatic Speech Recognition.

In this work, we use version 9.0 of CV, accessed on April 27, 2022. We keep only the valid utterances, i.e. 16 hours of speech from 325 contributors ($19-49$ years old, 67% male / 23% female). Speakers' age distribution is shown in fig. 36.

[2] `https://commonvoice.mozilla.org/`

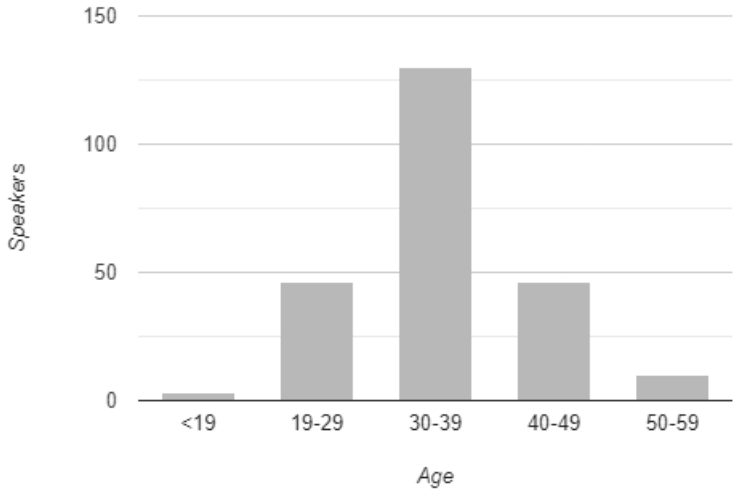

Figure 36: Speakers' age distribution

5.2.2 Logotypografia - Newscasts

"Logotypografia" is the first Greek speech corpus in the field of journalism. The work was carried out as a part of a project named Logotypografia (Logos = logos, speech and Typografia =typography) and was funded by the General Secretariat of Research and Development of Greece. [29]

The project's objective was to establish the infrastructure for continuous speech recognition in Modern Greek with the help of one of the most established journalistic organizations in Greece, "Eleftherotypia". This included adapting an efficient speech recognition system involving a large lexicon and its installation in the editing environment of the "Eleftherotypia" newspaper, research and develop in relation to the systems involving large recognizability and laying the groundwork for the development of a complete Greek Speech Recognition system. Having accomplished the aforementioned goals, Digalakis et al. then integrated a large vocabulary, speaker independent Greek dictation system using the SRI Decipher speech recognition engine [112].

The speakers that participated in the recording sessions were employees of "Eleftherotypia". More specifically, 291 sessions were completed by 125 speakers, with the participation of 55 male speakers and 70 female speakers, with a distribution of 44% to 56%, respectively. Fig. 37 depicts the age distribution of the speakers. Three separate settings—an office, a quiet area, and a soundproof room—were used for the recordings. This information is summarized in Fig. 38. After discarding 25% of bad utterances, the total collected utterances lasted an average of 7.8 seconds and comprised of around 72 hours of speaking. Spontaneous speech from 30 more sessions was captured as well. The corpus was divided into two sets, one with $23,136$ utterances that were transcribed by the Technical University of Crete's speech recognition group and one with $10,000$ utterances that were transcribed by the Institute of Language and Speech Processing in Athens.

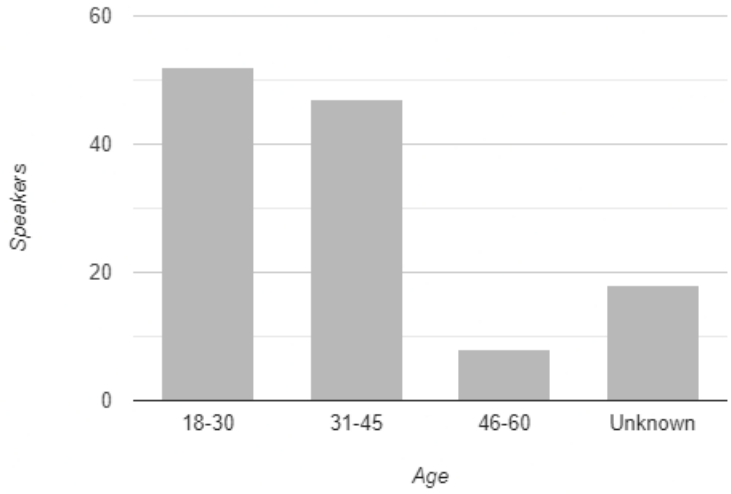

Figure 37: Speakers' age distribution

	Sound Proof Room	Quiet Room	Office Room	
Number of Utterances	180	150	150	
Number of Sessions	79	110	102	
Total Utterances	14,220	16,500	15,300	**46,020**

Figure 38: Utterances in different settings

5.2.3 CSS10 - Audiobook

CSS10 [120] is a multi-lingual collection of single-speaker speech datasets primarily aimed for Text-To-Speech applications. Data are collected from public domain audiobooks [3]. Park et al. constructed two neural text-to-speech (TTS) models—Tacotron [164] and Deep Convolutional TTS (DCTTS) [153]—on each dataset and ran Mean Opinion Score tests on the synthesized voice samples to confirm the system's high quality. The audio from LibriVox typically comes in large files with long audio segments that are not appropriate for the TTS task, therefore they divided them into many smaller files. They used the audio editor Audacity[4] to automatically detect split-points of the audio whenever there was more than a 0.5-second duration of silence, with the exception of Spanish audiobooks, where a 0.25-second length was employed. The positions were then changed such that adjacent clips could be combined to achieve a runtime of around 10 seconds. They discovered that these techniques enhanced computing efficiency.

In this work, we utilized the Greek part of CSS10 which contains an audiobook of a classic kid's tale called "A Tale without a Name" by Pinelopi Delta, resulting to 4 hours of speech. Since the narrator is a single-speaker, the speech data can easily be characterized as clear and comprehensible which makes it suitable for ASR purposes. Fig. 39 displays the distribution of audio lengths for the Spanish dataset. About 85% of the samples have a duration between 5 and 11 seconds. All other languages, including Greek, have distributions similar to this.

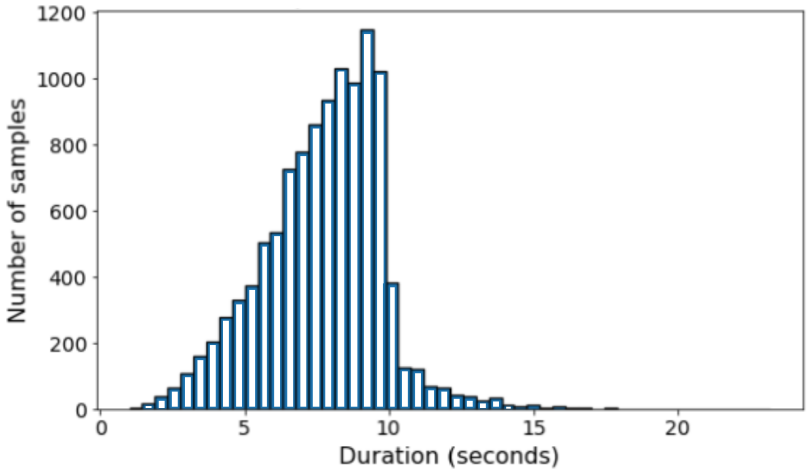

Figure 39: Audio durations for the Spanish dataset. All other languages, including the Greek, have distributions similar to this.

[3] https://librivox.org/
[4] https://www.audacityteam.org/

6. EVALUATION EXPERIMENTS

For performing experiments on ASR tasks, all subcorpora of the GREC dataset were properly curated as per Kaldi format which includes the general files wav.scp, utt2spk, spk2utt and text. It is extremely convenient to use this kind of data format as the time-consuming stage of data preparation when using Kaldi pipelines can easily be skipped. Apart from that, inside each subset there is a directory with all the necessary audio files that were converted to .wav format with sampling rate of 16 KHz in single channel mode. On top of that, all data were splitted to train, development, evaluation sets following the 80%-10%-10% pattern. Experiments were performed on an ILSP server with an AMD EPYC 7443 24-Core Processor, equipped with 4 NVIDIA GeForce RTX 3090 24GB GDDR6X and 128GB RAM. We considered two evaluation benchmark setups. In section 6.1, we first discuss the ASR setup following with the results of the experiments listed in Table 7, four of which adopt a GMM-HMM ASR architecture and the last adopting a DNN-HMM architecture. Next, in section 6.2, we present the speaker diarization setting with the experimental results displayed in Table 8.

6.1 ASR Evaluation Setup

During the experiments, several variations of the GREC dataset were generated for the training stage by consolidating the subcorpora in various combinations. Evaluation was performed on each individual subcorpora of GREC in order to assess the performance of GREC on the domain-shift challenge.

For acoustic model training, we used a modified Wall Street Journal recipe [121] for Greek in Kaldi provided by ILSP. First, speech utterances go through a typical feature extraction process. The audio input is divided into short frames and for each frame 13-dimensional MFCCs are extracted. The resulting features, together with their first and second order derivatives, are then arranged into a single observation vector of 39 components. Following that, cepstral mean and variance normalization is performed. A monophone system GMM-HMM (tri1) is trained and all the data are then aligned using this system. A first delta + delta-delta triphone model (tri2b) training on the full training set follows, and the data is re-aligned using the monophone model. Features are spliced together, projected to 40-dimensional space using linear discriminant analysis, and a de-correlation based on the maximum likelihood linear transform is applied leading to training an LDA+MLLT system. Again, data is re-aligned using the tri2b model. In the last step, the model is retrained using speaker adaptive training (LDA+MLLT+SAT) in order to become independent of training speakers and generalize better to unseen testing speakers. The system is then tested with silence and pronunciation probabilities and decoding initiates.

Based on the tri3b model, we train a DNN-HMM architecture [122] following the process described in the mini-librispeech recipe in Kaldi. 40 Mel-Frequency Cepstral Co-efficients (MFCCs) and 100-dimensional iVector features are computed in each time-step. Standard speed and volume perturbation techniques are applied for data augmentation. For the DNN network we use input Batch Normalization followed by 12 factorized TDNN layers [123] and 1 feed-forward layer. The baseline system is depicted in Fig. 40.

For language modeling, the SRILM toolkit was employed to build a 3-gram language model [151] using a dictionary-lexicon data collection provided by the ILSP with a vocabulary size of approximately $132K$ words.

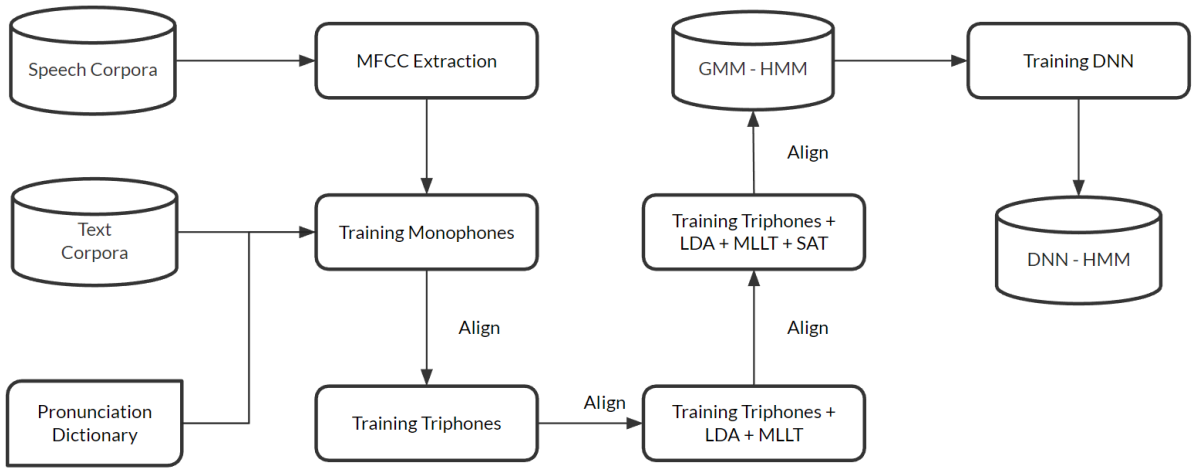

Figure 40: Training stages of a traditional GMM-HMM triphone-based and a hybrid HMM-DNN acoustic model on Kaldi

6.1.1 Metric

The performance of the ASR system was evaluated in function of the error rate. The purpose of ASR evaluation is to provide a comparison criterion between different systems or methods and measure the performance and the progress on specific tasks based on errors statistics. There are two areas of interest related to ASR errors. The first one is the reference-recognised alignment which consists of finding the best word alignment between the reference and the automatic transcription and the second one being the evaluation metrics measuring the performance of the ASR systems such as this ASR setup. Word Error Rate is the most popular metric for ASR evaluation, measuring the percentage of incorrect words regarding the total number of words processed. It is defined as

$$WER = \frac{S + I + D}{N} \tag{6.1}$$

where:

- **S** – the word substitutions
- **I** – the word insertions
- **D** – the word deletions
- **N** – the total number of words spoken.

Table 7: Word error rate on evaluation sets with the baseline acoustic models and the TDNN trained on the GREC dataset and several variations of it.

Train Corpus	Model	Evaluation Corpus (WER)			
		HParl	CV9	Logotypografia	CSS10
All (GREC)	Monophones	26.56	40.98	48.10	35.41
	Triphones	17.65	28.85	35.43	26.34
	Triphones + LDA + MLLT	16.56	26.29	33.58	26.15
	Triphones + LDA + MLLT + SAT	15.03	23.17	31.25	25.09
	Deep Neural Network	13.49	11.55	26.70	17.08
All excluding HParl	Monophones	43.20	37.58	46.85	32.55
	Triphones	34.53	26.96	35.13	24.55
	Triphones + LDA + MLLT	35.33	24.94	33.68	24.38
	Triphones + LDA + MLLT + SAT	23.99	23.37	32.06	24.33
	Deep Neural Network	18.77	10.67	27.12	16.07
All excluding CV9	Monophones	26.35	49.29	47.84	43.14
	Triphones	17.72	39.00	35.14	33.23
	Triphones + LDA + MLLT	16.37	39.15	33.78	32.58
	Triphones + LDA + MLLT + SAT	15.04	38.47	31.23	32.01
	Deep Neural Network	13.39	16.66	26.93	22.64
All excluding Logotypografia	Monophones	26.72	54.79	58.87	43.88
	Triphones	18.28	35.53	46.44	27.56
	Triphones + LDA + MLLT	17.05	27.48	45.18	25.93
	Triphones + LDA + MLLT + SAT	16.00	24.26	40.40	23.16
	Deep Neural Network	14.26	11.19	36.54	17.78
All excluding CSS10	Monophones	26.20	40.74	48.11	37.55
	Triphones	17.69	28.39	35.64	28.73
	Triphones + LDA + MLLT	16.44	26.82	33.73	28.70
	Triphones + LDA + MLLT + SAT	15.09	24.54	31.38	26.83
	Deep Neural Network	13.46	11.63	26.75	19.30
HParl	Monophones	26.40	73.06	60.13	74.23
	Triphones	18.50	67.23	51.30	68.78
	Triphones + LDA + MLLT	17.23	69.32	51.14	70.08
	Triphones + LDA + MLLT + SAT	16.04	71.06	41.52	53.95
	Deep Neural Network	14.27	38.37	37.49	44.99
CV9	Monophones	82.43	21.21	81.88	41.19
	Triphones	78.09	19.69	74.44	35.62
	Triphones + LDA + MLLT	78.39	19.95	73.66	38.12
	Triphones + LDA + MLLT + SAT	72.39	19.87	72.01	25.44
	Deep Neural Network	66.53	11.13	18.44	18.44
Logotypografia	Monophones	50.45	69.24	46.98	54.49
	Triphones	43.75	64.28	35.27	45.64
	Triphones + LDA + MLLT	56.57	71.57	33.82	45.91
	Triphones + LDA + MLLT + SAT	30.60	72.60	32.11	45.15
	Deep Neural Network	19.62	32.71	26.91	35.62
CSS10	Monophones	96.91	93.63	87.12	43.88
	Triphones	99.03	98.44	86.88	41.38
	Triphones + LDA + MLLT	99.66	99.40	87.06	40.54
	Triphones + LDA + MLLT + SAT	99.31	99.28	85.20	41.46
	Deep Neural Network	90.63	50.82	75.92	30.36

6.1.2 Results

In Table 7 we report the baseline word error rate results obtained by Kaldi models. During the first experiment, the entirety of GREC was used as training input to the models. The results derived from this experiment serve as the main baseline of this work and as the core essence to draw conclusions about the UDA ability of GREC for ASR models. The next four experiments involve variations of the GREC dataset as training data by excluding one subset at a time, while for the last four experiments, the training set consists of each individual subcorpus. Evaluation is performed on all four subcorpora separately.

As might be expected, all experiments showed a decrease in WER within every step of training the traditional GMM-HMM acoustic models. Needless to say, the DNN-HMM hybrid approach demonstrated the best performance in all cases by reducing significantly the error rate of each triphone (LDA + MLLT + SAT) model. This is not surprising likewise considering the information provided in section 2.5.2.3 on why DNNs perform better than GMMs when paired with HMMs in automatic speech recognition tasks. Therefore, the following points of discussion mainly refer to the results derived from systems of the DNN architecture.

For the evaluation of HParl the system that performed best was the DNN trained on all subcorpora excluding CV9 achieving 13.39% WER. In the opposite direction, the best WER score for the evaluation of CV9 was reported by the DNN model trained on all subcorpora excluding HParl with 10.67%. It is worth underlining the fact that, at first look, both corpora appear to be closely correlated with the aforementioned experiments suggesting their ”inversely proportional” effect to accuracy. This may be justified by their entirely different domain type, which can create a clash between them during evaluation. HParl contains long political speeches in a large parliamentary room that may include reverberation, background noise, distance speaking and conversational interactions such overlaps and unexpected interruptions. On the opposite, CV9 includes short clean crowd-sourced single-speaker uninterrupted speech sharing no common grounds with the audio recordings of HParl. CSS10 was best evaluated when HParl was discarded from the training corpus reporting 16.07% WER, slightly less than 17.08% WER by utilizing the complete GREC corpus. Generally, the presence of CV9 improves significantly the evaluation accuracy of CSS10. This can be further confirmed from the 3rd experiment in conjunction with the 7th. Last but not least, for the evaluation of Logotypografia, CV9 as the only training corpus of the DNN model presented the best performance with 18.44% WER.

With regard to the 2nd-5th experiment, word error rate for the subcorpus that was skipped from the training data was expectedly higher than the initial experiment, but still not to a great extent since the other subcorpora filled the gap in the training data. In the absence of HParl, WER for Logotypografia relatively increased by 2% and 29% for the HParl test set. On the contrary, WER for CV9 was improved relatively by 8% and for CSS10 by 6%, validating once again the effect of the domain shift on the performance. Note that when excluded CSS10 or CV9 from the training data the performance of the systems were barely affected suggesting that the rest of subcorpora are dominant in the GREC corpus with reference to their total volume and, consequently, correlation. Lastly, excluding Logotypografia resulted in increasing the total word error rate for all test sets but the CV9 one.

For the last four experiments, Logotypografia as exclusive training input demonstrates the best performance in evaluating the other subcorpora with HParl, CV9 and CSS10 following next. Even so, the evaluation of Logotypografia test set reported 26.91% WER which could

not outperform GREC achieving 26.70% WER. In a similar way, HParl performance on evaluating itself resulted in 14.27% WER, in comparison with 13.49% WER of GREC on HParl and CSS10 resulted in 30.36%, in comparison with 17.08% of GREC on CSS10. On the other hand, CV9 reported 11.13% WER, which is slightly better than GREC with 11.55%. Interestingly, CV9 and CSS10 performed poorly on the evaluation of HParl with 66.53% and 90.63% WER respectively which is likely due to the amount and nature of their data. In point of fact, CSS10 performed poorly on all other subcorpora with 50.82% on CV9 and 75.92% on Logotypografia.

In the first experiment the performance of the models would be expected to improve significantly with the utilization of all subcorpora in the training data. However, the WER reduction compared to the 2nd experiment where HParl is discarded is negligible. Given that HParl is the only subcorpus with truly noisy data, it makes sense that the evaluation on the remaining subcorpora is underwhelming. It's highly probable, although, that the performance of GREC will stand out while analyzing data obtained from other noisy domains. Ultimately, the purpose of the present work is to create a multi-domain corpus that can be employed in different evaluation scenarios with unforeseen recording conditions. In terms of correlation to the training corpora combinations, Logotypografia stands out as the "odd" one in every instance, meaning that performance isn't really affected significantly by the training data. This could be the case because, even though Logotypografia doesn't include spontaneous speech, the speakers of each broadcast may vary depending on the news while the recording settings of the speakers that aren't present in the production studio can be unpredictable. Apart from that, the remaining test sets appear to be heavily affected by the input data and demonstrate a common performance fluctuation. As mentioned before, CV9 and HParl domains are mutually exclusive in terms of recording and speaker conditions. The same applies to the CSS10 and HParl pair to a certain degree.

The results obtained from the models trained on GREC for the automatic speech recognition (ASR) task are a promising indication of the potential for creating a standardized test-bed for unsupervised domain adaptation. It is important to note that these results serve only as baseline systems for the Kaldi toolkit and do not represent the current state-of-the-art performance of Kaldi or other available toolkits. Further improvement in the results could be achieved through proper tuning of hyperparameters at each step, and as the amount of data increases, performance is expected to significantly improve.

6.2 Speaker Diarization Evaluation Setup

Along with ASR experimentation, the dataset was evaluated in a diarization setup based on the kaldi DIHARD second challenge recipe [137]. The second in a series of diarization competitions concentrating on ”hard” diarization was called the DIHARD II. The essence of the challenge was to perform successfully speaker diarization to challenging recordings where it was anticipated that the state-of-the-art wouldn't perform well. The diarization baseline is based on the previously published Kaldi [124] recipe[1] for JHU's submission to DIHARD I [143].

At a high level, the system performs diarization by dividing each recording into short overlapping segments, extracting x-vectors, scoring with probabilistic linear discriminant analysis (PLDA), and clustering using agglomerative hierarchical clustering (AHC). In contrast to the original JHU system, the Variational Bayes resegmentation step is ommited [28]. As the pipeline was not capable of handling the overlapping speech that was present in the existing segments data, the segments were recreated from scratch using an energy-based speech activity detection (SAD) system. Data augmentation step was also removed from the procedure in order to establish a basic reliable pipeline without data optimization techniques.

The HParl subcorpus is the only suitable corpus for diarization system training as it features audio data with long duration, multiple speakers, clear speech separations and limited overlaps. The other three subcorpora are unsuitable for training with limited single-speaker speech. Experiments including these subcorpora were also conducted nonetheless to observe the diarization system's behavior and gain insights into its limitations and capabilities, despite their inapplicability in real-life scenarios. The development and evaluation sets were formed by concatenating all subcorpora. The development set was used as the validation set to tune the parameters. The DER was computed on the evaluation set and was calculated with no unscored collars and including overlapping speech.

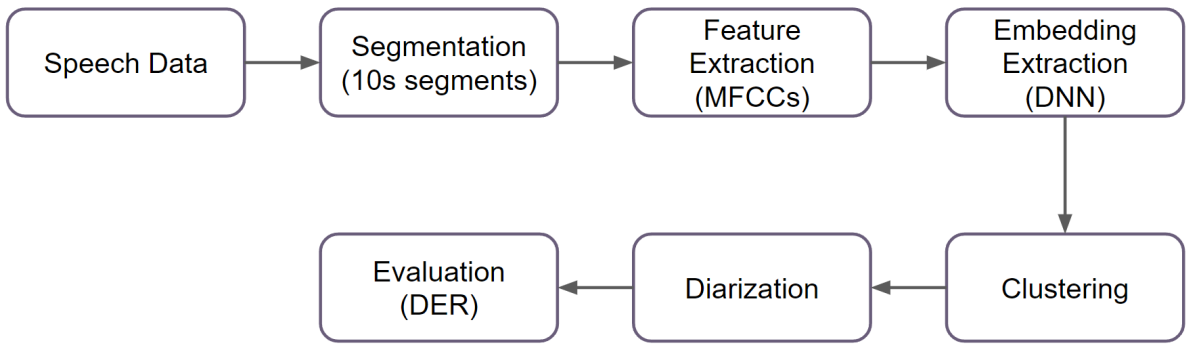

Figure 41: The diarization evaluation pipeline

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/dihard_2018/v2

6.2.1 Metric

Our methods were evaluated with Diarization Error Rate (DER), a common metric for diarization [145], as described and used by NIST in the RT evaluations [42]. The metric is computed in two steps: the first step is to establish a mapping between the speaker tags provided by the system and the speaker identities found in the reference. The second step then computes the error rate using that mapping. DER is defined as in Equation 6.2:

$$DER = \frac{confusion + missed\ detection + false\ alarm}{total} \tag{6.2}$$

where:

- **false alarm** – the duration of non-speech incorrectly classified as speech
- **missed detection** – the duration of speech incorrectly classified as non-speech
- **confusion** – the duration of speaker confusion, and total is the sum over all speakers of their reference speech duration
- **total** – the total length of the reference (ground truth)

Table 8: DER (%) on the GREC corpus.

Train Corpus	Development	Evaluation
All (GREC)	41.30	25.23
All excluding HParl	45.19	23.28
All excluding CV9	15.47	5.66
All excluding Logotypografia	63.03	50.74
All excluding CSS10	47.45	25.55
Logotypografia, CSS10	12.27	4.43
Logotypografia, HParl	9.40	4.72

It is important to note that a score of zero in the context of word error rate signifies flawless performance, whereas higher scores, which may surpass 100, correspond to a reduction in performance quality. For a comprehensive understanding of this concept, it is recommended to refer to section 6.1 of the NIST RT-09 evaluation plan.

6.2.2 Results

DER results of the baseline system on both the development and evaluation sets for train corpus are presented in Table 8. It is evident that the performance of the system varies greatly across the different variations of GREC.

The strongest impact in the performance of a diarization system derives from the speaker's speech duration. If the speech duration is relatively short (i.e half a minute or less), it is debatable whether a speaker diarization model can accurately identify a speaker as a distinct, individual speaker. If it fails, one of two things may happen: either the speaker would be labeled as ”unknown” or their utterances would be integrated with those of a more dominant speaker.

The second important factor of diarization accuracy level is the pace and nature of the communication, with conversational interaction being the most straightforward to properly diarize. It is considerably more likely that the model will properly identify each speaker when the dialogue is well defined, with each speaker taking turns speaking clearly, with no one talking over the speaker or interrupting, and with little to no background noise. The model's accuracy will, however, decline if the dialogue is more dynamic, includes speakers who frequently interrupt or talk over one another, or has excessively loud background noise. If there is a lot of speech overlap, the model could even mistakenly detect a third speaker whose speech contents are the actual overlapping speech segments. One potential challenge in using the HParl subcorpus as the primary training input for the speaker diarization system is the occurrence of multiple overlapping speech segments in its audio data. It is worth noting that the audio data of HParl may also comprise of speech that is spontaneous in nature, as well as recovered voices, instances of disfluencies, speakers that are situated at a distance, and background noise. These factors must be taken into consideration when utilizing HParl as the core input for training the speaker diarization system.

It is imperative to re-emphasize that the HParl subcorpus is the sole corpus that is deemed suitable for training the speaker diarization system, as it comprises audio data of prolonged duration, with multiple speakers, clear separations between speech segments, and limited instances of overlapping speech. Conversely, the remaining three subcorpora feature limited single-speaker speech in each audio file, rendering them unsuitable for use as training input for the diarization system. Despite being inappropriate in a realistic application scenario, we conducted experiments that included them in order to examine the performance of the diarization system. This was to observe and understand the behavior of the diarization system when faced with suboptimal training data, despite the lack of applicability to practical scenarios. By including these subcorpora, valuable insights could be gained into the limitations and capabilities of the diarization system, which could be used to guide further research and development towards achieving a robust and effective speaker diarization system.

The diarization performance of the baseline system on the development set gives a DER of 41.30%, while on the evaluation set improves dramatically with the value of 25.23%. It is important to note that, political speeches, by their nature, tend to exhibit overlapping speech patterns as they often resemble dialogues rather than conventional speeches. This factor may exacerbate the challenge posed by the presence of overlapping speech segments in the audio data of HParl. GREC without the HParl subcorpus results in a better DER with 23.28% than the baseline one. The best diarization score comes from Logotypografia combined with CSS10 achieving an error rate of 4.43% which is a rela-

tive improvement of 83%. Next, following close behind is again Logotypografia combined with HParl, now reporting 4.72% DER with a relative reduction of approximately 82%. The score obtained by disregarding Logotypografia and training with the remaining subcorpora of GREC is as a matter of fact the worst across the experiments with 50.74% DER. Additionally, CV9 absence from the training data produces an error rate of 5.66%, a relative improvement of 78%. Furthermore, disregarding CSS10 from GREC has barely any impact on the evaluation set. As stated in section 5.2.2, Logotypografia contains speech data from broadcast news and most of the research efforts on speaker diarization have focused on this specific domain [8, 154, 155]. This type of data would work well within the specific ASR task as they typically feature multiple speakers, they are recorded in a controlled environment with consistent recording conditions, ensuring that the data quality is high and they include a mix of speakers with different ages, genders, and accents, providing a diverse training dataset for the system to learn from. Unfortunately, the audio data in Logotypografia consist of single speaker instances, which are not suitable for use as training input for a speaker diarization system.

As previously observed, utilizing single speaker datasets as the training input for a speaker diarization system may result in a low Diarization Error Rate. However, this does not provide a comprehensive evaluation of the system's performance. Speaker diarization systems are designed to separate and identify multiple speakers within an audio signal. The training of such a system with a single speaker dataset will not furnish the system with the capability to differentiate between multiple speakers. In this scenario, a low DER would only indicate the accuracy in recognizing the singular speaker in the training data, but it cannot be relied upon to determine the system's performance in situations that involve multiple speakers, which is the primary function of a speaker diarization system. It is essential to train the system with multi-speaker datasets to obtain a more meaningful evaluation of its performance.

To date, HParl is the sole subcorpus considered suitable for training a diarization system, yet in a subsequent iteration of GREC, it may be feasible to incorporate other datasets containing multiple speakers. The results obtained from the HParl subcorpus, although encouraging, also indicate the necessity for further research in order to fully harness the potential of a robust speaker diarization system. The system must be capable of addressing the diversity of interaction patterns and recording conditions encountered in real-world scenarios. Refining the evaluation methodology can facilitate future assessments aimed at identifying speakers in full.

7. CONCLUSIONS & FUTURE WORK

The present work demonstrates the creation of a large, multi-domain corpus for Automatic Speech Recognition for the Greek language. GREC is a collection of three subcorpora over different domains, plus a novel speech corpus derived from parliamentary speech data. For the latter, we described in detail our data collection, preprocessing and alignment setup, which are based on the open-source speech recognition toolkit Kaldi. Moreover, we performed extensive analysis on the recognition performance of GMM-HMM and DNN-HMM models over the different acoustic domains. On top of that, we incorporated speaker diarization capabilities to Kaldi-gRPC-Server, which is a tool based on PyKaldi and gRPC for streamlined deployment of Kaldi based speech recognition.

The preliminary results of the experiments conducted have showed how the proposed data pipeline is capable of extracting data of substantial quality which could be useful both for evaluating the performance of out-of-domain systems in this task, together with system adaptation to the specific domain of parliamentary conversations. Nevertheless, the results listed in this work are purely extracted for the purpose of providing a baseline system for future improvement. They might not reflect the state-of-the-art performance of each acoustic model since hyperparameter tuning is a significant factor to search for proper optimizations.

The long-term goal of this work is to extend the GREC corpus systematically by appending the newest proceedings speech and transcription data to the existing by using the current data preparation pipeline. For the next versions of the dataset, there are plethora of unutilized data to be integrated from the parliamentary archive, while new sessions are uploaded every day. Conveniently, data augmentation won't be needed since the expanding volume of data is sufficient for training. As a direct consequence, any model trained to previous dataset editions will be effectively improved.

In conjunction with the aforementioned future objectives, the collected metadata will be enriched in order to include statistics about the speech segments, properties of speakers' characteristics and political identities. With regard to the speaker distribution along the speech data, having already handled the proportional number of male and female speakers between the training and the test sets, the gender balance will be ensured in each and every subset in future versions of the dataset. On the same topic, a key point in achieving diversity and in avoiding overfitting outcomes is to regulate the presence of each speaker in the speech data, considering the fact that a large number of them appear several times in each parliamentary speech session.

With reference to the pipeline, optimizing the preparation stages and the recognition process by varying the model's parameters could increase the utility and performance of the entire system. Language model rescoring could improve both in-domain as well as out-of-domain results. The performance of the pipeline will also be significantly improved with the incorporation of speaker adaptation methods. It should be underlined that this work sets the baseline for future research and ASR applications.

Undoubtedly, the availability of this multi-source, multi-target corpus and the related techniques for data retrieval and processing offer a valuable resource for continued research in the advancement of language technology for Modern Greek. In the near future, the data and code will be made accessible to the public. Data and code will be made publicly available in the near future.

ABBREVIATIONS - ACRONYMS

ILSP	Institute for Language and Speech Processing
CC0	Creative Commons
CV	Common Voice
CV9	Common Voice version 9
EPPS	European Parliament Plenary Session
SUR	Speech Understanding Research
IBM	International Business Machines Corporation
DARPA	Defense Advanced Research Projects Agency
ARPA	Advanced Research Projects Agency
MDCC	Multi-domain Cantonese Corpus
SRI	Stanford Research Institute
CMU	Carnegie Mellon University
CMUdict	Carnegie Mellon University Pronunciation Dictionary
EPPS	European Parliament Plenary Session
ASR	Automatic Speech Recognition
LVCSR	Large Vocabulary Continuous Speech Recognition
LV-ASR	Large Vocabulary Automatic Speech Recognition
TTS	Text-To-Speech
UDA	Unsupervised Domain Adaptation
DCTTS	Deep Convolutional TTS
NIST	National Institute of Standards and Technology
MIT	Massachusetts Institute of Technology
TIMIT	TIMITTexas Instruments, Inc. and MIT
MLS	Multilingual LibriSpeech
SID	Speaker Independent
SD	Speaker Dependent
RT	Rich Transcription
SAD	Speech Activity Detection
VAD	Voice Activity Detection

STT	Speech-To-Text
AM	Acoustic Model
LM	Language Model
GMM	Gaussian Mixture Models
SGMM	Subspace Gaussian Mixture Models
HMM	Hidden Markov Models
DNN	Deep Neural Network
RNN	Recurrent Neural Networks
TDNN	Time delay neural network
RNNLM	Recurrent Neural Network Language Modeling
LSTM	Long Short Term Memory
KL-HMM	Kullback-Leibler based Hidden Markov Model
GRU	Gated Recurrent Unit
DBN	Deep Belief Network
RBM	Restricted Boltzmann Machine
CAS	Childhood Apraxia of Speech
HTK	Hidden Markov model Toolkit
SRILM	The SRI Language Modeling Toolkit
DTW	Dynamic Time Warping
gRPC	gRPC Remote Procedure Calls
BLAS	Basic Linear Algebra Subroutines
LAPACK	Linear Algebra PACKage
WER	Word Error Rate
DER	Diarization Error Rate
MFCC	Mel-frequency Cepstral Coefficients
DCT	Discrete Cosine Transformation
MLE	Maximum Likelihood Estimation
DFT	Discrete Fourier Transform
PDF	Probability Density Function
PDF-ID	Probability Density Function Identifier
ML	Maximum Likelihood
MLLR	Maximum Likelihood Linear Regression

fMLLR	Feature space Maximum Likelihood Linear Regression
EM	Expectation-Maximization
LBG	Linde-Buzo-Gray algorithm
MAP	Maximum A Posteriori
UBM	Universal Background Model
LPC	Linear Prediction Coding
LPC	Linear Prediction Coefficients
LPCC	Linear Prediction Cepstral Coefficients
DWT	Discrete Wavelet Transform
FFT	Fast Fourier Transform
LSF	Line Spectral Frequencies
PLDA	Probabilistic Linear Discriminant Analysis
PLP	Perceptual Linear Prediction
FST	Finite State Transducer
WFST	Weighted Finite State Transducer
WFSA	Weighted Finite State Acceptor
HCLG	Decoding graph FST
CD	Context-Dependent
VTLN	Vocal Tract Length Normalization
CMVN	Cepstral mean and variance normalization
LDA	Linear Discriminant Analysis
AHC	Agglomerative Hierarchical Clustering
HLDA	Heteroscedastic Linear Discriminant Analysis
STC	Semi-Tied Covariance
MLLT	Maximum Likelihood Linear Transform
SAT	Speaker Adaptive Training
FST	Finite State Transducers
WFST	Weighted Finite State Transducers
G2P	Grapheme-To-Phoneme
API	Application Programming Interface

REFERENCES

[1] An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[2] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[3] Sabur Ajibola Alim and N Khair Alang Rashid. *Some commonly used speech feature extraction algorithms*. IntechOpen London, UK:, 2018.

[4] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. Openfst: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer, 2007.

[5] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus, 2019.

[6] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. The third 'chime'speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 504–511. IEEE, 2015.

[7] Jon Barker, Shinji Watanabe, Emmanuel Vincent, and Jan Trmal. The fifth'chime'speech separation and recognition challenge: dataset, task and baselines. *arXiv preprint arXiv:1803.10609*, 2018.

[8] Claude Barras, Xuan Zhu, Sylvain Meignier, and J-L Gauvain. Multistage speaker diarization of broadcast news. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1505–1512, 2006.

[9] Homayoon Beigi. *Signal Processing of Speech and Feature Extraction*, pages 143–204. Springer US, Boston, MA, 2011.

[10] Peter Bell, Joachim Fainberg, Ondrej Klejch, Jinyu Li, Steve Renals, and Pawel Swietojanski. Adaptation algorithms for neural network-based speech recognition: An overview. *IEEE Open Journal of Signal Processing*, 2:33–66, 2020.

[11] Jacob Benesty, M Mohan Sondhi, Yiteng Huang, et al. *Springer handbook of speech processing*, volume 1. Springer, 2008.

[12] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Automatic speech recognition for under-resourced languages: A survey. *Speech communication*, 56:85–100, 2014.

[13] Sahar E Bou-Ghazale and John HL Hansen. Duration and spectral based stress token generation for hmm speech recognition under stress. In *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages I–413. IEEE, 1994.

[14] Sahar E Bou-Ghazale and John HL Hansen. Generating stressed speech from neutral speech using a modified celp vocoder. *Speech Communication*, 20(1-2):93–110, 1996.

[15] Dogan Can, Victor R. Martinez, Pavlos Papadopoulos, and Shrikanth S. Narayanan. Pykaldi: A python wrapper for kaldi. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[16] Paolo Castiglioni. Levinson-durbin algorithm. *Encyclopedia of Biostatistics*, 4, 2005.

[17] Aimilios Chalamandaris, Athanassios Protopapas, Pirros Tsiakoulis, and Spyros Raptis. All greek to me! an automatic greeklish to greek transliteration system. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, 2006.

[18] Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, Mingjie Jin, Sanjeev Khudanpur, Shinji Watanabe, Shuaijiang Zhao, Wei Zou, Xiangang Li, Xuchen Yao, Yongqing Wang, Yujun Wang, Zhao You, and Zhiyong Yan. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio, 2021.

[19] Heidi Christensen, Jon Barker, Ning Ma, and Phil D Green. The chime corpus: a resource and a challenge for computational hearing in multisource environments. In *Eleventh Annual Conference of the International Speech Communication Association*. Citeseer, 2010.

[20] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[21] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Librimix: An open-source dataset for generalizable speech separation. *arXiv preprint arXiv:2005.11262*, 2020.

[22] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

[23] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

[24] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[25] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at microsoft. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8604–8608. IEEE, 2013.

[26] Li Deng and Doug O'Shaughnessy. *SPEECH PROCESSING — A Dynamic and Optimization-Oriented Approach*. Marcel Dekker Inc., June 2003.

[27] Pavel Denisov, Ngoc Thang Vu, and Marc Ferras Font. Unsupervised domain adaptation by adversarial learning for robust speech recognition. In *Speech Communication; 13th ITG-Symposium*, pages 1–5. VDE, 2018.

[28] Mireia Diez, Lukás Burget, and Pavel Matejka. Speaker diarization based on bayesian hmm with eigenvoice priors. In *Odyssey*, pages 147–154, 2018.

[29] Vassilios Digalakis, Dimitrios Oikonomidis, D. Pratsolis, N. Tsourakis, C. Vosnidis, N. Chatzichrisafis, and V. Diakoloukas. Large vocabulary continuous speech recognition in greek: corpus and an automatic dictation system. In *Proc. 8th European Conference on Speech Communication and Technology (Eurospeech 2003)*, pages 1565–1568, 2003.

[30] Vassilios V Digalakis, Peter Monaco, and Hy Murveit. Genones: Generalized mixture tying in continuous hidden markov model-based speech recognizers. *IEEE Transactions on Speech and audio Processing*, 4(4):281–289, 1996.

[31] Dimitrios Dimitriadis, A Metallinou, Ioannis Konstantinou, G Goumas, Petros Maragos, and Nectarios Koziris. Gridnews: A distributed automatic greek broadcast transcription system. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1917–1920. IEEE, 2009.

[32] Sree HK Parthasarathi e al. fmllr based feature-space speaker adaptation of dnn acoustic models. In *Proc. Interspeech*, pages 3630–3634, 2015.

[33] Tomaž Erjavec, Maciej Ogrodniczuk, Petya Osenova, Nikola Ljubešić, Kiril Simov, Andrej Pančur, Michał Rudolf, Matyáš Kopp, Starkaður Barkarson, Steinþór Steingrímsson, et al. The parlamint corpora of parliamentary proceedings. *Language resources and evaluation*, pages 1–34, 2022.

[34] Jason Ernst and Manolis Kellis. ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods*, 9(3):215–216, February 2012.

[35] Anmol Gulati et al. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech*, pages 5036–5040, 2020.

[36] Ehsan Hosseini-Asl et al. Augmented cyclic adversarial learning for low resource domain adaptation. In *Proc. ICLR*, 2018.

[37] Salil Deena et al. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition and alignment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):572–582, 2018.

[38] Siva R. Gangireddy et al. Unsupervised adaptation of recurrent neural network language models. In *Proc. Interspeech*, pages 2333–2337, 2016.

[39] Taichi Asami et al. Domain adaptation of dnn acoustic models using knowledge distillation. In *Proc. ICASSP*, pages 5185–5189. IEEE, 2017.

[40] Tom Ko et al. A study on data augmentation of reverberant speech for robust speech recognition. In *ICASSP*, pages 5220–5224. IEEE, 2017.

[41] Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. Irstlm: an open source toolkit for handling large scale language models. In *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[42] Jonathan Fiscus, Jerome Ajot, Martial Michel, and John Garofolo. The rich transcription 2006 spring meeting recognition evaluation. pages 309–322, 01 2006.

[43] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[44] Sadaoki Furui. A training procedure for isolated word recognition systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(2):129–136, 1980.

[45] Mark Gales, Steve Young, et al. The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

[46] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[47] Daniel Galvez, Greg Diamos, Juan Ciro, Juan Felipe Cerón, Keith Achorn, Anjali Gopi, David Kanter, Maximilian Lam, Mark Mazumder, and Vijay Janapa Reddi. The people's speech: A large-scale diverse english speech recognition dataset for commercial usage. *arXiv preprint arXiv:2111.09344*, 2021.

[48] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

[49] Diana Geneva, Georgi Shopov, and Stoyan Mihov. *Building an ASR Corpus Based on Bulgarian Parliament Speeches*, pages 188–197. 09 2019.

[50] Shabnam Ghaffarzadegan, Hynek Bořil, and John HL Hansen. Deep neural network training for whispered speech recognition using small databases and generative model sampling. *International Journal of Speech Technology*, 20(4):1063–1075, 2017.

[51] Christian Gollan, Maximilian Bisani, Stephan Kanthak, Ralf Schï, and Hermann Ney. Cross domain automatic transcription on the tc-star epps corpus. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, 1, 01 1998.

[52] Kyle Gorman, Jonathan Howell, and Michael Wagner. Prosodylab-aligner: A tool for forced alignment of laboratory speech. *Canadian Acoustics*, 39:192, 2011.

[53] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772. PMLR, 2014.

[54] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[55] Kyu J Han, Samuel Kim, and Shrikanth S Narayanan. Strategies to improve the robustness of agglomerative hierarchical clustering under data source variation for speaker diarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1590–1601, 2008.

[56] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[57] John HL Hansen, Pongtep Angkititrakul, Jay Plucienkowski, Stephen Gallant, Umit Yapanel, Bryan Pellom, Wayne Ward, and Ron Cole. " cu-move": Analysis & corpus development for interactive in-vehicle speech systems. In *Seventh European Conference on Speech Communication and Technology*, 2001.

[58] Inga Rún Helgadóttir, Róbert Kjaran, Anna Björk Nikulásdóttir, and Jón Guðnason. Building an asr corpus using althingi's parliamentary speeches. In *INTERSPEECH*, 2017.

[59] Inga Rún Helgadóttir, Anna Björk Nikulásdóttir, Michal Borskỳ, Judy Y Fong, Róbert Kjaran, and Jón Guðnason. The althingi asr system. In *INTERSPEECH*, pages 3013–3017, 2019.

[60] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *The Journal of the Acoustical Society of America*, 87 4:1738–52, 1990.

[61] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[62] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[64] Harald Höge, Christoph Draxler, Henk van den Heuvel, Finn Tore Johansen, EP Sanders, and Herbert S Tropf. Speechdat multilingual speech databases for teleservices: across the finish line. 1999.

[65] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23. IEEE, 2017.

[66] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[67] David Imseng, Hervé Bourlard, and Philip N Garner. Using kl-divergence and multilingual information to improve asr for under-resourced languages. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4869–4872. IEEE, 2012.

[68] David Imseng, John Dines, Petr Motlicek, Philip N Garner, and Hervé Bourlard. Comparing different acoustic modeling techniques for multilingual boosting. In *Proceedings of Interspeech*, number CONF, 2012.

[69] David Imseng, Ramya Rasipuram, and Mathew Magimai Doss. Fast and flexible kullback-leibler divergence based acoustic modeling for non-native speech recognition. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 348–353. IEEE, 2011.

[70] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. 2012.

[71] Photina Jaeyun Jang and Alexander G Hauptmann. Improving acoustic models with captioned multimedia speech. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 767–771. IEEE, 1999.

[72] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

[73] Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

[74] B.-H. Juang. On the hidden markov model and dynamic time warping for speech recognition-a unified view. *AT&ampT Bell Laboratories Technical Journal*, 63(7):1213–1243, September 1984.

[75] Biing-Hwang Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of markov chains. *AT&T Technical Journal*, 64:1235–1249, 1985.

[76] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 09 2022.

[77] Athanasios Katsamanis, Matthew Black, Panayiotis G Georgiou, Louis Goldstein, and Shrikanth Narayanan. Sailalign: Robust long speech-text alignment. In *Proc. of workshop on new tools and methods for very-large scale phonetics research*, 2011.

[78] Brian Kingsbury, Tara N Sainath, and Hagen Soltau. Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization. In *Thirteenth annual conference of the international speech communication association*, 2012.

[79] Keisuke Kinoshita, Marc Delcroix, Takuya Yoshioka, Tomohiro Nakatani, Emanuel Habets, Reinhold Haeb-Umbach, Volker Leutnant, Armin Sehr, Walter Kellermann, Roland Maas, et al. The reverb challenge: A common evaluation framework for dereverberation and recognition of reverberant speech. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.

[80] Andreas Kirkedal, Marija Stepanović, and Barbara Plank. Ft speech: Danish parliament speech corpus. *arXiv preprint arXiv:2005.12368*, 2020.

[81] Thomas Kisler, Florian Schiel, and Han Sloetjes. Signal processing via web services: the use case webmaus. In *Digital Humanities Conference 2012*, 2012.

[82] Aldebaro Klautau. Arpabet and the timit alphabet. *an archived file. https://web. archive. org/web/20160603180727/http://www. laps. ufpa. br/aldebaro/papers/ak_arpabet01. pdf (Accessed Mar. 12, 2020)*, 2001.

[83] Matyáš Kopp, Vladislav Stankov, Jan Oldřich Krza, Pavel Straňák, and Ondřej Bojar. Parczech 3.0: A large czech speech corpus with rich metadata. In *International Conference on Text, Speech, and Dialogue*, pages 293–304. Springer, 2021.

[84] Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459, 2017.

[85] Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The cmu sphinx-4 speech recognition system. In *Ieee intl. conf. on acoustics, speech and signal processing (icassp 2003), hong kong*, volume 1, pages 2–5, 2003.

[86] Akinobu Lee and Tatsuya Kawahara. julius-speech/julius: Release 4.5, January 2019.

[87] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech & language*, 9(2):171–185, 1995.

[88] S.E. Levinson. Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech Language*, 1(1):29–45, 1986.

[89] Na Li and Matthew Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–2233, December 2003.

[90] Guan-Ting Lin, Shang-Wen Li, and Hung-yi Lee. Listen, adapt, better wer: Source-free single-utterance test-time adaptation for automatic speech recognition. *arXiv preprint arXiv:2203.14222*, 2022.

[91] L Liporace. Maximum likelihood estimation for multivariate observations of markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, 1982.

[92] Nikola Ljubešić, Danijel Koržinek, Peter Rupnik, and Ivo-Pavao Jazbec. Parlaspeech-hr-a freely available asr dataset for croatian bootstrapped from the parlamint corpus. In *Proceedings of the Workshop ParlaCLARIN III within the 13th Language Resources and Evaluation Conference*, pages 111–116, 2022.

[93] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[94] Bruce T Lowerre. *The harpy speech recognition system.* Carnegie Mellon University, 1976.

[95] Jonas Lööf, Christian Gollan, Stefan Hahn, Georg Heigold, Bjorn Hoffmeister, Christian Plahl, David Rybach, Ralf Schlüter, and Hermann Ney. The rwth 2007 tc-star evaluation system for european english and spanish. pages 2145–2148, 08 2007.

[96] Mathew Magimai-Doss, Ramya Rasipuram, Guillermo Aradilla, and Hervé Bourlard. Grapheme-based automatic speech recognition using kl-hmm. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[97] Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur. Jhu kaldi system for arabic mgb-3 asr challenge using diarization, audio-transcript alignment and transfer learning. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 346–352, 2017.

[98] Akhil Mathur, Fahim Kawsar, Nadia Berthouze, and Nicholas D. Lane. Libri-adapt: a new speech dataset for unsupervised domain adaptation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2020.

[99] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pages 498–502, 2017.

[100] Andrew McCallum. Hidden markov models baum welch algorithm. *Introduction to Natural Language Processing CS585; University of Massachusetts Amherst: Massachusetts, USA*, 2004.

[101] Carsten Meyer and Hauke Schramm. Boosting hmm acoustic models in large vocabulary speech recognition. *Speech Communication*, 48(5):532–548, 2006.

[102] Yajie Miao, Hao Zhang, and Florian Metze. Towards speaker adaptive training of deep neural network acoustic models. In *Proc. Interspeech*, pages 2189–2193, 2014.

[103] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*, 20(1):14–22, 2011.

[104] Abdel-rahman Mohamed and Geoffrey Hinton. Phone recognition using restricted boltzmann machines. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4354–4357. IEEE, 2010.

[105] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[106] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[107] Mehryar Mohri and Michael Riley. An efficient algorithm for the n-best-strings problem. In *Seventh International Conference on Spoken Language Processing*, 2002.

[108] Pedro J Moreno and Christopher Alberti. A factor automaton approach for the forced alignment of long speech recordings. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4869–4872. IEEE, 2009.

[109] Pedro J Moreno, Christopher F Joerg, Jean-Manuel Van Thong, and Oren Glickman. A recursive algorithm for the forced alignment of very long audio segments. In *ICSLP*, volume 98, pages 2711–2714, 1998.

[110] Iosif Mporas, Todor Ganchev, Theodoros Kostoulas, Katia Kermanidis, and Nikos Fakotakis. Automatic speech recognition system for home appliances control. In *2009 13th Panhellenic Conference on Informatics*, pages 114–117. IEEE, 2009.

[111] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[112] Hy Murveit, Michael Cohen, Patti Price, Gay Baldwin, Mitch Weintraub, and Jared Bernstein. Sri's decipher system. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*, 1989.

[113] Navnath S Nehe and Raghunath S Holambe. DWT and LPC based feature extraction methods for isolated word recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2012(1), January 2012.

[114] H. Niemann, E. Nöth, E. G. Schukat-Talamazzini, A. Kiessling, R. Kompe, T. Kuhn, K. Ott, and S. Rieck. Statistical modeling of segmental and suprasegmental information. In *Speech Recognition and Coding*, pages 192–209. Springer Berlin Heidelberg, 1995.

[115] Dimitrios Oikonomidis and Vassilios Digalakis. Stem-based maximum entropy language models for inflectional languages. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[116] Douglas O'Shaughnessy. Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, 41(10):2965–2979, 2008.

[117] Jia Pan, Cong Liu, Zhiguo Wang, Yu Hu, and Hui Jiang. Investigation of deep neural networks (dnn) for large vocabulary continuous speech recognition: Why dnn surpasses gmms in acoustic modeling. In *2012 8th International Symposium on Chinese Spoken Language Processing*, pages 301–305. IEEE, 2012.

[118] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[119] F Pantazoglou, N Papadakis, and G Kladis. Implementation of the generic greek model for cmu sphinx speech recognition toolkit. *Proceedings of eRA-12*, 2017.

[120] Kyubyong Park and Thomas Mulc. Css10: A collection of single speaker speech datasets for 10 languages. *arXiv preprint arXiv:1903.11269*, 2019.

[121] Douglas B Paul and Janet Baker. The design for the wall street journal-based csr corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[122] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth annual conference of the international speech communication association*, 2015.

[123] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, pages 3743–3747, 2018.

[124] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[125] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[126] Simon JD Prince and James H Elder. Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[127] Lawrence Rabiner. First hand: the hidden markov model. *IEEE Global History Network*, pages 4–15, 2013.

[128] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[129] Lawrence R Rabiner. Readings in speech recognition. chapter a tutorial on hidden markov models and selected applications in speech recognition, pages 267–296. *Google Scholar Google Scholar Digital Library Digital Library*, 1990.

[130] Ramya Rasipuram, Peter Bell, and Mathew Magimai Doss. Grapheme and multilingual posterior features for under-resourced speech recognition: a study on scottish gaelic. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7334–7338. IEEE, 2013.

[131] Ramya Rasipuram, Marzieh Razavi, and Mathew Magimai-Doss. Probabilistic lexical modeling and unsupervised training for zero-resourced asr. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 446–451. IEEE, 2013.

[132] Mirco Ravanelli, Luca Cristoforetti, Roberto Gretter, Marco Pellin, Alessandro Sosi, and Maurizio Omologo. The dirha-english corpus and related tasks for distant-speech recognition in domestic environments. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 275–282. IEEE, 2015.

[133] Chandan KA Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke. A scalable noisy speech dataset and online subjective test framework. *arXiv preprint arXiv:1909.08050*, 2019.

[134] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[135] Jürgen Riedler and Sergios Katsikas. Development of a modern greek broadcast-news corpus and speech recognition system. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 380–383, 2007.

[136] Ingrid Rosenfelder, Josef Fruehwald, Keelan Evanini, and Jiahong Yuan. Fave (forced alignment and vowel extraction) program suite. *URL http://fave. ling. upenn. edu*, 2011.

[137] Neville Ryant, Kenneth Church, Christopher Cieri, Alejandrina Cristia, Jun Du, Sriram Ganapathy, and Mark Liberman. The second dihard diarization challenge: Dataset, task, and baselines. *arXiv preprint arXiv:1906.07839*, 2019.

[138] David Rybach, Stefan Hahn, Patrick Lehnen, David Nolden, Martin Sundermeyer, Zoltán Tüske, Simon Wiesler, Ralf Schlüter, and Hermann Ney. Rasr - the rwth aachen university open source speech recognition toolkit. 12 2011.

[139] Md. Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication*, 54(4):543–565, May 2012.

[140] Hardik B Sailor, Ankur T Patil, and Hemant A Patil. Advances in low resource asr: A deep learning perspective. In *SLTU*, pages 15–19, 2018.

[141] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 24–29. IEEE, 2011.

[142] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

[143] Gregory Sell, David Snyder, Alan McCree, Daniel Garcia-Romero, Jesús Villalba, Matthew Maciejewski, Vimal Manohar, Najim Dehak, Daniel Povey, Shinji Watanabe, et al. Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge. In *Interspeech*, pages 2808–2812, 2018.

[144] Mostafa Shahin, Beena Ahmed, Jacqueline McKechnie, Kirrie Ballard, and Ricardo Gutierrez-Osuna. A comparison of gmm-hmm and dnn-hmm based pronunciation verification techniques for use in the assessment of childhood apraxia of speech. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[145] Stephen Shum, Najim Dehak, Ekapol Chuangsuwanich, Douglas Reynolds, and James Glass. Interspeech 2011 exploiting intra-conversation variability for speaker diarization. pages 945–948, 01 2011.

[146] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[147] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[148] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170. IEEE, 2016.

[149] Young Steve. A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine*, 13(5):45, 1996.

[150] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[151] Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. Srilm at sixteen: Update and outlook. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*. IEEE SPS, December 2011.

[152] Hang Su, Gang Li, Dong Yu, and Frank Seide. Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6664–6668. IEEE, 2013.

[153] Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788. IEEE, 2018.

[154] SE Tranter and Douglas A Reynolds. Speaker diarisation for broadcast news. In *Odyssey04-The Speaker and Language Recognition Workshop*, 2004.

[155] Sue E Tranter and Douglas A Reynolds. An overview of automatic speaker diarization systems. *IEEE Transactions on audio, speech, and language processing*, 14(5):1557–1565, 2006.

[156] Emmanouil G Tsardoulias, Andreas L Symeonidis, and Pericles A Mitkas. An automatic speech detection architecture for social robot oral interaction. In *Proceedings of the Audio Mostly 2015 on Interaction With Sound*, pages 1–8. 2015.

[157] Gokhan Tur and Andreas Stolcke. Unsupervised language model adaptation for meeting recognition. In *Proc. ICASSP*, volume 4, pages IV–173–IV–176. IEEE, 2007.

[158] Savitha S Upadhya, AN Cheeran, and Jagannath H Nirmal. Thomson multitaper mfcc and plp voice features for early detection of parkinson disease. *Biomedical Signal Processing and Control*, 46:293–301, 2018.

[159] Spyridoula Varlokosta, Spyridoula Stamouli, Athanassios Karasimos, Georgios Markopoulos, Maria Kakavoulia, Michaela Nerantzini, Aikaterini Pantoula, Valantis Fyndanis, Alexandra Economou, and Athanassios Protopapas. A greek corpus of aphasic discourse: collection, transcription, and annotation specifications. In *Proceedings of LREC 2016 Workshop. Resources and Processing of Linguistic and Extra-Linguistic Data from People with Various Forms of Cognitive/Psychiatric Impairments (RaPID-2016), Monday 23rd of May 2016*, number 128. Linköping University Electronic Press, 2016.

[160] Emmanuel Vincent, Jon Barker, Shinji Watanabe, Jonathan Le Roux, Francesco Nesta, and Marco Matassoni. The second 'chime'speech separation and recognition challenge: An overview of challenge systems and outcomes. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 162–167. IEEE, 2013.

[161] Anja Virkkunen, Aku Rouhe, Nhan Phan, and Mikko Kurimo. Finnish parliament asr corpus - analysis, benchmarks and statistics, 2022.

[162] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition, 2004.

[163] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Wölfel. Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems*, 12 2004.

[164] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[165] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

[166] Robert Weide. The carnegie mellon pronouncing dictionary [cmudict. 0.6]. *Pittsburgh, PA: Carnegie Mellon University*, 2005.

[167] Michael J Witbrock and Alexander G Hauptmann. Improving acoustic models by watching television. Technical report, Carnegie-Mellon University. Department of Computer Science, 1998.

[168] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. HMM-based audio keyword generation. In *Advances in Multimedia Information Processing - PCM 2004*, pages 566–574. Springer Berlin Heidelberg, 2004.

[169] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. The htk book. *Cambridge University Engineering Department*, 3, 2002.

[170] Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, Di Wu, and Zhendong Peng. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition, 2022.

[171] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

[172] Imed Zitouni. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104, 2007.