UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: Using Mobile Audio Transcription to Generate High-Quality Text Corpora for Low-Resource Languages

Author: Fardoza Tohab (THBFAR002)

Project Abbreviation: SABC2TXT

Supervisor(s): Hussein Suleman (hussein.suleman@uct.ac.za)

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Using Mobile Audio Transcription to Generate High-Quality Text Corpora for Low-Resource Languages

Final paper

Fardoza Tohab
Department of Computer Science
University of Cape Town
thbfar002@myuct.ac.za

ABSTRACT

The scarcity of electronic resources for many South African languages, such as isiXhosa, poses a barrier to text-based research and experimentation. To overcome this barrier, this paper investigates the feasibility of using mobile devices to automatically transcribe unstructured audio to generate a high-quality textual corpus using the PocketSphinx speech recognition toolkit across various dimensions such as background noise and conversational difficulties. The results reveal notable limitations, including high Word Error Rates (WER) and difficulties in capturing casual speech. Hardware differences and the rate of speech also influence performance. The results show that with the current data utilized, PocketSphinx is not viable for generating high-quality isiXhosa textual corpora on mobile devices, highlighting an urgent need for specialized model improvements.

CCS CONCEPTS

• Computing methodologies → Natural language processing;

• Computing methodologies → Speech recognition;

• Applied computing → Document management and text processing

KEYWORDS

Speech/language, automatic speech recognition, mobile devices, natural language processing, low-resourced languages, PocketSphinx

1 INTRODUCTION

In South Africa, nine out of the eleven official African languages lack electronic linguistic resources such as books or documents [1][2], which is detrimental to computational and statistical systems such as language models that rely on these documents. Access to high-quality linguistic resources, such as annotated corpora and core technologies, is one of the fundamental prerequisites for research, development, and assessment in Natural Language Processing (NLP) [2]. The development of speech technology is closely related to resource gathering since the statistical models that dominate Text-To-Speech (TTS) and Automatic Speech Recognition (ASR) systems depend on the availability of suitable resources for the determination of their parameters [3].

Automatic Speech Recognition (ASR) is the process of turning a speech signal into a series of words using an algorithm that is implemented as a computer program [39]. ASR is a technique that enables computing devices to translate human spoken words into computer-readable text via microphone or telephone input [40]. Speech recognition is distinct from voice recognition in that the former entails a machine's capacity to identify the words that are said (i.e., what is said), whereas the latter requires a machine's capacity to identify speaking style (i.e., who said something) [5].

The speech research community has demonstrated considerable interest in ASR for low-resource languages over the past decade. [6][7]. According to Berment [2004], a language is considered to be low-resourced if it lacks some (but not all) of the following characteristics: a distinctive writing system, linguistic knowledge, web resources, and electronic resources for speech and language technology [48]. It is crucial to emphasize that it differs from a minority language, which is a language spoken by a minority of a territory's people [8]. Only a small portion of the world's approximately 7,000 languages provide the materials needed for the application of human language technologies [9].

This project aims to explore and evaluate the feasibility of automatically transcribing structured and unstructured audio to create a high-quality textual corpus using standard speech tools and models. The project is divided into three components: transcription of structured audio, such as SABC broadcast news, transcription of unstructured audio using mobile devices in noisy or uncontrolled environments, and the creation of a gold standard corpus for transcription accuracy validation. This paper delves into the second component, examining the capability of mobile devices to capture and transcribe audio within casual or acoustically challenging environments. Given the rapid increase in mobile phone usage in Sub-Saharan Africa, which currently accounts for 60% of the population [10], even speakers of low-resource languages may collect and transcribe audio with their devices, highlighting the potential of mobile devices in linguistic resource development.

2 BACKGROUND

2.1 IsiXhosa as a low-resource language

More than 8 million people, or 16% of South Africa's population, speak isiXhosa (Xhosa) as their first language [15]. However, isiXhosa is classified as a low-resource language [2][16]. This means that digital information, linguistic models, and tools, as well as Information Retrieval (IR) services like search and translation, are scarce. The majority of South Africa's eleven official languages face this challenge. For this project, we will be using isiXhosa as the language of focus.

2.2 The structure of speech

Phonemes are the smallest structural component of speech and abstract speech signals [17]. Most languages have a particular collection of phonemes, with very similar sets. For example, IsiXhosa has 66 consonant phonemes, 18 of which are click consonants found in fewer than 2% of the world's languages [18]. Phonemes group together similar sounds, ignoring details such as speech pace or accent. Current approaches for converting speech to text frequently interpret the audio signal into phonemes and then build sentences using algorithms [17]. In contrast to the abstract nature of phonemes, phones are the physical basic sound units that include vowels or consonants with distinct characteristics[17][20]. For example, the phrase "hello there" might be phonetically broken down into HH, EH, L, OW, DH, EH, R. The context is frequently considered when attempting to identify which phoneme a spoken sound belongs to. Triphones are phones that are derived from context, taking into account the phones said before and after. Triphones create words, which create sentences [19].

2.3 Speech recognition models

Three models are employed in speech recognition systems for translations: the acoustic model, phonetic dictionary, and language model [19][26][24].

2.3.1 Acoustic model

The acoustic model is used to convert data from an audio signal to the most probable phones uttered. It provides statistical mappings for each phone (or phoneme) and defines how words sound. Some speech recognition algorithms may use abstract statistical representations in their acoustic models [20].

2.3.2 Phonetic dictionary

The link between words and their phones is mapped out in the phonetic dictionary. Due to varying pronunciations, certain terms in the dictionary may have many variations. The decoding time for speech recognition can be considerably impacted by the size of this model [20].

2.3.3 Language model

To distinguish between words with similar sounding pronunciations, such as "two" and "too," speech recognition systems utilize language models to estimate the likelihood of word sequences. Context is crucial because words can change their meaning depending on the words they are used with. N-gram models, where the context is given by the (N-1) preceding words, are frequently used to estimate a word's likelihood by taking into account a certain number of prior words [19][20][24].

2.4 The speech recognition process

In a speech recognition system, a recording of spoken words is used as the input, and the anticipated output is a string that accurately captures the spoken sentence. Pre-processing, feature extraction, decoding, and post-processing are the four main steps that may be used to summarize this translation process, as shown in Figure 1. [17][27].

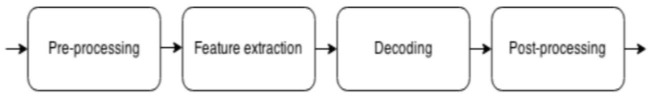


Figure 1: The speech recognition process

2.4.1 Pre-processing

The audio signal must have a predetermined number of channels and, depending on the system, a frequency rate of typically 8000 KHz or 16000 KHz to convert speech to text. During the pre-processing stage, some speech recognition systems have the ability to re-sample the audio signal to the appropriate format. This phase involves making an effort to remove any speech-free portions of the audio signal. This includes silences, such as at the start or end of an audio clip, as well as noisy sections [20].

2.4.2 Feature extraction

Before the actual decoding process can begin, certain filtering steps must be completed due to the enormous quantities of data that may be extracted from an audio file. Depending on the system, the data, which is now presumed to be speech, is broken up into overlapping frames that last around 25 milliseconds. In the next phases, parameters are taken from each of these frames to represent the audio signal. Feature vectors are a common term used to describe the resultant frames that were extracted from the data [20].

2.4.3 Decoding

During the decoding phase in the speech recognition process, the most representative sentence for the feature vectors is chosen using Bayes Rule, represented as [20]:

$$w^* = argmax_w\{p(Y|s)p(s)\} \tag{1}$$

where Y signifies feature vectors and S denotes the probable word sequence. For modeling stochastic signals, dynamic Bayesian networks called Hidden Markov Models (HMMs) are crucial. They connect audio signals to phones, each of which is identified by its HMM state. Using algorithms like the Viterbi or Forward, the most likely spoken sequence is determined [20][24]. Figure 2 shows how the speech waveform from the audio source was transformed into a series of fixed-size acoustic vectors via feature extraction. The decoder then makes use of the speech models to construct the word sequences that produce the feature vector.

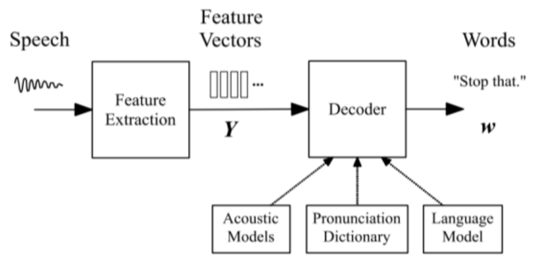


Figure 2: An HMM-based recognizer architecture

2.4.4 Post-processing

In the post-processing phase, the list of likely spoken sequences determined by the Viterbi search is examined once again utilizing other sources of data to enhance the correctness of the transcription. Higher level language models are typically employed to better consider grammar while choosing the most appropriate statement [28].

2.5 CMU Sphinx: PocketSphinx

The speech recognition toolkit that was used for this project is CMU Sphinx's PocketSphinx.

PocketSphinx is a user-dependent speech recognition system. It was developed at Carnegie Mellon University (CMU), along with other speech recognition systems [29]. It is the quickest speech recognizer CMU has ever produced and was built to operate in real-time on low-performance devices. To convert speech into text, the system employs a probabilistic method and Hidden Markov Models.

Because a translation using PocketSphinx is user-dependent, the outcomes can considerably differ from speaker to speaker. The system is capable of being trained to adapt to any language or dialect. The trigram model, which PocketSphinx employs, is an N-gram model with N = 3. When generating a word, N-gram models take into account the (N-1) preceding words. This suggests that word choice depends in part on context [30]. Numerous comprehensive acoustic models that have been tuned for various languages are available. For the purpose of this project, a new isiXhosa language model and an acoustic model were trained.

2.6 Sequitur G2P: Grapheme-to-phoneme conversion

Finding a word's pronunciation based on its written form is known as grapheme-to-phoneme conversion (G2P) [46]. A G2P model transforms a word from a collection of symbols or graphemes to a pronunciation from a collection of phones [47]. Sequitur G2P, a data-driven tool developed by Maximilian Bisani at RWTH Aachen University [46], is used to generate phonetic representations of isiXhosa words not present in the static dictionary. This tool is commonly used in speech recognition models.

3 RELATED WORK

Numerous studies have been conducted on how mobile devices can aid in the creation of electronic resources and documents, which can then be used to support studies in search engines, machine learning, and other language processing tasks.

The rise of mobile devices as multifunctional recording tools has revolutionized ASR data collection, especially for languages with limited resources. The decreasing cost and growing availability of mobile devices, even in developing regions, have made dynamic, high-quality speech data collection more feasible [13]. Hughes et al. [2010] pioneered this trend with an Android application designed to easily gather varied and transcribed speech corpora [14]. Although powerful, this approach had limitations such as inaccuracies due to speaker errors. Lane et al. [2010] further streamlined the speech data collection process, enabling speakers to record prompted speech directly on their phones, but faced challenges with the quality of recordings [42]. These collection methods largely focus on "read speech," but their data have proven invaluable for developing ASR systems [13].

Reitmaier et al. [2022] introduced the Voice Notes Android app, capable of processing audio from apps like WhatsApp. Despite its capabilities, the ASR application has its shortcomings, including difficulties with uncommon phrases and the loss of certain metadata during transfers [43]. Meanwhile, the open-source program Woefzela, introduced by de Vries et al. [44], specifically targeted low-resource languages in developing regions. Despite quality control features, the collected data can sometimes be suboptimal, impacting the ASR systems' accuracy. Badenhorst et al. [45] later suggested more intricate quality checks, targeting transcription mismatches.

Lakdawala et al. [11] presented an offline speech-to-text transcription system for healthcare organizations. It can be used by counsellors and non-governmental groups to capture talks during surveys, convert them to text, and then save the messages. This system includes an open-source application. The CMUSphinx toolkit is utilized for speech recognition. The system can recognize multiple languages. The language model, phonetic dictionary, and acoustic model are all utilized by the CMUSphinx toolkit. The user captures their voice using the mobile application, and the CMUSphinx toolkit analyses and transcribes it. The transcription file is saved as a text file in the device's memory; the user can use the application to upload and download data to and from the database server. Although the CMUSphinx speech recognition toolkit is dependable and accurate, mistakes happened nevertheless. Background noise, speaker accents, and rate of speech, among other things, also affected how accurately a transcript was made [11].

Liu and Zhou [12] introduced a Chinese small-vocabulary offline speech recognition system based on the PocketSphinx toolkit. The language model is built through the online tool LMTool, and the acoustic models are renewed by enhancing the Sphinx models that already exist. Then an offline speech recognition system that functions on an Android smartphone was created. The outcomes of the experiment demonstrated that the system used to recognize speech commands for mobile phones performs well in terms of recognition [12].

This paper contributes to the existing body of knowledge on mobile-based Automatic Speech Recognition (ASR) systems with an emphasis on isiXhosa, a South African language that is notably underrepresented in digital resources. While Hughes et al. and Lane et al. have previously conducted research to develop ASR technology, addressing issues like read speech and simplicity of data collection, the challenges of creating a high-quality textual corpus from unstructured audio in a low-resource language have not been investigated yet. Additionally, new variables such as hardware differences, gender of the speaker, and rate of speech are introduced in this paper.

4 PROBLEM DEFINITION AND RESEARCH QUESTION

4.1 Problem definition

The scarcity of electronic linguistic resources for several South African languages hinders computational and statistical systems that rely on them. The lack of documents in these languages is an important issue, making it difficult to build high-quality text corpora required for text-based research. Collecting and producing these materials is often time-consuming, costly, and not feasible for low-resource languages [2]. As a result, it is critical to examine the usage of standard speech tools such as PocketSphinx to automatically transcribe unstructured audio, offering a critical data source for natural language processing.

4.2 Research question

How accurate is speech-to-text transcription of unstructured audio using the PocketSphinx speech recognition toolkit on mobile devices?

Accuracy in this context, can be defined using the metrics of Word Error Rate (WER) and Levenshtein distance at both word and character level. The research question investigates if the PocketSphinx speech recognition toolkit, when used on mobile devices, can accurately convert unstructured audio into text. It specifically intends to evaluate the toolkit's ability to transform spontaneous and unscripted audio recordings into written text when used on mobile devices, which may include casual conversations or audio recorded in unpredictable environments. The emphasis on mobile devices is important because, given their varying hardware capabilities, potential for ambient noise, and other related factors, they offer unique challenges and benefits from desktop systems.

5 EXPERIMENTAL METHODOLOGY

5.1 Evaluation Metrics

5.1.1 Word Error Rate

The word error rate (WER) is a metric for assessing how accurate a speech recognition system is [20]. The number of errors that occurred during the translation of an audio stream may be determined by calculating WER. According to Anusuya and Katti [32], WER is determined by adding up all of the errors in the hypothesis and dividing them by the total number of words in the correct sentence. An error occurs when the sentence's hypothesis differs from the actual sentence due to an improper word substitution, word deletion, or word insertion. The word error rate is calculated using Equation 2. The key success factor for a speech recognition system is a low WER.

$$WER = \frac{substitutions + deletions + insertions}{total\ number\ of\ words} \qquad (2)$$

5.1.2 Levenshtein distance

The Levenshtein distance (LD) is a measurement of how many modifications must be made to a word sequence to make it equivalent to another. For example, if the two phrases S1 ="how are you doing" and S2 ="what are you doing", the word "how" in S1 can be replaced with the word "what" to consist of the same words as S2. As a result, the Levenshtein distance between the two statements is 1 [37]. For this project, the Levenshtein distance at both word-level and character-level was used.

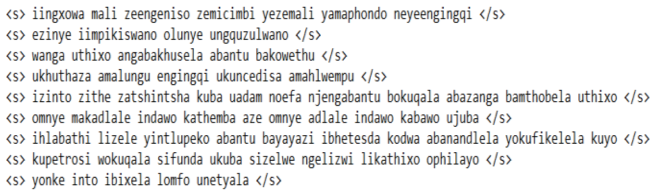
$$LD = substitutions + deletions + insertions \qquad (3)$$

5.2 Training database preparation

Since there are no previously trained language models or acoustic models in isiXhosa, a new language model and an acoustic model need to be trained. Training a new acoustic model requires using the command line prompts provided by CMU SphinxTrain [36]. SphinxTrain is a set of tools designed to create speech recognition systems for any language with adequate acoustic data, compatible with the CMUSphinx speech recognizer.

5.2.1 Building a language model

To build a language model for isiXhosa, the language modeling toolkit for CMUSphinx (CMUCLMTK) was used. The CMU-Cambridge Language Modeling Toolkit (CMUCLMTK) is a collection of UNIX software tools created to make language modeling work in the academic community easier [33]. To develop the language model, a reference text was constructed using the NCHLT isiXhosa Speech Corpus, a 56-hour speech collection comprised of orthographically transcribed utterances that capture a wide range of frequencies. Each entry in this corpus is presented in normalized text, enclosed by <s> and </s> tags. For a richer reference text, the initial text was combined with both the Lwazi isiXhosa TTS corpus and Lwazi II isiXhosa TTS Corpus, which offer orthographic and phonemically aligned transcriptions. All these resources were obtained from the South African Centre for Digital Language Resources (SADiLaR) [1] and feature audio recordings of isiXhosa speech paired with their transcriptions. The combination of these corpora resulted in a comprehensive reference text, as exemplified by a snippet in Figure 3. After this, a vocabulary file encompassing every unique word from the reference text was formulated. Lastly, with this data, a language model was generated in the ARPA format [34].

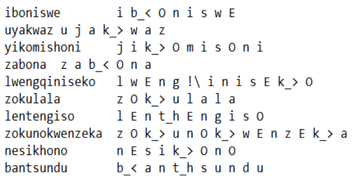


```
<s> iingxowa mali zeengeniso zemicimbi yezemali yamaphondo neyeengingqi </s>
<s> ezinye iimpikiswano olunye ungquzulwano </s>
<s> wanga uthixo angabakhusela abantu bakowethu </s>
<s> ukhuthaza amalungu engingqi ukuncedisa amahlwempu </s>
<s> izinto zithe zatshintsha kuba uadam noefa njengabantu bokuqala abazanga bamthobela uthixo </s>
<s> omnye makadlale indawo kathemba aze omnye adlale indawo kabawo ujuba </s>
<s> ihlabathi lizele yintlupeko abantu bayayazi ibhetesda kodwa abanandlela yokufikelela kuyo </s>
<s> kupetrosi wokuqala sifunda ukuba sizelwe ngelizwi likathixo ophilayo </s>
<s> yonke into ibixela lomfo unetyala </s>
```

Figure 3: A snippet of the isiXhosa reference text used to generate the language model.

5.2.2 Phonetic dictionary and G2P model training

The isiXhosa phonetic dictionary file was obtained from the NCHLT-inlang pronunciation dictionaries from SADiLaR, which are broad phonemic transcriptions for 15,000 common words in 11 different languages [35]. The phonetic dictionary contains the phonetic representation of all the words contained in the transcription files as exemplified by a snippet in Figure 4, allowing the decoder to understand how to pronounce each word.



```
iboniswe        i b_< O n i s w E
uyakwaz u j a k_> w a z
yikomishoni     j i k_> O m i s O n i
zabona  z a b_< O n a
lwengqiniseko   l w E n g !\ i n i s E k_> O
zokulala        z O k_> u l a l a
lentengiso      l E n t_h E n g i s O
zokunokwenzeka  z O k_> u n O k_> w E n z E k_> a
nesikhono       n E s i k_> O n O
bantsundu       b_< a n t_h s u n d u
```

Figure 4: A snippet of the phonetic dictionary that consists of the word and its corresponding phonetic representation.

The NCHLT-inlang isiXhosa dictionary was limited and some words that were present in the transcription file were not present in the dictionary. SphinxTrain expects all words in the transcription file to be present in the phonetic dictionary. To solve this issue, a grapheme-to-phoneme model was trained using Sequitur G2P to generate the phonetic representation of new words based on words present in the NCHLT-inlang dictionary. The Sequitur G2P model was trained in three iterations and the final model was used to generate the phonetic representation of any new words that were not present in the current dictionary.

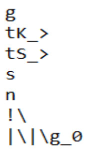
5.2.3 Training database structure

In the process of training an acoustic model, a training database is required. The database provides the data needed to create an acoustic model that extracts statistics from speech. The database was divided into two sections: a training section and a testing section. The testing section was about 1/10th of the total data size and did not exceed more than 4 hours of audio. The database prompts with post-processing contained the following database structure:

```
├─ etc
│  ├─ your_db.dic                    (Phonetic dictionary)
│  ├─ your_db.phone                  (Phoneset file)
│  ├─ your_db.lm.DMP                 (Language model)
│  ├─ your_db.filler                 (List of fillers)
│  ├─ your_db_train.fileids          (List of files for training)
│  ├─ your_db_train.transcription    (Transcription for training)
│  ├─ your_db_test.fileids           (List of files for testing)
│  └─ your_db_test.transcription     (Transcription for testing)
└─ wav
   ├─ speaker_1
   │  └─ file_1.wav                (Recording of speech utterance)
   └─ speaker_2
      └─ file_2.wav
```

Figure 5: The file structure for the database used for the acoustic model training

In the structure of the speech recognition training database in Figure 5, several file types play distinctive roles. The **.fileids* files are text documents that enumerate the names of individual recordings, or utterance IDs, by providing file-system paths relative to the wav directory. Complementing this, the **.transcription* files provide transcriptions corresponding to each audio recording. The actual audio data is stored in the **.wav* folder. It is crucial to ensure consistency between these recordings and their transcripts; any discrepancies can drastically reduce recognition accuracy. The *your_db.dict* file serves as the phonetic dictionary, presenting each word followed by its phonetic representation on a new line. The *your_db.phone* file houses the list of phones, detailing a list of phone sets that correspond to the phonetics in the dictionary as shown in Figure 6. For the language model, it is stored as a DMP in the *your_db.lm.DMP* file. Lastly, the *your_db.filler* file functions as the filler dictionary, accounting for non-linguistic sounds like breaths, "hmm", or laughter, which the primary language model does not cover.



```
g
tK_>
ts_>
s
n
!\
|\|\g_0
```

Figure 6: A snippet of some of the phone sets that can be found in the phonetic dictionary.

5.3 Acoustic model training

The training of the acoustic model was done in two iterations. The first iteration involved using data from the NCHLT isiXhosa Speech Corpus. The trained acoustic model in the first iteration had

[1] https://sadilar.org/index.php/en/resources

a limitation where it did not recognize words that were not included in the NCHLT isiXhosa Speech Corpus or the NCHLT-inlang isiXhosa dictionary. To overcome this limitation, the second iteration of the acoustic model training involved extending the dictionary and transcription file to include more words by combining the NCHLT isiXhosa Speech Corpus with the data from the Lwazi isiXhosa TTS corpus and Lwazi II isiXhosa TTS Corpus. The Word Error Rate (WER) and Sentence Error Rate (SER) of the model were computed by the decoder using the test portion of the training database and the reference transcripts throughout the training phase. The training results are shown in Table 1.

During the training of the acoustic model, an error emerged from *backward.c* indicating: "Failed to align audio to transcript: final state of the search is not reached." This was due to a mismatch between the audio in the database and its corresponding transcription. To address this, prompts that did not align properly were filtered out as they can compromise the acoustic model's quality. The solution was to activate the forced alignment stage during the training process.

The word error rate for the first iteration of the acoustic model training is 30.1% which indicates that, on average, 2442 out of 8107 words transcribed by the model were incorrect when compared to the reference transcription. An SER of 33.2% indicates that 920 out of 2770 sentences transcribed by the model were incorrect in some way when compared to the reference transcription. From Table 1, it can be seen that extending the dictionary and transcription file in the second iteration did not improve the WER but slightly decreased it by 0.1%, however, the SER remained the same. This was because the training data was extended so that the system could recognize more words and the test data remained the same.

Table 1: Results for the first and second iterations of the acoustic model training

Iteration	**Error rate evaluation**	**Error**	**Transcription errors**
Iteration 1	Sentence Error Rate (SER)	33.2%	920/2770
	Word Error Rate (WER)	30.1%	2442/8107
Iteration 2	Sentence Error Rate (SER)	33.2%	920/2770
	Word Error Rate (WER)	30.2%	2442/8107

5.4 Android app integration and implementation

To enable speech recognition on a mobile device, the acoustic model previously trained, the phonetic dictionary and the language model need to be integrated into an Android app. The CMUSphinx official website contains a demo Android app that uses PocketSphinx to do speech recognition in English [38].

5.4.1 Integration of speech models

The default demo app contains a directory called “models”, where it stores all the models required for speech recognition. By default, the app uses grammar files to create grammar-based searches and recognize digits. It also uses language models in the DMP format to recognize phones and weather forecasts in English. To recognize English words, the app makes use of the Carnegie Mellon University Pronouncing Dictionary, which is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations [50].

To modify the app to recognize isiXhosa words instead, the trained isiXhosa models and phonetic dictionary were added to the “models” directory. These models were then referenced directly in the main activity program called *PocketSphinxActivity.java*.

5.4.2 Customizations and Configurations

The default app is designed to present three different recognition demos. When the app detects the key phrase "oh mighty computer," it prompts users to select one of three demo options: "digits," "weather," or "phones." The "digits" demo detects digits from 0 to 9, the "weather" demo recognizes weather forecasts and the "phones" demo demonstrates phonetic recognition capabilities. The main functionality of the app is found in the *PocketSphinxActivity.java* file.

Several important changes were made while switching from the default PocketSphinx application to the new isiXhosa app. Firstly, the named searches for "digits," "phones," and "menu" were removed, reducing the application's functionality to primarily focus on keyword activation and recognition of isiXhosa words. The activation keyword was changed from "oh mighty computer" to "ubuntu," reflecting a term more culturally appropriate to the isiXhosa language and that is present in the isiXhosa phonetic dictionary. The acoustic model and dictionary paths in the *setupRecognizer* function were also adjusted to refer to the isiXhosa acoustic model and phonetic dictionary. A transcription logging system was also added, ensuring that recognized speech is reported to Android's Logcat for debugging and analysis. Details of the inner workings of the app and design can be found at [38].

5.4.3 App functionality

Upon launching the app, there's a brief initialization period for the recognizer. Following this, the app uses the device's microphone to await audio input. The app listens for the “ubuntu” key phrase that triggers the recognition process. Once this key phrase is detected, the app shifts into a continuous recognition mode until the end of the audio input. Subsequently, it displays the transcription of the audio on the screen as seen in Figure 7.

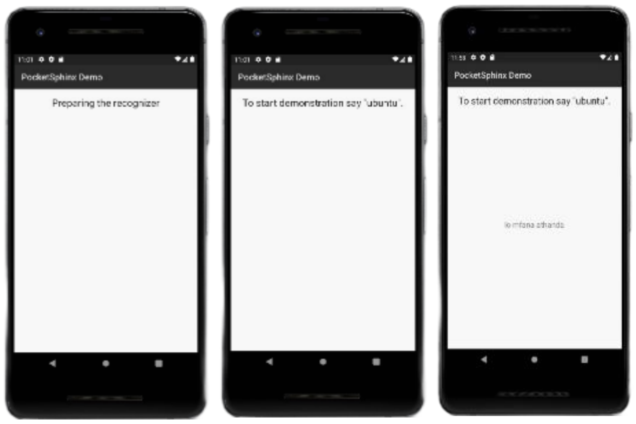


Figure 7: A demonstration of how the Android app works

5.5 Experimental Design

The experimental design is set up to rigorously assess the performance of the isiXhosa speech recognition system across diverse scenarios. The evaluation process consists of performing transcriptions using a variety of audio types such as audio with ambient background noise, spontaneous casual conversations, and audio clips of varying speech rates. Other factors such as the hardware of the device and the gender of the speaker are also taken into consideration.

5.5.1 Baseline audio

The baseline audio recordings that were used for the evaluation process are unstructured and structured audio recordings obtained from the gold standard corpus (the third component of the project), which consists of audio samples and their corresponding transcription. For the structured audio, SABC broadcast news that has been transcribed as part of the gold standard corpus was used. The unstructured audio used consisted of audio samples that had two or more people engaging in casual conversations in a noisy environment. The baseline audio also had a mixture of male and female speakers as well as speakers who spoke at different speeds. For this project, approximately 53 minutes of both structured and unstructured audio samples were used.

5.5.2 Data preprocessing

To experiment with diverse scenarios, the baseline audio needs to be segmented into smaller chunks and manipulated to either remove or add background noise and normalization if needed.

Segmentation: The duration of the audio samples used for the experiments was originally 10 minutes each. To better align with PocketSphinx's design for processing shorter utterances, these audio samples were segmented to produce shorter, more manageable chunks using the *AudioSegment* module from the *pydub* Python library. The structured audio samples (SABC broadcast news) were segmented by speaker and the unstructured audio samples were segmented in one, two, and three-minute segments.

Artificial background noise: Another key part of the experiment is to see how well the speech recognition system performs with audio that has background noise. For consistency, the structured audio samples segmented in the previous step were used as the baseline audio for this step. A Python script using the *pydub* Python library for audio file manipulation was used to introduce controlled levels of city-traffic background noise to the previously segmented audio samples. The script combined the clean audio with background noise at varying decibel levels, ranging from -20dB to 0dB in 4dB increments, and new audio samples with varying volumes of noise were produced.

Casual Conversations: This is audio data of two or more people engaging in casual conversations in isiXhosa with overlaps, interruptions, code-switching, and other casual speech characteristics. A Python script was used to preprocess the casual audio samples to remove background noise and add normalization using the *pydub*, *numpy,* and *scipy.signal* libraries. Each audio file was initially converted to a mono format in the processing pipeline, ensuring a coherent single-channel representation across all files. This conversion simplifies the audio by representing it with a single set of samples instead of numerous channels, which might add variability. The discrete signal amplitudes from the audio were retrieved as samples for further mathematical calculations after conversion. A Butterworth low-pass filter with a cutoff frequency of 4000 Hz is used to handle high-frequency noise and interference. This filter is recognized for its smooth frequency response with no ripples [49]. A normalization process is used to ensure that the audio maintains a constant amplitude level over the entire dataset. This process adjusts the amplitude of the audio samples so that they fit inside the dynamic range of the int16 data type, thus increasing the volume without causing distortion.

5.5.3 Different hardware

The transcriptions of the audio were done on the Android studio emulator by default. However, to test how the hardware of the recognition device such as the quality of the microphone affects the transcription accuracy, different mobile phones were employed. The mobile phones used are the Samsung Galaxy A72, the Samsung Galaxy Pocket, and the Xiaomi Redmi 6A. These mobile devices were tested for transcription accuracy using a standardized 25-second audio sample of a male speaker in a silent setting.

5.5.4 Transcription procedure

Since the PocketSphinx Android app uses a microphone to listen for input, the preprocessed audio was played on the Android app using a mobile phone with a good speaker (an iPhone). Once the app transcribed the audio, the transcription was recorded and saved to a text file. The transcription test was done on the Android studio emulator except for the hardware experiment section. For the hardware test, similar to the Android emulator, the audio was played from the iPhone onto the device's microphone.

Because the transcription varied each time the same audio was played, the transcription of each audio sample was done three times and the average WER and Levenshtein distance at both character and word level was taken to improve accuracy.

6 RESULTS AND DISCUSSIONS

The table of results for the various experiments can be found in the *Supplementary Information* section.

6.1 Background noise

The presence of background noise in audio recordings greatly affects the transcription accuracy. To test how well the isiXhosa speech recognition system works with different volumes of background noise, 36 audio samples with durations that ranged between 4 and 28s and volumes of background noise that ranged between 0 and -20 dB were used. The results are presented in the graphs in Figures 8 and 9.

A high background noise level of 0 dB resulted in an average WER of 1, which is considerably high given that the key success factor for speech recognition systems is a low WER. The standard deviation at this noise level was zero, meaning that the system consistently produced a WER of 1 for all samples. In comparison, at a quieter -20 dB, the system's mean WER improved to 0.94 with a standard deviation of 0.13, demonstrating improved transcription accuracy with lower levels of noise. The Levenshtein distance for 0 dB is 34 at the word level and 285 at the character level. Their standard deviations, 18.35 and 127.42, indicate the dispersion or variability in these distances from their averages. Figure 9 demonstrates a considerable drop in these distances when background noise levels decrease. The distances were 33 (word-level) and 243 (character-level) at -20 dB. Their standard deviations, 18.90 and 120.89, show the distribution of outcomes around these averages.

From the result, it is evident that the quality of the audio input is crucial for accurate transcription when using the PocketSphinx Android app for speech recognition, particularly for languages with complex phonetic structures like isiXhosa. As the volume of the background noise in the audio is decreased, the experimental findings in Figures 8 and 9 show a considerable reduction in both WER and Levenshtein distance. Phonemes, the smallest units of sound, are recognized individually by speech recognition algorithms to identify spoken information. These phonemes are obscured or distorted by ambient noise, leading to incorrect identification. More phonemes were correctly identified thanks to the reduction in the volume of the background noise, increasing the accuracy of transcription.

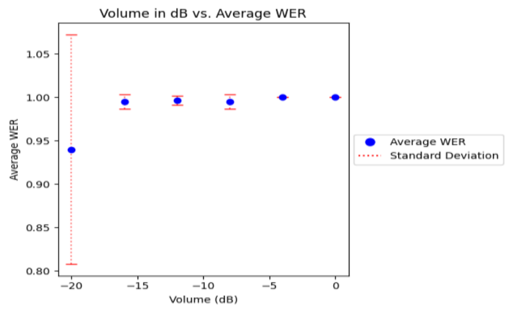


Figure 8: The average WERs for decreasing volumes of background noise.

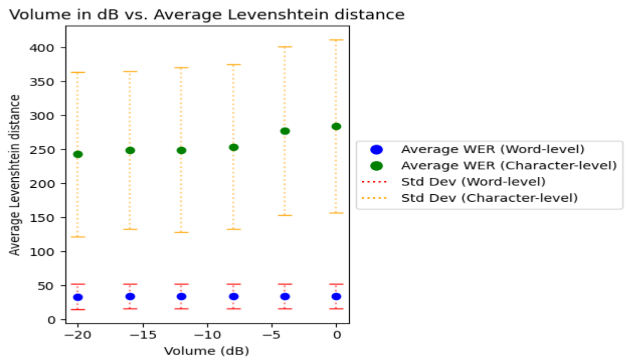


Figure 9: The average Levenshtein distance at word and character levels for decreasing volumes of background noise.

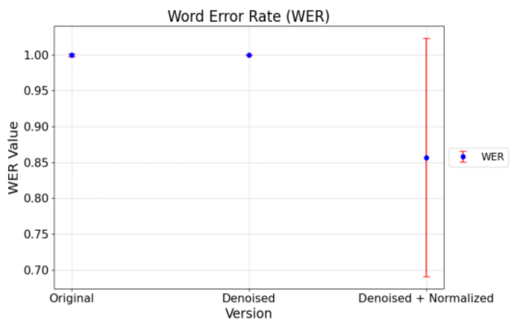
6.2 Casual conversations

The effectiveness of the speech recognition system when it is presented with audio that is conversational with overlaps, interruptions, background noise, and other casual speech characteristics was examined using 30 audio samples that ranged between 1 and 3 minutes. The results are shown in Figure 10.

The original audio samples produced an average WER of 1 with a standard deviation of 0.001, an average Levenshtein word distance of 141 with a standard deviation of 63, and an average Levenshtein character distance of 1025 with a standard deviation of 526. These are considerably high results and a WER of 1 indicates that the transcription was completely wrong. This is due to the presence of background noise and other factors. To improve accuracy, the original audio samples were preprocessed to remove background noise. The results in Figure 10 for the audio samples with background noise removed show no improvement with an average WER still at 1 with a standard deviation of 0. There is also a slight increase in the Levenshtein distances at both word and character levels. This might be because the noise removal technique might be too aggressive and is inadvertently removing or altering parts of the actual speech signal making some words or phonemes indistinguishable. The denoised audio was then normalized to see if this would improve the transcription accuracy. As can be seen in Figure 10, normalization improved the accuracy to an average WER of 0.86 with a standard deviation of 0.12. The Levenshtein word and character distances also improved to 123 and 912, respectively.

Conversations on the audio that was utilized for the transcription test frequently switched between English and isiXhosa. Although code-switching is a common linguistic occurrence, it presents special difficulties for speech recognition technology. When English was introduced, the acoustic model and language model, which were trained in isiXhosa, misinterpreted or altogether missed words because they were not intended to recognize multilingual input. Figure 11 shows how the system performs when it is presented with audio that contains English words. The system completely missed these words and did not transcribe them which is noted by the decrease in transcription length when compared to the original audio transcription. The isiXhosa words were also incorrectly transcribed due to other factors such as background noise or the words used not being present in the phonetic dictionary used to train the speech models.

Slang, acronyms, and colloquialisms were also used in the audio conversations. These often deviate from the standard language models as PocketSphinx's acoustic model for isiXhosa is mainly trained on more formal or standardized datasets and can be absent from the system's training data.



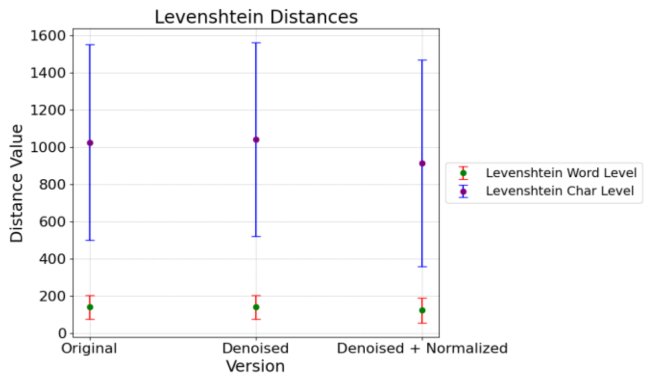
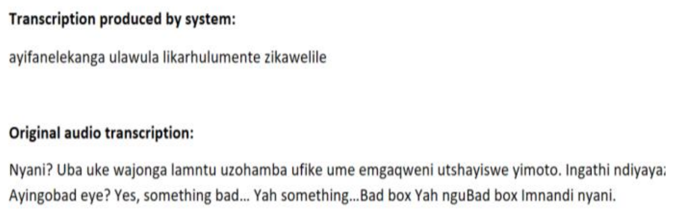


Figure 10: The results obtained after transcribing conversational audio with background noise, with no background noise, and normalization for WER (top) and Levenshtein distances (bottom).


Transcription produced by system:

ayifanelekanga ulawula likarhulumente zikawelile

Original audio transcription:

Nyani? Uba uke wajonga lamntu uzohamba ufike ume emgaqweni utshayiswe yimoto. Ingathi ndiyaya
Ayingobad eye? Yes, something bad... Yah something...Bad box Yah nguBad box Imnandi nyani.

Figure 11: An example of how the system performs when it is presented with audio that is conversational with code-switching.

6.3 Hardware implications on transcription accuracy

Three mobile phones were tested for transcription accuracy using a standardized 25-second audio sample of a male speaker in a silent setting. The original transcription of this audio contains 33 words and 290 characters excluding spaces. With a Word Error Rate (WER) of 0.36 and the smallest Levenshtein distances at both word and character levels (12 and 39) as shown in Table 2, the Xiaomi Redmi 6A performed the best. The Samsung Galaxy Pocket, on the other hand, substantially underperformed, having the highest Levenshtein distances at both word and character levels (25 and 136) and a WER of 0.77. With a WER of 0.49, the Samsung Galaxy A72 fell somewhere in the middle. These discrepancies could be caused by variations in microphones, hardware for processing audio, and other embedded technologies. These findings highlight the necessity of taking hardware characteristics like microphone quality and onboard audio processing technology into account when assessing or implementing speech recognition solutions on mobile devices.

Table 2: The results from performing transcription on different phones. The Xiaomi Redmi 6A performs the best.

Mobile phone type	**Average WER**	**Average Levenshtein distance (word level)**	**Average Levenstein distance (character level)**
Samsung Galaxy A72	0.49	16	91
Samsung Galaxy Pocket	0.77	25	136
Xiaomi Redmi 6A	0.36	12	39

6.4 Implications of gender on transcription accuracy

The effectiveness of the speech recognition system was assessed for speakers who were male or female. For reference, five audio samples that had male speakers and five audio samples that had female speakers were used. The results in Table 3 reveal subtle variations in speech recognition performance between male and female speakers. Female speakers had a WER of 0.98 compared to male speakers' WER of 0.99, implying a marginally better recognition of female voices. Table 3 also shows that female voices showed marginally better results in Levenshtein word and character distances, with scores of 45 and 274, respectively. Male voices produced more consistent WER values, with a standard deviation of 0.0063 compared to 0.0207 for females. However, when it comes to Levenshtein Distances, male voices showed significantly larger variability, with significantly higher standard deviations for both Levenshtein word and character distance compared to females.

Table 3: The transcription results for both male and female speakers

Gender	Average WER	Average Levenshtein distance (word level)	Average Levenstein distance (character level)
Female	0.98	45	274
Male	0.99	41	281

6.5 Rate of speech

The acoustic model for the isiXhosa speech recognition system was trained using audio samples that had a maximum speech rate of approximately 70 words per minute (WPM) which is considered to be relatively slow. According to Rodero [32], a speech rate of less than 130 WPM is considered to be slower than normal. A speech rate of between 170 WPM and 190 WPM is considered to be fast and a speech rate of greater than 190 WPM is considered to be faster than normal. Figure 12 represents the results for 15 audio samples of different durations. According to Figure 12, the system consistently produces a high WER, ranging between 0.95 and 1. This indicates that the system does not perform well for the audio samples used despite some samples having a speech rate of lower than 70 WPM. Other factors, such as the lack of certain words from the audio samples in the acoustic model's training vocabulary, are likely impacting this accuracy.

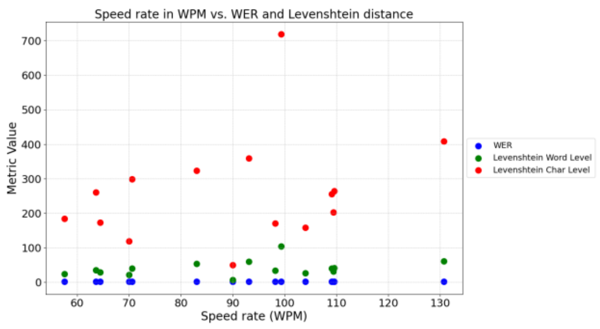


Figure 12: Results for the performance of the speech recognition system for various rates of speech.

6.6 Discussion of results, limitations, and future work

The evaluation of the PocketSphinx Android app for isiXhosa speech recognition revealed critical limitations. Regardless of preprocessing strategies, the system's restriction to words included in its training corpora had a direct influence on transcription accuracy. The absence of words from audio samples in the model's training dictionary or transcription files emphasizes the importance of comprehensive training. For optimal performance, the PocketSphinx speech recognition system should be trained on all possible linguistic sentences and words that are available in isiXhosa.

The system's sensitivity to background noise, a result of its training mostly on clean audio, restricts its applicability across a wide range of real-world scenarios, underlining the need for further improvement to accommodate a variety of settings. Given the widespread use of Android smartphones, guaranteeing reliable performance in a variety of settings, from calm inside locations to busy outdoor environments, becomes critical.

The transcription of informal isiXhosa conversations, particularly those involving code-switching, varied dialects, slang, and colloquialisms, also presented challenges. These constraints, when combined with the phonetic intricacies of isiXhosa and potential recording quality issues, highlight the need for models that are capable of handling informal speech dynamics and multilingual situations. Given the wide range of real-world speech rates, the present acoustic model's preference for slower speech rates reveals adaptation restrictions. Furthermore, the subtle but important recognition differences between male and female speakers necessitate investigation, which may be anchored in different speech frequencies or tonalities.

The evaluation of the PocketSphinx Android app for isiXhosa speech recognition leaves scope for future work. Training on conversational datasets might improve the acoustic model's recognition of informal constructs and typical conversation fillers. A dedicated focus on training with different speech rates and a variety of auditory circumstances, such as a mix of noisy and clean audio, might improve its overall performance. Its robustness can be increased by features like active noise cancellation and environment detection. The system can be made more user-friendly and robust in various auditory conditions by including real-time noise adaptation algorithms and giving users instructions on how to utilize it best. Extensive multilingual training might help with code-switching issues, and a deeper dive into informal lexicons could help with understanding slang and colloquial terminology. Finally, extending the phonetic vocabulary and transcription file will provide the system with a more extensive and inclusive repository of isiXhosa words, guaranteeing a more comprehensive and inclusive speech recognition capacity.

7 CONCLUSIONS

The evaluation of the PocketSphinx Android app for isiXhosa speech recognition revealed considerable flaws in the system's capacity to automatically transcribe unstructured audio on mobile devices. As the use of mobile phones surges in sub-Saharan Africa, the potential of mobile transcription to facilitate electronic resource creation for low-resource South African languages is evident. According to the results and data used, it is presently not possible to produce a high-quality textual corpus on a mobile device using the PocketSphinx speech recognition toolkit without making substantial improvements. Noise sensitivity and transcription of informal conversations are only a few of the difficulties that must be overcome to improve the accuracy of transcription. These problems underscore the urgent need for model improvement, flexible training, adaptable algorithms, and device-centric optimization. While promising in its current condition, there is still a long way to go before mobile devices can be effectively used for the gathering and creation of high-quality linguistic data.

REFERENCES

[1] L. Besacier, E. Barnard, A. Karpov, and T. Schultz. 2014. Automatic speech recognition for under-resourced languages: A survey. In Speech Communication, vol. 56, pp. 85-100.

[2] R. Eiselen and M. J. Puttkammer. 2014. Developing Text Resources for Ten South African Languages. In Language Resource and Evaluation Conference, pp. 3698-3702.

[3] Etienne Barnard, Davel, Marelie H, van Heerden, De Wet, Febe, and Jaco Badenhorst. 2014. The NCHLT Speech Corpus of the South African languages. Nwu.ac.za (2014). DOI:https://researchspace.csir.co.za/dspace/handle/10204/7549

[4] J. Lai, C.-M. Karat, and N. Yankelovich. 2008. Conversational speech interfaces and technologies. In A. Sears & J. A. Jacko (Eds.), The human-computer interaction handbook: Fundamentals, evolving technologies, and emerging applications (2nd ed.), pp. 381–91. New York, NY: Erlbaum.

[5] John M. Levis and Ruslan Suvorov. 2012. Automatic Speech Recognition. ResearchGate. Retrieved September 13, 2023 from https://www.researchgate.net/publication/261287458_Automatic_Speech_Recognition.

[6] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. 2014. Automatic speech recognition for underresourced languages: A survey. Speech Communication 56, 1 (January 2014), 85–100.

[7] Mark J. F. Gales, Kate M. Knill, Anton Ragni, and Shakti P. Rath. 2014. Speech recognition and keyword spotting for low-resource languages: Babel project research at CUED. In 4th Workshop on Spoken Language Technologies for Under- Resourced Languages (SLTU 2014), St. Petersburg, Russia, May 14-16, pp. 16–23.

[8] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. 2014. Automatic speech recognition for under-resourced languages: A survey. Speech Communication 56, (January 2014), 85–100. DOI:https://doi.org/10.1016/j.specom.2013.07.008

[9] Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2021. The State and Fate of Linguistic Diversity and Inclusion in the NLP World. https://arxiv.org/pdf/2004.09095.pdf.

[10] J. C. Aker and I. M. Mbiti. 2010. Mobile Phones and Economic Development in Africa. Journal of Economic Perspectives, pp. 207–232, August 2010.

[11] B. Lakdawala, F. Khan, A. Khan, Y. Tomar, R. Gupta, and A. Shaikh. 2018. Voice to Text transcription using CMUSphinx: A mobile application for healthcare organizations. In Second International Conference on Inventive Communication and Computational Technologies, April 2018.

[12] X. Liu and H. Zhou. 2014. A Chinese Small Vocabulary Offline Speech Recognition System Based on Pocketsphinx in Android Platform. Applied Mechanics and Materials, pp. 267–273, August 2014.

[13] Nic J. de Vries, Marelie H. Davel, Jaco Badenhorst, Willem D. Basson, Febe de Wet, Etienne Barnard, and Alta de Waal. 2014. A smartphone-based ASR data collection tool for under-resourced languages. Speech Communication 56, 1 (January 2014), 119–131. DOI:https://doi.org/10.1016/j.specom.2013.07.001.

[14] Thad Hughes, Kaisuke Nakajima, Linne Ha, Atul Vasu, Pedro Moreno, and Mike Lebeau. 2010. Building transcribed speech corpora quickly and cheaply for many languages. Retrieved March 11, 2023, from https://storage.googleapis.com/pubtools-public-publication-data/pdf/36801.pdf.

[15] Statistics South Africa. 2012. Census 2011 Census in Brief. Statistics South Africa, Pretoria, South Africa. http://www.statssa.gov.za/Census2011/Products/Census_2011_Census_in_brief.pdf.

[16] Johnson, Kristine K. 2011. Xhosa-English Machine Translation: Working with a Low-Resource Language. http://www.cra.org/Activities/craw_archive/dmp/awards/2011/Johnson/kkjohnson_report.pdf.

[17] R. E. Gruhn, W. Minker, and S. Nakamura. 2011. Statistical pronunciation modeling for non-native speech processing. Springer Science & Business Media.

[18] Xhosaroots.com. 2018. Reading: Xhosa Phonology. Retrieved August 2, 2023 from https://xhosaroots.com/insights/xhosa-phonology/.

[19] Nickolay Shmyrev. 2023. Basic concepts of speech recognition. CMUSphinx Open Source Speech Recognition. Retrieved August 2, 2023 from https://cmusphinx.github.io/wiki/tutorialconcepts/.

[20] Pocketsphinx Hjulström and Rickard Hjulström. 2015. Evaluation of a speech recognition system. Retrieved August 2, 2023 from https://www.diva-portal.org/smash/get/diva2:852179/FULLTEXT01.pdf.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and others. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 6 (2012), 82–97.

[22] F. Ruggeri, R. Kenett, and F. W. Faltin. 2007. Encyclopedia of statistics in quality and reliability. John Wiley.

[23] Phil Blunsom. 2004. Hidden Markov Models. Retrieved from https://www.cs.wmich.edu/~alfuqaha/Fall11/cs6570/lectures/hmm-tutorial.pdf.

[24] M. Gales and S. Young, "The application of hidden markov models in speech recognition," Foundations and Trends in Signal Processing, vol. 1, no. 3, pp. 195–304, 2008.

[25] R. E. Gruhn, W. Minker, and S. Nakamura. 2011. Statistical pronunciation modeling for non-native speech processing. Springer Science & Business Media.

[26] S. Russel and P. Norvig. 2014. Artificial Intelligence: A Modern Approach. Pearson Education Limited. vol. 7, pp. 928–935.

[27] Nickolay Shmyrev. 2023. Adapting the default acoustic model. CMUSphinx Open Source Speech Recognition. Retrieved August 3, 2023 from https://cmusphinx.github.io/wiki/tutorialadapt/.

[28] H.-L. Lou. 1995. Implementing the viterbi algorithm. IEEE Signal Processing Magazine, vol. 12, no. 5 (1995), pp. 42–52.

[29] Carnegie Mellon University. 2023. Homepage - CMU - Carnegie Mellon University. Retrieved August 3, 2023 from https://www.cmu.edu/.

[30] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In ICASSP 2006 Proceedings: 2006 IEEE International Conference on Acoustics, Speech and Signal Processing. vol. 1, pp. I–I. IEEE.

[31] M. Anusuya and S. K. Katti. 2010. Speech recognition by machine, a review. arXiv preprint arXiv:1001.2267.

[32] Emma Rodero. 2012. A comparative analysis of speech rate and perception in radio bulletins. Text & Talk 32, 3 (January 2012), 273-288. DOI:https://doi.org/10.1515/text-2012-0019.

[33] MacPorts. 2023. cmuclmtk | MacPorts. https://ports.macports.org/port/cmuclmtk/details/. Accessed August 5, 2023.

[34] Nickolay Shmyrev. 2023. Building a language model. CMUSphinx Open Source Speech Recognition. Retrieved August 5, 2023 from https://cmusphinx.github.io/wiki/tutoriallm/#training-an-arpa-model-with-cmuclmtk.

[35] SADiLaR. 2017. Sadilar.org. https://repo.sadilar.org. Accessed August 5, 2023.

[36] Nickolay Shmyrev. 2023. Training an acoustic model for CMUSphinx. CMUSphinx Open Source Speech Recognition. Retrieved August 6, 2023 from https://cmusphinx.github.io/wiki/tutorialam/#training

[37] K. Beijering, C. Gooskens, and W. Heeringa. 2008. Predicting intelligibility and perceived linguistic distances by means of the levenshtein algorithm. Linguistics in the Netherlands, vol. 15 (2008), pp. 13–24.

[38] Nickolay Shmyrev. 2023. PocketSphinx on Android. CMUSphinx Open Source Speech Recognition. Retrieved August 9, 2023 from https://cmusphinx.github.io/wiki/tutorialandroid/

[39] S. M. A. Anusuya. 2009. Speech Recognition by Machine: A Review. International Journal of Computer Science and Information Security, vol. 6, 3 (2009), pp. 154–162.

[40] R. Prakoso. 2016. Indonesian Automatic Speech Recognition System Using CMUSphinx Toolkit and Limited Dataset. In Proceedings of the International Symposium on Electronics and Smart Devices (ISESD), pp. 283–286.

[41] S. Gales. 2008. The Application of Hidden Markov Models in Speech Recognition. Foundations and Trends in Signal Processing, vol. 1, no. 3, pp. 195-304.

[42] Ian Lane, Alex Waibel, Matthias Eck, and Kay Rottmann. 2010. Tools for Collecting Speech Corpora via Mechanical Turk. In Proceedings of the Association for Computational Linguistics. Retrieved March 11, 2023 from https://aclanthology.org/W10-0729.pdf.

[43] Thomas Reitmaier, Electra Wallington, Dani Kalarikalayil Raju, Ondrej Klejch, Jennifer Pearson, Matt Jones, Peter Bell, and Simon Robinson. 2022. Opportunities and Challenges of Automatic Speech Recognition Systems for Low-Resource Language Speakers. In CHI Conference on Human Factors in Computing Systems (CHI '22). DOI:https://doi.org/10.1145/3491102.3517639.

[44] De Vries, NJ, J Badenhorst, MH Davel, E Barnard, and De Waal, A. 2017. Woefzela - An open-source platform for ASR data collection in the developing world. In Proceedings of the CSIR Research Space. DOI:http://hdl.handle.net/10204/5149.

[45] Jaco Badenhorst. 2012. Quality Measurements for Mobile Data Collection in the Developing World. In Proceedings of the ISCA Speech and Language Technology for Under-Resourced Languages. Retrieved March 9, 2023 from https://www.iscaspeech.org/archive_v0/sltu_2012/papers/su12_139.pdf.

[46] Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. Speech Communication 50, 5 (May 2008), 434-451. DOI:https://doi.org/10.1016/j.specom.2008.01.002.

[47] Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '15). DOI:https://doi.org/10.1109/icassp.2015.7178767.

[48] Vincent Berment. 2004. Methods to computerize “little equipped” languages and groups of languages. Theses, Universite Joseph-Fourier - Grenoble I. Retrieved from https://tel.archives-ouvertes.fr/tel-00006313.
[49] Vadim Kim. 2010. How to Design 10 kHz filter (Using Butterworth filter design). Application notes. Retrieved from https://www.egr.msu.edu/classes/ece480/capstone/fall11/group02/web/Documents/Ho
[50] Carnegie Mellon University. 2023. The CMU Pronouncing Dictionary. Retrieved September 10, 2023 from http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

SUPPLEMENTARY INFORMATION

1. Structured audio results

Audio	Duration	Type	Speed rate (WPM)	Gender	WER	LDW	LDC
Audio 1	38s	News report	93.16	Female	1	59	359
Audio 2	39s	News report	83.08	Female	0.99	53	323
Audio 3	17s	Studio	109.41	Male	1	31	203
Audio 4	64s	New report	99.38	Male	0.98	104	719
Audio 5	4s	Studio	90.00	Male	1	6	50
Audio 6	27s	News report	64.44	Male	1	29	173
Audio 7	33s	News Report	63.64	Male	1	35	260
Audio 8	25s	News report	57.60	Male	1	24	184
Audio 9	34s	News report	70.59	Male	1	40	299
Audio 10	18s	News report	70.00	Male	1	21	119
Audio 11	15s	Studio	104.00	Male	1	26	158
Audio 12	28s	Studio	130.71	Male	1	61	408
Audio 13	23s	Studio	109.57	Female	0.98	41	264
Audio 14	22s	News report	98.18	Female	0.95	34	170
Audio 15	22s	Studio	109.09	Female	1	40	255

2. Casual conversations results

2.1 Original audio

Audio	WER	LDW	LDC
Audio 5	1	43	232
Audio 6	1	184	1062
Audio 7	1	137	719
Audio 13	1	138	872
Audio 16	1	173	883
Audio 17	1	69	507
Audio 18	1	112	749
Audio 19	1	104	768
Audio 20	1	96	675
Audio 21	1	76	543
Audio 25	1	89	629
Audio 26	0.99	115	712
Audio 27	1	80	541
Audio 28	1	74	459
Audio 29	1	81	545
Audio 30	1	32	246
Audio 31	1	152	1217
Audio 32	1	201	1522
Audio 33	1	186	1381
Audio 34	1	197	1502
Audio 35	1	196	1489
Audio 36	1	143	1161
Audio 37	1	137	1153
Audio 38	1	179	1343
Audio 39	1	165	1242
Audio 40	1	156	1307
Audio 41	1	241	2088
Audio 42	1	277	2150
Audio 43	1	280	2174
Audio 44	1	121	886

2.2 Denoised audio

Audio	WER	LDW	LDC
Audio 5	1	43	256
Audio 6	1	184	1065
Audio 7	1	137	761
Audio 13	1	138	905
Audio 16	1	173	918
Audio 17	1	69	505
Audio 18	1	112	829
Audio 19	1	104	781
Audio 20	1	96	669
Audio 21	1	76	559
Audio 25	1	89	637
Audio 26	1	116	798
Audio 27	1	80	578
Audio 28	1	74	494
Audio 29	1	81	557
Audio 30	1	32	245
Audio 31	1	152	1221
Audio 32	1	201	1538
Audio 33	1	186	1386
Audio 34	1	197	1506
Audio 35	1	196	1492
Audio 36	1	143	1162
Audio 37	1	137	1151

Audio 38	1	179	1345
Audio 39	1	165	1247
Audio 40	1	156	1310
Audio 41	1	241	2087
Audio 42	1	277	2155
Audio 43	1	280	2182
Audio 44	1	121	901

2.3 Denoised + Normalized

Audio	**WER**	**LDW**	**LDC**
Audio 5	0.67	29	171
Audio 6	0.67	123	709
Audio 7	0.67	91	510
Audio 13	0.67	92	586
Audio 16	1	173	920
Audio 17	0.67	46	341
Audio 18	0.67	75	521
Audio 19	0.67	69	511
Audio 20	0.67	64	447
Audio 21	0.67	51	371
Audio 25	1	89	626
Audio 26	0.67	77	522
Audio 27	1	80	566
Audio 28	1	74	484
Audio 29	1	81	558
Audio 30	1	32	241
Audio 31	1	152	1219
Audio 32	0.67	134	1025
Audio 33	1	186	1397
Audio 34	1	197	1505
Audio 35	0.67	131	994
Audio 36	1	143	1160
Audio 37	0.67	91	770
Audio 38	1	179	1345
Audio 39	1	165	1248
Audio 40	1	156	1310
Audio 41	1	241	2089
Audio 42	1	277	2156
Audio 43	1	280	2180
Audio 44	1	121	894

3. Background noise results

3.1 Audio sample 1

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	31	279
-4	1	31	271
-8	1	31	252
-12	1	31	241
-16	1	31	259
-20	1	31	260

3.2 Audio sample 2

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	6	63
-4	1	6	63
-8	1	6	56
-12	1	6	57
-16	1	6	54
-20	0.67	4	37

3.3 Audio sample 3

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	26	234
-4	1	26	228
-8	1	26	194
-12	1	26	197
-16	1	26	210
-20	1	26	189

3.4 Audio sample 4

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	61	421
-4	1	61	417
-8	1	61	406
-12	1	61	421
-16	1	61	409
-20	1	61	393

3.5 Audio sample 5

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	42	377
-4	1	42	355
-8	0.99	42	334
-12	0.99	42	310
-16	0.99	42	289
-20	0.98	41	307

3.6 Audio sample 6

Volume (dB)	**WER**	**LDW**	**LDC**
0	1	40	333
-4	1	40	330
-8	0.98	39	283
-12	0.99	40	272
-16	0.98	39	273
-20	0.99	40	272