Universität Zürich[UZH]

Master's Thesis
presented to the Faculty of Arts and Social Sciences of the University of Zurich for the Degree of
Master of Arts

Multilingual Grapheme-to-phoneme Conversion

Author: Deborah N. Jakobi
Matriculation number: 16-054-165

Supervisor: PD Dr. Tanja Samardžić

Co-supervisor: Dr. Steven Moran

Department of Computational Linguistics

Date of submission: (xx.xx.2022)

Abstract

In this master's thesis I tackle the problem of multilingual grapheme-to-phoneme conversion. In a first part, I familiarized myself with the current data situation. Most G2P data is available as word lists that contain mappings from graphemes to phonemes. I created models for 22 different languages using an existing transformer model. As an additional challenge I tried to incorporate phonetic features by extending the input phonemes. The models I trained could keep up with state-of-the-art G2P models and outperformed existing models in different languages. Phonetic transcriptions are not as standardized as writing systems used for written language. As I tested the models on different datasets, this became clear. As for many of my languages there is no result available, my models can serve as a baseline for future research. All scripts and files produced along with this thesis are found in my GitHub repository: `https://github.com/theDebbister/masterThesis`.

Zusammenfassung

In dieser Masterarbeit beschäftige ich mich mit dem Problem der mehrsprachigen Graphem-Phonem-Konvertierung. In einem ersten Teil habe ich mich mit der aktuellen Datenlage vertraut gemacht. Die meisten G2P-Daten sind als Wortlisten verfügbar, die Zuordnungen von Graphemen zu Phonemen enthalten. Ich erstellte Modelle für 22 verschiedene Sprachen, indem ich ein bestehendes Transformer-Modell verwendete. Als zusätzliche Herausforderung habe ich versucht, phonetische Features einzubeziehen, indem ich die Eingabephoneme erweitert habe. Die von mir trainierten Modelle konnten mit den modernsten G2P-Modellen mithalten und übertrafen die bestehenden Modelle in verschiedenen Sprachen. Phonetische Transkriptionen sind nicht so standardisiert wie Schriftsysteme, die für geschriebene Sprache verwendet werden. Dies wurde deutlich, als ich die Modelle an verschiedenen Datensätzen testete. Da für viele meiner Sprachen noch keine Ergebnisse vorliegen, können meine Modelle als Grundlage für künftige Forschungen dienen. Alle Scripts und Dokumente, die ich im Zusammenhang mit dieser Arbeit erarbeitete, sind auf meinem GitHub zu finden: `https://github.com/theDebbister/masterThesis`.

Acknowledgements

First of all, I'd like to thank my supervisor Tanja Samardžić. She did a great job at encouraging me throughout the process and providing me with important knowledge whenever I needed inspiration. Her passion for using precise and understandable language helped me a great deal to pin down the most important insights in an comprehensible way. I was so lucky as to have another co-supervisor, Steven Moran. His in-depth knowledge of Unicode, PHOIBLE and the IPA was an indispensable help to find a way through the jungle of messy data.

In order to digitize some of the text I used that was only available as a scan, there were different people that help me out. I would like to thank Phillip Ströbel from the CL institute at the UZH for his help with OCR technologies. Lysander Jakobi for digitizing the Hebrew orthographic transcription. Florina Vogel and Gazal Hakimifard for helping with the Farsi transcription. Olga Sozinova helped me with both Japanese and Russian. Ximena Gutierrez pointed me to suitable Python libraries for tokenizing different scripts.

Contents

List of Figures

List of Tables

List of Acronyms

IPA	International Phonetic Association
IPAlpha	International Phonetic Alphabet
WER	word error rate
CER	character error rate
PER	phone error rate
G2P	grapheme-to-phoneme
FST	finite-state transducer
seq2seq	sequence-to-sequence
JIPA	Journal of the International Phonetic Association
SIGMORPHON	Special Interest Group on Computational Morphology and Phonology
NLP	natural language processing
OCR	optical character recognition
PoS	part-of-speech
UZH	University of Zurich
WALS	World Atlas of Language Structures
ASR	automatic speech recognition
TTS	text-to-speech
HTR	handwritten text recognition
NWS	*The North Wind and the Sun*
RNN	reccurent neural network
WFST	weighted finite-state transducer

1 Introduction

With the advent of technologies that process huge amounts of data, many linguistic tasks that were originally very tiresome and expensive to do, can now be accomplished much faster. Well known examples for this branch called natural language processing (NLP) are machine translation or search engines. A lot of available tools and consequently research done in this area is concerned with written text. For many scenarios like machine translation, large corpora of written text in many languages are collected that are used to train computational models that solve those tasks. Following, there is an ever-growing set of models that are trained on written text. Often the goal is to reach or outperform human solutions to those various tasks. The corpora that are gathered for tasks like machine translation serve as an example for how the outcome of the task in question should look like. For example, a corpus for machine translation contains a text in one language with its corresponding translation into an other language. It is therefore necessary that this training data represents the language and the task well enough for a machine to reach the required performance. From a linguistic point of view, the question comes up if focusing on written language only can ever represent human language adequately. Most of communication and daily language use happens through speaking. This is a first potential limitation of many (written) language technologies. It is not easy to find out what characteristics of a language can be observed in written representations. Although many NLP tasks use written language corpora exclusively, there are technologies like automatic speech recognition (ASR) or text-to-speech (TTS) that use corpora of spoken language as training data. Spoken language in those cases is mostly represented as phonetic transcriptions. As phonetic transcriptions are strings of characters they are much easier to process for a machine than raw audio files. In order to represent spoken language as phonetic transcriptions, we need to know a lot about how spoken and written language relate. Research on ASR or TTS contributes to a better understanding of the relation between spoken and written language as both of the tasks involve mapping of spoken to written language. But still, this does not answer the question of how well written language represents language in general. The representative power of written text is much less studied. This is where this current thesis connects to cutting-edge research.

I am going to present my attempt of studying a multilingual phonetic corpus and comparing it to its written-text version.

1.1 Research questions & goals

I will try to answer the following question as the result of my thesis: **Is it essential for the study of multilingual corpora to perform analyses on phonetic text (i.e. speech representations) rather than only written text?** This question addresses many different topics from the field of linguistics but also more technical aspects as I have mentioned in the paragraph above. None of these topics is trivial and the question cannot be answered easily. While in this thesis I cannot possibly discuss everything from the use of phonetic transcriptions to the nature of human language use, the aim is to make a step into the direction of quantifying the representative power of written text. The goal is now to collect a corpus of phonetic transcriptions in various languages based on an already existing written corpus. Once I have the phonetic corpus, it is then possible to perform analyses on the phonetic corpus which can then be compared to the same analyses on the written corpus. In order to add a phonetic corpus to an already existing one, various steps need to be performed which are outlined below:

1. **Data collection:** The given dataset contains no phonetic transcriptions of those 100 languages. The first step is to find already existing data.
2. **Computational model to create phonetic transcriptions:** As existing data will not be available in sufficient amounts to perform meaningful analysis, the next step is to actually create phonetic transcriptions of as many languages as possible of the corpus. This will require to create and train a computational model.
3. **Tests and analyses:** Once the transcriptions have been obtained, the newly created phonetic corpus can be analysed and tests can be performed on the phonetic data.

The first two steps will already take up quite a lot of time which is hard to estimate in advance. It is not clear how much of the third step can be accomplished. I still think it is important to keep the bigger picture in mind and be aware of the overall importance of the topic. By performing the steps above I will have to tackle different challenges. Some of them are already known now, but there will be many more that will only become apparent while working on the tasks.

1. What types of phonetic data is available and how can it be used?
2. Which computational models can be used to create phonetic transcriptions automatically?
3. How can we use phonetic features to create phonetic transcriptions?

All of these challenges above are interesting and important, but the third challenge is of particular importance as it is a new approach to creating phonetic transcriptions.

1.2 Thesis structure

The thesis is subdivided into six chapters of which the first chapter is this introduction. Chapter 2 covers the linguistic basics that are necessary to understand the topic. It presents the linguistic foundation of phonetics and phonology and an introduction to corpus linguistics or rather corpus phonetics. Chapter 3 sets the boundaries of the technical background. I will present an overview of the possibilities for automated creation of phonetic transcriptions. In chapter 4, I document my first practical experiments which are concerned with data collection. It contains a descriptions of the data I will later use to create models for phonetic transcriptions. In chapter 5 I will present one of my experiments where I compare the word length of phonetic and written text. Chapter 6 presents my own experiments to create phonetic transcriptions of the corpus. I will present all models that I created and their results. Chapter 7 summarizes my findings and presents ideas for future research.

2 Representations of Written and Spoken Language

From a linguistic point of view there are two high-level concepts that are important for this thesis. The first and most significant one is that of the **relation of written and spoken language**. Both spoken and written language are a way of representing a language in general. I would like to find out what traditional linguistics states about the relation of written and spoken language. Both written and spoken language are *valid* representations of the same language. As we will see later in this chapter, mapping a spoken language to its written representation is far from easy and never perfect. Elaborated writing systems like we are used to today came only much later compared to language in general[1] [?].

Whenever we study language we look at samples of that language. It is simply impossible to study an *entire* language as we would need all texts that were ever produced in that language. Consequently, we need to ask ourselves how much and what material of a language is enough, to study it properly. In order to do that we can make use of techniques and methods from corpus linguistics. In linguistics and also in computational linguistics we typically talk about a corpus when we talk about a huge dataset that is used to represent a language. It does not matter if the corpus contains written or spoken samples of a language although it is mostly written [?]. Corpus linguistics allows for both qualitative and quantitative analysis of text. Due to recent technological advancement it has become possible to store large digital collections of speech recordings and their aligned transcriptions. These possibilities gave rise to a wider acknowledgement of corpus phonetics. Corpus phonetics deals with an abundance of linguistic variation. In addition to language, style or vocabulary variation, there are differences in dialect and idiolect, physiological state of the speakers and their attitude [??]. In section 4.3.2, I will present an interesting study on corpus phonetics [?]. The authors of the study aim to answer how much phonetic data is needed to cover all sounds of a language.

[1] `https://www.youtube.com/watch?v=-sUUWyo4RZQ&list=PL8dPuuaLjXtP5mp25nStsuDzk2blncJDW&index=18`

The second key concept is that of **multilingual analysis**. The corpus underlying this thesis is multilingual. I will introduce this corpus later in section 4.2. While a multilingual corpus in itself is not uncommon, the specific goal of the team maintaining the corpus is to compare the languages and study their similarities. Comparing languages and studying their similarities and differences is part of a well-established branch of traditional linguistics called comparative linguistics. For example, **?** performed a study on a multilingual corpus comparing different subword tokenizations. They compared 47 languages to find out how languages can be tokenized such that it leads to similar distributions among that languages. This analysis was performed on written language and is an example for what could be done on a phonetic corpus as well.

While there is a lot that could be said about any of these topics, I am not going into more detail about these. The point I would like to make is that the task of creating phonetic transcriptions of a multilingual corpus with the long-term objective of comparing written and spoken language is deeply rooted in linguistics and not at all trivial. In the following sections I will mostly write about phonetics and phonology in general and writing systems.

2.1 Phonetics and phonology

Given that phonetics and phonology is a sub-area of traditional linguistics and often only touched on superficially in computational linguistics, I will summarise the most important assumptions and terms that are necessary to understand what phonetic transcriptions are. A very important terminological distinction is between **phonetics** and **phonology**. While phonetics refers to the study of actual sounds, phonology refers to the study of sound *systems*. In phonetics, it is not so much important what the different sounds mean, but how they are produced and perceived and what different sounds a human being can produce and perceive at all. When it comes to human communication using spoken language, many sounds that can be produced are not actually meaningful. A human being is perfectly capable of producing many different sounds but some just do not mean anything to other people. The study of sound systems, phonology, is used to describe the set of sounds that make up a language. The sounds within a sound system of a language can be used to construct meaningful utterances. For this thesis, phonology is more important.

Example 2.1 shows how the word ‘request’ can be pronounced in English. The pronunciation is written in the International Phonetic Alphabet (IPAlpha) which I will explain in more depth in section 2.3. For now it is important to note that the

same word can be pronounced differently.

(2.1) Different pronunciations of the English word 'request' (except for the last example). The spaces indicate how the letters can be mapped to the phonetic symbols on the right side:

(a) r e qu e s t → [ɹ ɪ kw ɛ s t]

(b) r e qu e s t → [ɾ ɪ kw ɛ s t]

(c) r e qu e s t → [r ɪ kw ɛ s t]

(d) b e qu e s t → [b ɪ kw ɛ s t]

Example 2.1 a shows how the word 'request' is pronounced in British English. Even someone that does not know the IPAlpha can see that the letter 'r' can be mapped to the pronunciation symbol [ɹ]. Another English speaker might pronounce the letter 'r' a bit different as in example 2.1 b. We can see that the pronunciation symbol [ɾ] for the 'r' differs from the one in the first example. However, an English speaker would still understand the word 'request' as the different pronunciation does not change the meaning of the word. In phonetic terms we would say that in the British English sound system there exists an abstract sound /ɹ/ which is called the **phoneme** /ɹ/. Note that we write a phoneme between slashes. Also, in our case there exist two **phones** [ɹ] and [ɾ] which are specific sounds that can be used to pronounce the phoneme /ɹ/. No matter which phone I use, it does not change the meaning of the word. Note that we write a phone between square brackets. In British English, [ɾ] is an **allophone** of the phoneme /ɹ/. Allophones are specific phones that are concrete realizations of the same phoneme. Realization in this case just means that it is a sound in an actual utterance. Allophones of the same phoneme can be used interchangably without changing the meaning of the utterance. Very importantly, allophones are language specific. In other languages I cannot use the same phones to pronounce a phoneme. In example 2.1 c, the 'r' is mapped to the phone [r] which is part of the Spanish sound system. [r] is not an allophone of /ɹ/ in British English as typically British English speakers do not say the 'r' like in Spanish. The phone [r] does not exist in the British English sound system. Still, a British English speaker would understand the word 'request' uttered with the Spanish [r]. However, this is due to the inherent similarity of the Spanish [r] and the British [ɹ] and must not be the case for other sounds used in other languages. If we replaced the [ɹ] by another phone that is not an allophone of /ɹ/ but of a different phoneme in the British English sound system (see example 2.1 d) we can see that the meaning of the utterance changes. This change in meaning is because in examples 2.1 a and 2.1 b the phones were realizations of the same phoneme, while in example 2.1 d I introduced

a new phoneme. Unlike allophones, phonemes in a language's sound system can be used to change the meaning of an utterance. Each language has a **phoneme inventory** of abstract sounds that exist in the language's sound system that can be used to construct meaningful utterances. Apart from the phoneme inventory of a language, there is a set of phones which are concrete realizations of abstract phonemes. Further, for each phoneme in a language, there exists a subset of all phones which are allophones of that phoneme. Allophones can be used to replace another allophone in the same set without changing the meaning of an utterance.

When it comes to phonetic transcriptions, the difference between phoneme, phone and allophone is very important. It is important to note at this point that the terms phonetic and phonemic, and phone and phoneme are sometimes used interchangeably. Their linguistic definition as given above is relatively clear while the definition on the computational side is often less strict. In section 2.3, I will elaborate more on this problem.

In the following, I will outline different terms and concepts that appear a lot in research and literature concerned with phonetics and phonetic transcriptions.

Vowels and consonants In order to categorize the sounds in the sound systems of different languages, sounds are typically split into vowels and consonants. The terms to describe both vowels and consonants are inspired by the physical position of the tongue and the mouth when the sound is produced. To describe vowels, we use the position of the tongue in the mouth and if the lips are rounded or not. Using those two categories enables us to distinguish every possible vowel. The position of the tongue in the mouth is described along two axes: The tongue can be moved from the back of the mouth cavity to the front. This is the back-front axis. The second axis is that the tongue can be moved up and down in the mouth. In phonetics, up and down is referred to as close and open. A vowel can then, for example, be described as close-back rounded. This means that in order to produce the sound, a person needs to move the tongue up and back in the mouth and round the lips. Figure 1 shows the vowel chart how it is usually represented in the IPAlpha. More on this special alphabet follows in section 2.3.

Consonants are defined by the place and the manner of their physical production. The place, again, refers to the position of the tongue in the mouth and the overall form of the vocal tract. While for vowels we used the open-close and back-front schema to describe the position of the tongue, the consonant tongue positions are categorized differently. Table 1 shows all consonant place categories.

bilabial	labiodental	dental	alveolar
postalveolar	retroflex	palatal	velar
uvular	pharyngeal	glottal	

Table 1: This table displays all tongue positions and shapes of the vocal tract that are used to produce consonants.

Unlike for vowels, for consonants the entire vocal tract, the lips and the tongue are used to block the air and make it flow in a specific way. For example, a dental consonant means that the tip of the tongue is pressed against the upper front teeth. For palatal consonants, the body of the tongue is pressed against the hard palate in the back of the mouth cavity. The manner of consonant production describes very precisely how the air is lead through the mouth to produce a sound[2]. Examples for the manner of a consonant are plosive or trill. For a trill the tongue needs to move in a vibrating way which consequently makes the air vibrate. Note that the place is not changed by this vibration. The tip of the tongue can still be pressed against the teeth or be in the back of the mouth while vibrating. For a plosive sound we first completely block the air such that no air can leave the mouth and then the air is pushed out of the mouth in a fast manner, a bit like an explosion, therefore the name. The complete consonant chart is depicted in figure 1 as well and table 2 shows all consonant manner categories.

plosive	nasal	trill	tap or flap
fricative	lateral fricative	approximant	lateral approximant

Table 2: This table displays manner categories of consonants.

Once I have introduced the IPAlpha in section 2.3, I will give examples of how the IPAlpha helps us to write sounds with respect to the above introduced categories of vowels and consonants and how it differs from orthographic representations.

Syllables Letters can be grouped into larger units called *syllables*. Syllables can be an entire word or a part of a word. A syllable can be subdivided into different parts called *onset*, *nucleus* and *coda*. For every syllable in every language it is true that the nucleus cannot be empty. The onset and the coda can be empty. Example 2.2 shows English words with syllable boundaries.

[2] `https://www.youtube.com/watch?v=vyea8Ph9BOM`

(2.2) English syllables: the hyphen denotes a syllable boundary.

(a) the-sis

(b) o-ver

In English, the onset is typically a sequence of consonants. The nucleus is a sequence of vowels or just one vowel. The coda is another sequence of consonants. The second syllable in example 2.2 a) follows this structure exactly: 's' is a consonant followed by a vowel 'i' followed by another consonant 's'. The first syllable in example 2.2 b) on the other hand, consists only of the nucleus, in this case the vowel 'o'. Other than this three-part structure, syllables are organized very differently in different languages [?]. For computational analysis it is important that syllables are a way of segmenting written or spoken language. They are larger than individual letters or phonemes but often still smaller than individual morphemes or words.

Diphthongs A diphthong is a sequence of vowels that is considered as just one phoneme if it is within one syllable. If a syllable ends with a vowel and the next one starts with a vowel, this vowel sequence is not called a diphthong. An example is the German word 'Chaos'. The two vowels in the middle are not a diphthong as there is a syllable boundary right in their middle: 'Cha-os'. A word like 'aus' contains a diphthong as it exists of only one syllable [?].

Suprasegmentals Apart from individual sounds, there are features of spoken language like stress or intonation. Those are referred to as suprasegmentals. They are often related to syllables. For example, we can put stress on a different syllable or raise the pitch. Semantically, some suprasegmentals in some languages distinguish meanings, some do not. A special case are tones. Tones are a special way of intonation. In some languages like Chinese or many African languages tones are used to distinguish meaning while in most European languages, the concept of tones does not exist [?].

2.2 Mappings of written and spoken language

Unlike spoken language that was a part of human interaction all the time, writing systems only developed over time. There are different writing systems that developed in different places at different times. The structure of the spoken language, the cultural context or the tools that were at hand to write are a few of many factors

that influenced the emergence of a specific writing system. In general, we can think of writing systems as mappings from sounds to written symbols. The systems used to represent sounds in different languages do not uniquely map a symbol to one specific phoneme. Most of the time, there is a standard pronunciation of each symbol in every language. However, as we have seen above, there is the notion of allophones which means that every sound can be replaced by some other sounds which are also understood in this language. These explanations make clear that the mapping of written symbols to spoken sounds in various languages is complex.

When taking a step back, we can see that a single symbol can represent either a single phoneme, a syllable or a word or even something in between. Different writing systems developed in parallel which means that today, we have an abundance of different strategies to put sounds into writing. Not all writing systems, which are typically called scripts, can be treated the same and this most certainly has implications on models to create phonetic transcriptions. Each major script that is used today will be presented below [3]. In the following I will use the terms letter or sign when referring to an individual symbol that is part of a writing system. In section 3.1, I will introduce a more fine-grained terminology when referring to symbols in writing systems and their computational representation.

ALPHABET When a letter maps roughly to one phoneme, we call the writing system an alphabet. In German, for example, the writing system is the Latin alphabet. The Latin alphabet is used for many different languages in western Europe and those languages that were influenced by colonisation. There are other alphabets like the Cyrillic or the Greek alphabet. If a language uses an alphabet this does not mean that each letter maps to exactly one phoneme. In fact, one letter can have many different realizations as example 2.3 shows.

(2.3) The examples show the different realizations of the English letter sequence 'ough'[4]. The first part is the letter sequence and the second part the phonetic transcription. The parts marked in red can be mapped onto each other.

(a) tough [tʌf]

(b) cough [kɒf]

(c) though [ðəʊ]

[3] https://www.youtube.com/watch?v=-sUUWyo4RZQ&list=PL8dPuuaLjXtP5mp25nStsuDzk2blncJDW&index=18

[4] https://www.youtube.com/watch?v=vyea8Ph9BOM

(d) through [θruː]

(e) bough [baʊ]

(f) brought [brɔːt]

The above examples show that it is not always clear how to pronounce a certain letter sequence. There is no simple one-to-one mapping from one letter or a sequence of letters to one phone or a sequence of phones within the English language. Let alone within all languages that use the Latin alphabet. In addition, alphabets typically have diacritic marks that can be used to extend the main letters. As it is the case with the German 'ä' which is a slightly changed version of the letter 'a' and pronounced differently. Just as with single letters, also diacritic marks cannot simply be mapped to a phone.

Abjad A special variant of an alphabet is abjad. Abjad represents only consonants and no vowels. This means that vowels need to be added while reading. Again, this means that there is a lot of ambiguity as it is not always clear which vowel should be added if there is no context. Semitic languages like Hebrew or Arabic make use of abjad.

(2.4) The example shows Hebrew words that are first mapped to the Latin alphabet, then to the Latin alphabet including vowels and in the end to the English translation.

(a) בצלם bzlm bzelem name of an association

(b) בצלם bzlm bzalam 'their onion'

Example 2.4 shows that in both cases each letter maps to the same consonant but it can be completed with different vowels. The words presented above do not have the same meaning depending on the vowels added although their letter sequence looks exactly the same.

Syllabary In syllabaries, a letter, or rather a sign in this case, represents a syllable instead of a single sound. An example is the Korean script. Syllabaries typically do not have any internal ambiguities in their pronunciation as one sign maps to exactly one phoneme.

Logographic systems Logographic systems represent entire words or morphemes as signs. Chinese is an example for a logographic system. We cannot break down Chinese signs into single morphemes or letters. One sign, which is called a logogram, is often pronounced in the same way regardless of the context of the sign. This means that the pronunciation of one sign is less ambiguous

than, for example, the pronunciation of one letter in an alphabet which can be mapped to different phonemes in different contexts. Logographic systems often have thousands of signs, while alphabets typically have less than one hundred letters. For each logogram, one must learn how to pronounce it.

As the examples above show, it is very difficult to have a clear mapping from sounds to written symbols. For most languages it is not possible to derive the exact pronunciation from the written symbols. Even for languages that use the same writing system. Many of the pronunciation rules of a language are based on convention. Speakers of a language just *know* how to pronounce a word. Still, there can arise heated debates about the correct pronunciation of certain words within a language. An example are Swiss German dialects. In Switzerland, dialects are considered very important by many people. Often, every town has their own dialect which people are proud of. This leads to an abundance of different pronunciations for one word. People are perfectly capable of understanding other dialects but just use their own pronunciation when it comes to speaking. As the personal way of speaking is often considered 'normal', there are a lot of playful but also more serious discussions about correct pronunciations.

Apart from different conventions, spoken and written languages change differently over time. Spoken languages are typically more flexible and ready to change while their written representation often stays the same [**?**]. This can lead to official governmental interventions like the German orthography reform of 1996 that intended to adapt the German spelling to represent the German pronunciation more adequately. Also, major inventions like printing machines gave rise to standardization of writing systems as reading and writing became more common[5].

2.3 The International Phonetic Alphabet (IPA)

The IPAlpha is a special alphabet where each letter is intended to represent exactly one phoneme[6] [**?**]. Figure 1 shows the full IPAlpha chart including all characters that the International Phonetic Association (IPA) decided to use. As usual, reality is more complex than what we wish it to be. Even with the IPAlpha there are inconsistencies. Although the IPAlpha seems very complete, there are still sounds that cannot be represented using the IPAlpha. This becomes clear when, for example, looking at the vowel chart (see figure 1). The tongue does not 'click into place'

[5] `https://www.youtube.com/watch?v=-sUUWyo4RZQ&list=PL8dPuuaLjXtP5mp25nStsuDzk2blncJDW&index=18`

[6] ibid.

for the vowels on the chart. Vowel characterisation happens on a continuum. This means that it is always possible to characterize a vowel as in between two vowels on the chart. But for such a vowel there does not exist an official symbol.

When sounds are mapped to written symbols, we refer to this process as creating a transcription. When creating such transcriptions using the IPAlpha, there are different levels of detail. Not all transcriptions represent the sounds in equal detail. Generally, there is the distinction of **broad** and **narrow** transcription. These two transcription types go back to the linguistic distinction of phone and phoneme as I have explained in section 2.1. Also, as I have pointed out, there is the notion of allophones that can all be used to represent a specific phoneme without changing the meaning of an utterance. When we speak about a broad transcription, also called **phonemic** transcription, we do not care about the exact sounds. If a person uses an allophone to pronounce the word, we still write down the respective phoneme. If we want a narrow transcription, also called **phonetic** transcription, we care about every detail of the pronunciation. If a person uses an allophone, we note down that allophone and not the respective phoneme. Strictly speaking, broad transcriptions are not allowed to contain allophones but should write the respective phoneme of that language.

Not using any allophones in broad transcriptions will not always be the case when it comes to data used in language technology [?]. Example 2.5 a shows a German broad transcription of the word 'Anrede'. Example 2.5 b shows a broad transcription of the German word 'Anredefall' which should be pronounced the same except for the last additional part. However, as we can see in the example, the first part marked in red looks different. In the German sound system, the [r] is actually an allophone of [ʁ]. Correctly speaking, the second example is wrong as in broad transcriptions, we are not allowed to use allophones. The correct broad transcription of 'Anredefall' would be as shown in example 2.5 c. The example is from the actual datasets I will be using later (see section 4.3).

(2.5) (a) Anrede → /a n ʁ eː d ə/

(b) Anredefall → /a n r eː d ə f a l/

(c) Anredefall → /a n ʁ eː d ə f a l/

Broad transcriptions are less complex and usually easier to create and understand as they contain fewer details. Narrow transcriptions present every speaker individual or dialectal sounds as exactly as possible and are consequently more difficult to create. Narrow and broad transcriptions can diverge greatly. It is important to treat broad and narrow transcriptions as two different kinds of transcriptions.

(2.6) /pɪˈkʊ kəˈzɚf/

(2.7) [pɪˈkʰʊ kʰəˈzɚf]

Example 2.7 is a narrow (phonetic) transcription of the beginning of the Mapudungun version of the short story *The North Wind and the Sun*. The same text is transcribed broadly (phonemic) in example 2.6. As becomes clear in this example, the narrow transcriptions is longer as it contains more different characters. In this case it is only the superscript ^{h} that is different. The problem, with especially the narrow transcriptions, is that the transcriber still needs to define what narrow means in a specific case. One could argue that there are as many narrow transcriptions of a language as there are speakers of that language. This becomes tricky when given a task to automatically transcribe text. The training data might employ one definition of narrow, while there are texts in the test set that might follow another definition. This is mostly important when talking about data preprocessing and cleaning.

In the example 2.5, we can also observe some differences between the IPAlpha version and the orthographic example. In subexample 2.5c, I marked different letters in orange. On the orthographic side, both letters look exactly the same. On the phonetic side, there are two different symbols used to represent the same letter. This again shows how ambiguous pronunciation is. The IPAlpha provides a way to dissolve some of these ambiguities as the example shows. In addition to resolving ambiguities, phonetic transcriptions using the IPAlpha can add additional information that is not typically represented in orthographic text. The length marker in /eː/ denoted by a symbol similar to the colon, is such an example. There is no way to see the lengthening of the 'e' in the orthographic text, while it can easily be observed in the phonetic text.

2.4 Phonetic features

As every language has a sound system and a set of phonemes that are part of that system, it is absolutely necessary that we can describe a phoneme very precisely. In computational linguistic analysis you would typically refer to a description of a phoneme as a set of features of that phoneme. For example, I could say that I use one feature to describe every phoneme. This one feature is if the phoneme is a consonant or not. If it is not a consonant, it follows that it is a vowel.

(2.8) The examples show how we can describe different phonemes using features.

'1' stands for: 'is-a-consonant'. '0' stands for: 'is-not-a-consonant' (which means that it is a vowel).

(a) /ə/ → 0

(b) /ʀ/ → 1

(c) /a/ → 0

With only one feature I cannot really distinguish different phonemes as becomes clear when looking at example 2.8. In this case, I can in fact only distinguish vowels and consonants. Therefore we need to add more features to describe phonemes better. In linguistics it is very common to use the vowel and consonant schemes described above to describe each phoneme. But it is also possible to use a feature like 'syllabic' that tells us whether a sound can be used as a syllable on its own or not. Or we could use a feature like 'suprasegmental' that tells use if a phoneme is a suprasegmental. In section 4.4 I will introduce a specific set of features that allows to describe many phonemes uniquely.

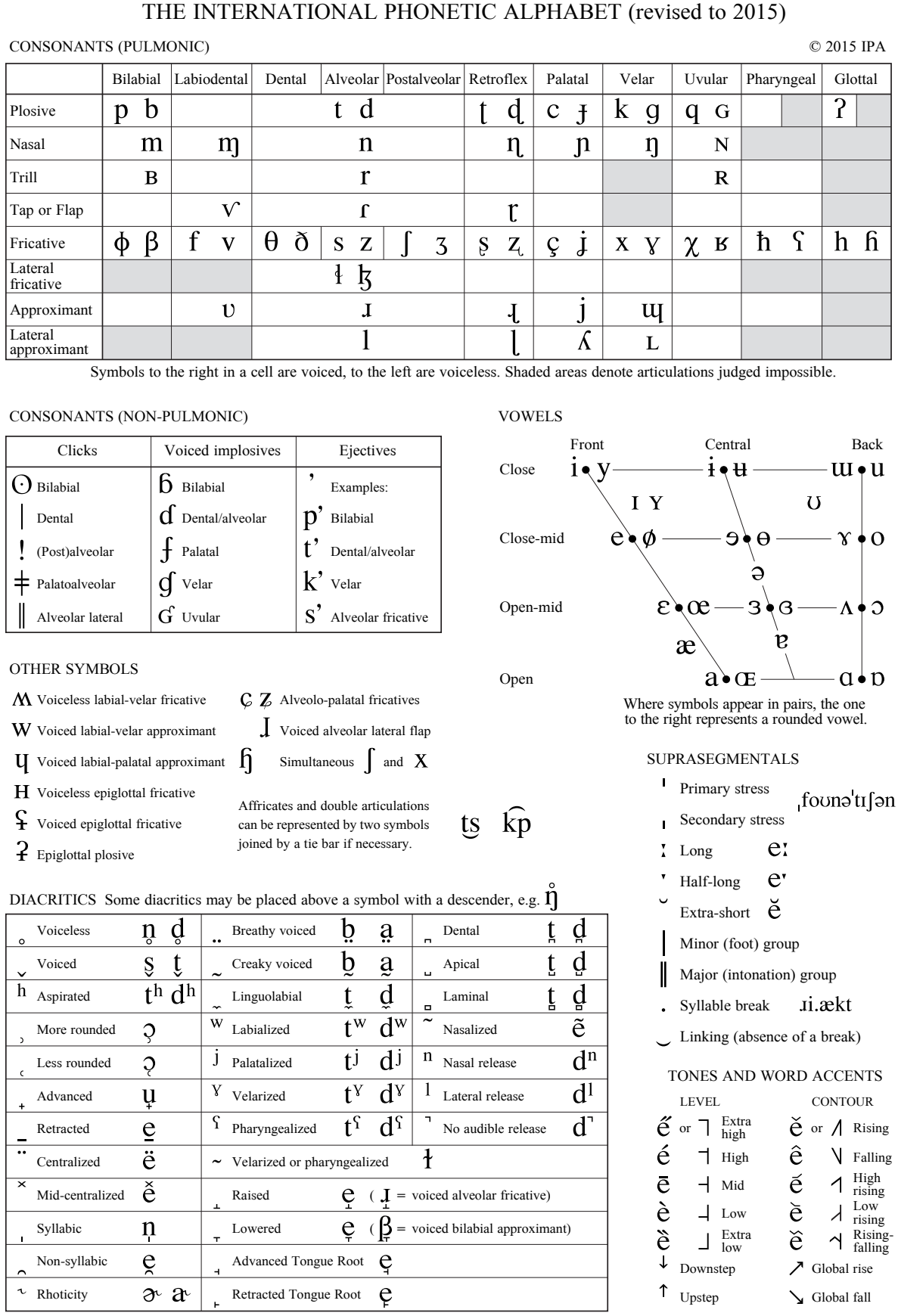

Figure 1: This is the full IPA chart, last updated in 2015

3 Computational Models to Create Phonetic Transcriptions

In this chapter I present the technical background that is needed to understand how we can create phonetic transcriptions using computational models. I will first set the basis and dive into general architectures and frameworks that are commonly used and then present current state-of-the-art models. In chapter 2, I have pointed out that there are many different written representations of language. In the first section of this chapter, I am going to explain how a computer represents language and different scripts.

3.1 Unicode and the International Phonetic Alphabet

Computational representations of languages are very important as no matter what task we try to solve, we need to understand how written text is processed by a computer. For the present thesis, this topic is relevant for multiple reasons:

1. The IPAlpha contains many special characters and many diacritics.
2. The language data is available in many different scripts.

It is crucial that all data files, be it phonetic or 'normal' scripts, are formatted and read correctly by any machine that needs to process them. There exists a standard that is widely used to computationally represent written text. This standard is called 'The Unicode Standard', in the following referred to as simply 'Unicode'.

When it comes to representing letters or signs in a machine-readable format things get very tricky, very quickly. In order to understand this fundamental problem it is necessary to understand the basic concept behind Unicode and how characters are represented behind the scenes. **?** present an overview in their book. As discussed in chapter 2, there are many different kinds of what I referred to as letters or signs. From now on, I will use grapheme to denote the smallest meaningful element of any writing system. Grapheme does not imply any specific writing system nor does

it get into the way of Unicode terminology. If I wish to distinguish the Unicode specifications I will use the correct Unicode term as described in this section.

Just as a human writer must be able to uniquely identify each different grapheme, so must a computer. In Unicode, graphemes are mapped to unique numbers that can be rendered differently depending on the font and the context. There are different stages of representation until a grapheme can be represented on screen:

CODE POINT A unique numerical, non-negative value usually expressed as a hexadecimal number (U+0000). Allows one-to-one mapping between characters and codes. Each code point has a set of properties attributed to it. Properties like the script, uppercase or not, etc.

CHARACTER An abstract representation of the shape of the grapheme. Can in theory not be represented visually, as this includes a font. A Unicode character is *not* the same as what we would call a grapheme in different writing systems.

GLYPH The rendered and therefore visual representation of one or more Unicode characters that can be identified by its code point(s). A glyph is rendered in a specific font in a specific context. For me as a user, two glyphs can look completely different. Still, both glyphs might have the exact same code point and thus Unicode treats them as the exact same character.

(3.1) (a) 𝔢 : Unicode code point U+0065

(b) e : Unicode code point U+0065

In example 3.1, both subexamples represent the same Unicode code point. Their glyphs look different because it is a different font that is used for both of them. Sometimes one character is represented as two glyphs. It is important to note here, that the exact visual representation of a glyph is not at all defined by its code point. This means that the exact same glyph can represent more than one code points. This happens sometimes in the IPAlpha. An example is the post-alveolar click:

(3.2) (a) ! : this glyph represents an exclamation mark with unicode code point U+0021.

(b) ! : this glyph represents a post-alveolar click with unicode code point U+01C3.

It is striking that both glyphs in example 3.2 look exactly the same. This ambiguity becomes important when, for example, I want to count the different characters in a text.

Unicode code points are often organized in blocks. Blocks are not needed by Unicode but they help users to organize characters that are related into larger groups. For example: all letters of the Latin alphabet are represented by a code point. Instead of just using any random code point for all letters it makes sense to use consecutive numbers as code points. Then I can say that all basic Latin letters have a code point in the range of U+0000 to U+007F. Those blocks are helpful although not always consistent. The IPAlpha is represented in a basic block but many IPAlpha symbols are actually found in other blocks.

Further confusion often arises from the fact, that one human-perceived grapheme is sometimes represented as more than one code point.

GRAPHEME CLUSTERS A grapheme cluster is one visual letter that is represented in Unicode as more than one code point. This is the case for diacritic marks. Diacritic marks are characters that cannot really exists without their base character. A problem with grapheme clusters is that some diacritic marks are underspecified. This means that when we want to split into clusters, we do not know if that character belongs to the left or the right base character. This is the case for many characters in the IPAlpha. Unicode does not specify these but leaves it to the user to create tailored grapheme clusters. An example is the superscript aspiration /ʰ/. It is typically written after the respective aspirated character (/pʰ/), but there is no reason why it could not be written before the respective character (/ʰp/), except for convention.

PRECOMPOSED CHARACTERS Note that sometimes, grapheme clusters can be precomposed and the combination of those two or more characters is assigned a new number. These clusters can be problematic if in a specific context, the graphemes should not be clustered but read separately. An example is the German 'ä', illustrated in example 3.3.

(3.3) This example shows a precomposed character and the different characters it is made of. Note the different code points. If the character was not precomposed, it would simply have two code points similar to sub-example (d).

(a) ä : U+00E4

(b) a : U+0061

(c) ¨ : U+00A8

(d) e̥ : U+0065 U+0325

An additional challenge is that of picking the right font. Our standard font format can only contain about half of all the Unicode code points. It is therefore simply not possible to display the entire set of Unicode characters with one font. Many problems encountered with displaying writing systems are somehow connected to the font rather than Unicode itself [?].

Segments library ? present a Python library called segments[1] that can be used to process phonetic text. It includes a method to segment IPAlpha strings into grapheme clusters. Those grapheme clusters are designed in a way such that they represent sound units better. Example 3.4 illustrates how this segmentation looks like. Different segments are separated by white space.

(3.4) /j uː z ɪ d͡ʒ/

In the IPAlpha there exists a length marker to denote vowel lengthening. The vowel and the length marker are displayed as one segment. Characters connected by a tie bar are displayed as one segment as well. This segmentation is adapted to the actual sounds as it is not possible to pronounce a vowel and the length marker as a separated sound. However, from a Unicode perspective those are both different code points. Later when processing the data I will make use of this library.

Unicode normalization forms

The above explanations make clear that there are considerable differences in what a human reader perceives and in what happens in the background. Unicode therefore provides normalization forms that can help to process written data. Unicode publishes extensive explanations along with their standards which also includes those normalization forms. I will therefore not explain everything in full detail as this is done already online [?].

There are two important aspects to normalization. One is that we can have decomposed or composed characters. The second aspect of normalization is that we have a compatibility form and a non-compatibility form. Based on these two aspects there are four normalization forms:

NFD : Canonical Decomposition

NFC : Canonical Decomposition, followed by Canonical composition

[1] `https://pypi.org/project/segments/`

NFKD : Compatibility Decomposition

NFKC : Compatibility Decomposition, followed by Canonical Composition

What is important is that each normalization form results in very different behaviour if a text is processed. What the normalization form of a text is, is not necessarily visible to the human reader. All of this happens at the level of internal representation of a computer which means that it happens at the level of Unicode code points. In a decomposed string, we split the characters into their individual components. This means that characters with diacritics are split up into two or more characters. This again means that a character that is assigned one code point in a composed normalization form can in a decomposed form have more than one code points. As this is very difficult to understand on a abstract level, figures 2 and 3 show what each normalization form changes about the characters and code points.

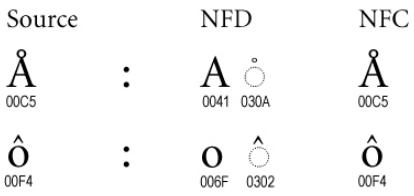

Figure 2: This figure illustrates what composed and decomposed normalization forms change about the character representation. The code points are below the glyphs[2].

In a composed normalization usually precomposed forms are kept. This means that some parts can be similar or the same to the decomposed version. If for a character there exists a precomposed version, the character is shown precomposed and not split up into its individual parts (see figure 3, first example).

If a normalization is according to compatibility decomposition, this means that any formatting is removed such that we receive the underlying character in its original form. Superscript characters are then shown as example 3.5 illustrates.

(3.5) $^{\text{h}} \rightarrow \text{h}$

How exactly these normalization forms work is not always equally important, but what is absolutely crucial is to make sure that when characters are compared or counted, the same normalization form is used. This is because when counting characters, we do not count the human perceivable glyphs, but Unicode code points.

In Python, it is possible to convert a text that is read from a file in a normalization form into another normalization form. None of this process is visible for the human reader. Similarly, it is not possible for a human reader to see what the normalization

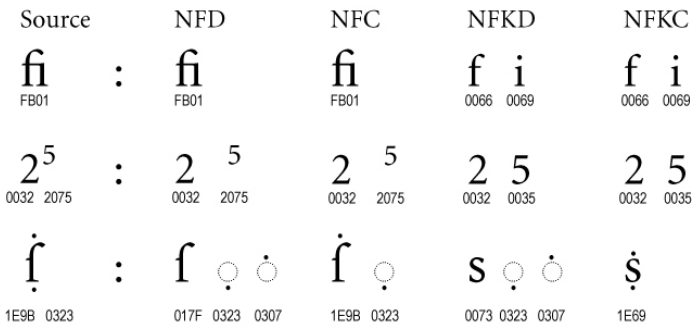

Figure 3: This figure illustrates the difference between all four Unicode normalization forms. The code points are below the glyphs[3].

form of a text is. This is because computational text representations depend on Unicode code points. But what the human reader perceives are the individual glyphs. And has I have pointed out, glyphs have nothing to do with the underlying Unicode representation.

Understanding Unicode and its normalization forms is the first step to be able to create computational models for phonetic transcriptions as we need to pass text data as input to the model. In the following section, I will focus on the actual model that performs the conversion from orthographic text to phonetic text.

3.2 G2P as sequence-to-sequence task

No matter what computational model I use, creating phonetic transcriptions can be referred to as a sequence-to-sequence (seq2seq) task. There are many other seq2seq NLP tasks whose goal it is to transform a sequence of symbols into another sequence of symbols. seq2seq is not only a type of task but also an architecture for computational models that are build to solve seq2seq tasks. Further down, I will introduce this architecture. Machine translation is a very well-known seq2seq task [?]. In machine translation we transform a sequence of words in one language into another sequence of words in another language. In the present case of creating phonetic transcriptions, the input sequence is a sequence of graphemes in any script. The output sequence is a sequence of phonemes[4] written in IPAlpha. This process

[4]Please refer to section 2.3 in order to understand the terminological implications of phoneme. As it is common in research, I will stick to the term *phoneme* although strictly speaking it is not always correct. Phoneme in this case just refers to any symbol that is used to represent a sound.

is typically referred to as grapheme-to-phoneme (G2P) conversion. There are a few problems and characteristics of seq2seq tasks that are important also for G2P conversion:

- The input and output sequences are not always of the same length. It is difficult to align input and output which means to map one or more input grapheme to one or more output phoneme. Not all systems rely on aligning input and output but often it is needed to analyse the results. For example to visualize which grapheme(s) is/are mapped to which phoneme(s).
- Due to the open-vocabulary situation and the impossibility to cover all possible words, all systems must be able to deal with rare and unseen words [??].
- Most of the research done in this area is limited to the English language. This is not uncommon in NLP research. The availability of English data resources and the unavailability and struggles to find data in other languages heavily influences this research.

There exist powerful seq2seq models that can deal with the first point mentioned in the listing above. Some but not all of the models I will present below can deal with the second point mentioned above while the third point does not depend on the model but on the availability of language resources. However, there are computational models that can deal better with only little available data as I will explain.

3.3 Computational models for G2P

In the following, I am going to explain the most important model types. Sometimes, it is possible to use more than one model type in combination to exploit the advantages of each model.

3.3.1 Look-up dictionary

The simplest approach to G2P is a look-up table that allows me to store a grapheme sequence (typically a word) and a phonetic transcription of that word. If I'd like to transcribe a specific word, I can search the table for that word and get its phonetic transcription. Such a look-up-table, which is also called a dictionary, is expensive to create and has a limited coverage as it requires to add each grapheme-phoneme pair manually. Although such a system is no longer useful on its own it can still be used in addition to other, for example, statistical models [?].

3.3.2 Rule-based models

The first systems to create phonetic transcriptions of text were rule-based systems. Although rule-based systems are outperformed by more recent neural models [??], I will introduce them as they were an important step towards G2P modelling. Additionally, rule-based models can be used together with other models to reach a better performance. Rule-based transcriptions models are built using linguistic pronunciation rules. For example, if in a language a certain word initial grapheme is always pronounced the same way, we can store a rule that states that if that grapheme is encountered word initially it should be transcribed using a specific phoneme. While this is a rule on grapheme level, there exist also rules on word or sub-word level [?]. In order to be able to create such a system, one needs to collect or create the pronunciation rules first which needs a lot of linguistic expertise. Another drawback of such systems is that they might fail when presented with unseen or rare words if rules are at the word level [?].

3.3.3 N-gram models / statistical models

N-gram models or statistical models models were used before neural models took over the field. These are sometimes referred to as traditional models. Statistical models need an alignment of graphemes and phonemes. This means that they need to map grapheme(s) to corresponding phoneme(s). This is needed because one grapheme can be realized as multiple phonemes and vice versa. It is not possible to simply have a one-to-one alignment. The necessity of grapheme-phoneme alignment is one reason why statistical models were outperformed by neural models. The main intuition about statistical models is that they, in some way or other, try to statistically model the relationship between graphemes and phonemes. A very common statistical model is the joint-sequence model. Other names for statistical models encountered in the research are finite-state transducers (FSTs) or weighted finite-state transducers (WFSTs).

3.3.4 Neural models

Neural G2P models have been reported to outperform most other models [?]. The most important of neural models will be introduced in the following. As I have pointed out earlier, G2P modelling is a seq2seq task that can be solved by seq2seq models. Neural seq2seq models include an encoder and a decoder or more than one of each depending on the exact implementation. The encoder first processes the entire

input sequence. Once the encoder is done, the output of the encoder is passed to the decoder as input. Then the decoder processes the sequence and outputs the final sequence. Figure 4 shows the basic structure of the encoder-decoder architecture. What makes such a model architecture powerful is that they can map an input sequence to an output sequence of a different length and different type which is exactly what we need for G2P conversion. As encoder and decoder we can use different types of reccurent neural networks (RNNs) which I present below.

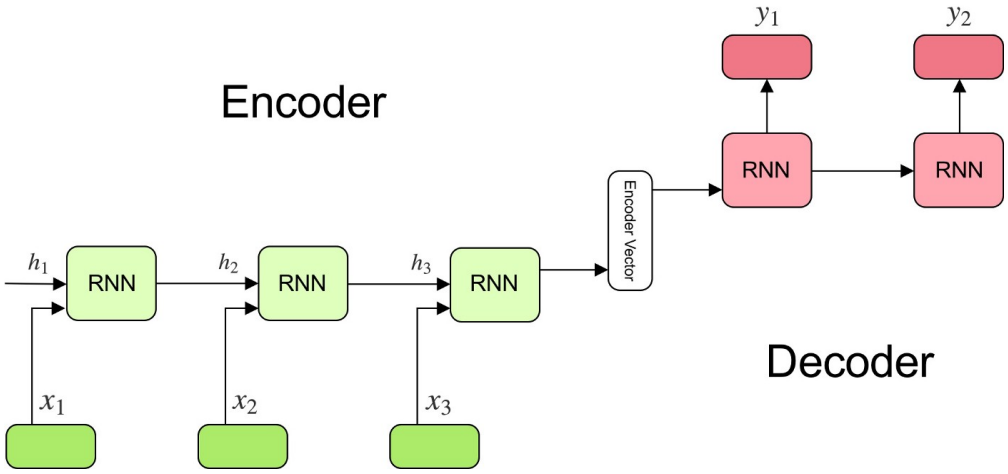

Figure 4: This figure shows how the basic encoder-decoder or sequence-to-sequence architecture looks like. The colored RNN boxes represent the computational model that is used as the encoder and decoder. The input sequence x is processed token by token by the model. The output of the encoder is passed to the decoder which outputs an output sequence y again token by token. Note that the input and the ouput sequence are not of the same length[5].

RNN The important fact about RNNs is that they process the input sequence sequentially. In our case, this means that a RNN performs a specific computation on each grapheme of the input sequence. It can only process the next input grapheme once it is done with the preceding grapheme. Instead of only outputting something for each grapheme, a hidden representation is passed on to be processed together with the next grapheme. This sequential processing is crucial as it means that the model has information about all graphemes preceding the grapheme that is currently processed. The output of a RNN is a sequence of the same length as the input sequence. Each token of the output represents a phoneme. Figure 5 shows a basic RNN model.

A special kind of RNN is the LSTM. LSTM means long short-term memory. As their name suggests they include what we could call a memory. Instead of just including

a representation of the preceding graphemes when processing the next grapheme in the sequence, LSTMs can more flexibly decide what information is added and what information should be forgotten [??]. LSTMs are used a lot for G2P conversion [????]. Neural transducers are also a variation of RNNs. When neural transducers are used as encoder and decoder in a seq2seq architecture they can start with the decoding step before all of the input sequence is processed by the encoder. This allows for more flexibility.

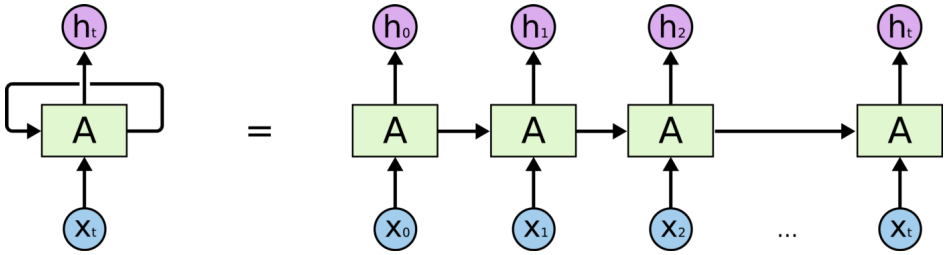

Figure 5: The figure illustrates how a single RNN processes an input sequence. x_0 - x_t denotes the input sequence. Each x_n is one grapheme in our case. We can see that a representation of each segment x_n is passed to the next block which processes the next segment x_{n+1}. h_0 - h_t denotes the output sequence of the RNN. We can see that there is one output segment for each input segment. A represents a computation of the model which could be a simple RNN but also a LSTM[6]

Transformer Transformers are seq2seq models as well. A central concept and what makes transformers powerful models is their attention mechanism. Attention allows transformers to model dependencies between tokens of one sequence. For example: if one grapheme is pronounced differently depending on another grapheme in the sequence, the transformer can model this dependency using the attention mechanism. In addition, it can also model dependencies between the input and the output sequence. This means that while creating the output sequence, the transformer has access to the input sequence. This is possible because transformers do not have to process the input sequence sequentially like RNNs or LSTMs. They can process the entire input sequence in parallel [?].

3.4 Training & evaluation

For the training of (neural) models, it is necessary to follow a specific training regimen. The data that is used to train the model is typically split into three parts:

- **Training set**: The training set is the largest part of the dataset. Typically it is about 80% or more of the entire data. When training a model this is what we pass to the model such that it can learn how to solve the task in question.
- **Development or validation set**: The development or validation set consists of about 5% - 15% of the entire data. It is used to evaluate the model performance *during* the training process. This means that we let the model predict its solutions for the data in the validation set and evaluate it by comparing it to the correct solutions. This procedure gives us an idea of how and if the model improves or not.
- **Test set**: The test set is often of the same size as the development set and contains about 5% - 10% of the data. This set is exclusively used to test the model once it is trained. This is necessary as we want to avoid that the model sees the entire data. Using a test set we can ensure that the model actually abstracted some rules from the training data and did not just learn how to reproduce the training data.

Evaluation metrics

No matter what model is used, we need to appropriately evaluate it. For neural models this evaluation is performed in the test set. This means we let the model produce output phonemes for a set of input grapheme sequences and then evaluate the performance of this output. In the case of G2P conversion we have a reference transcription and a system output that can be compared to the reference and then the difference can be quantified and given a score. The most common metric to evaluate G2P conversion is the word error rate (WER). The lower the score, the better the model. If the WER is 0, this means that the texts are exactly the same. The idea behind the score is that we can capture the cost that it takes to transform the system output into the reference phoneme sequence. Doing this we can find out how different the system output and the reference solution are. The following formula is used to calculate the WER:

$$WER = \frac{S + I + D}{N} \tag{3.6}$$

In equation 3.6 S stands for substitution, I for insertion, D for deletion and N denotes the total number of words in the reference sequence. Those numbers can be calculated by using an algorithm to get the edit distance. It is quite common to multiply the resulting number with 100 [**?**]. Note that the WER can be more

than 100%. This happens if, for example, the are a lot of additional insertions or deletions in the system text. If we would like to evaluate the performance of our model based on individual characters, the score is called character error rate (CER). It is calculated in the exact same way as the WER, but instead of words everything is calculated on character basis. In the phonetic transcriptions setting, the CER is typically replaced by the phone error rate (PER) to match the correct terminology. The calculations are not changed. In a multilingual setting, the same model is sometimes used for different language. In such cases it is custom to use a macro-averaged WER or CER. This means that the sum of the scores for each language is divided by the number of languages [**?**].

In the case of G2P conversion, the WER actually just reflects the rate of wrongly predicted words, as one sequence consists of only one word. It is therefore just 1 (or 100) minus the accuracy. The accuracy is the rate of the correctly predicted words (which means we divide all correctly predicted words by the total number of words). It is important to note, that the WER and the PER suggest slightly different things. If each word has exactly one wrong character this means that the WER would be 100 as all words are predicted wrongly. Now, if the words are really long, the PER score would be very good compared to the really bad WER. If the words are rather short, the PER would get worse as well but the WER would just stay the same. So, when analysing the results it is important to pay close attention to how those two metrics relate.

3.5 State-of-the-art G2P models

I will now have a look at the state-of-the-art models that have been used for G2P conversion. The Special Interest Group on Computational Morphology and Phonology (SIGMORPHON) [**?**] regularly organizes shared tasks concerned with morphology and phonology. For the years 2020 and 2021 they organized a G2P conversion task [**??**]. The tasks represent a first attempt at creating benchmarks for multilingual G2P conversion. Although there is other research on G2P, many recent publications have been made within the SIGMORPHON shared tasks. In the next sections, I will summarize the results and insights from G2P research with a focus on those shared tasks. As they covered so many different languages, I will use their results to compare my results to. Tables 3 and 4 list the models and their results from those two tasks.

3.5.1 Model architectures

In section 3.2, I explained the most important theoretical basics. Now, I familiarized myself a bit more with strategies that work well in practice and what some concrete problems are. The methodologies mentioned below are for the most part task-agnostic. This means that they often improve results on most NLP tasks and are not specifically developed for the G2P task.

Ensembles Many models that are used and achieve peek performance for G2P modelling are ensemble models. An ensemble is essentially just a pool of different models that are trained on the data with different settings or they are completely different models. The way such a model can be used for inference is that all of the models process the input and present their predicted results. Out of all possibilities, one prediction will be chosen that is then the final model output. In order to get the final output, an ensemble needs some kind of decision algorithm to output the best result. A disadvantage of ensembles is that the models need a lot of storage. Also, it is to some extent a bit of a *brute-force* approach as it could lead to preferring quantity over quality.

Learning edit actions Instead of learning to output a phoneme sequence, a model can also learn how to edit the input sequence in order to get the output sequence. Such a model would then output an edit sequence which can be applied to the input sequence in order to obtain the final phoneme output. The model therefore learns to create sequences of edit actions. The edit actions are typically 'insert', 'substitute' and 'delete'. The problem with this approach is that there are many possible sequences of edit actions that produce the same result. For example, it is always correct to delete every unit in the input sequence and then insert every unit from the output sequence. But this does not tell us anything about how graphemes and phonemes relate. To this end, we would also want to use substitution actions to see whether one grapheme is always substituted by the same phoneme. Imitation learning is proposed as a solution for this problem which is a type of reinforcement learning. Easily put, the idea is that the model learns to imitate the behaviour of an expert (for example, a human expert that provides correct samples of the task in question) [**?**].

Multi-task learning What has worked well for G2P models is to use multi-task learning. This means that the model is not only trained on one task but on multiple

tasks that are related. In the present case, a model was trained on phoneme-to-grapheme conversion as well [?].

Neural models Not surprisingly, models that achieve peek performance are almost exclusively neural models [?]. Due to their ability to process increasingly longer sequences and the above discussed techniques like attention, they are ideal for almost all NLP tasks. What type of neural model is chosen also depends on the amount of data available. Transformers are suggested to work better for larger datasets, while they are outperformed by LSTMs on medium-size datasets (a few thousand training pairs) [?].

3.5.2 Data manipulation

A model is only as good as the data that is used to train it. While this is a very basic paradigm, in reality assuring data quality is not always easy. In this section I list a few strategies that are used to preprocess and prepare G2P data and how to deal with too little available data.

Data quality Authors mostly include a section about their preprocessing and what should be done to ensure high quality datasets. The list given below is an incomplete list of potential problems and measures taken in different settings for G2P data [?]:

- **Exclusion of words with less than two Unicode characters or less than two phone segments**
- **Separation by script**: There is no obvious connection between the different scripts of a language and its pronunciation. It makes sense to treat different scripts as different languages.
- **Exclude foreign words with foreign pronunciations**: Foreign words in a language with their original pronunciation can add phonemes that are not in that language's phoneme inventory. If they were to be included it would make sense to include a pronunciation adapted to the actual language.
- **Words with multiple pronunciations in word lists**: While it is possible to exclude duplicates it might also be possible to add part-of-speech (PoS) tags or other linguistic information to distinguish at least some of these words.
- **Consistent broad transcriptions**: With broad transcriptions it is impor-

tant to be consistent and not use allophones. ? did this specifically for Bulgarian. They identified allophones of a language and replaced them by their respective phoneme.

- **Linguistic variation and processes**: Some transcriptions include examples for monophthongization or deletion which are ongoing linguistic processes but should not be part of a dataset representing a standard variation. Monophthongization just means that diphthongs are replaced by monophthongs which are just single vowels. ? dealt with monophthongization by choosing the longer of two transcriptions as this logically exclude the monophthonged version. This does of course only work if there is more than one pronunciation available. The idea behind this type of preprocessing is that G2P modelling should focus on current standard variations.
- **Tie bars**: Some languages (English and Bulgarian) have inconsistent use of tie bars. This can be corrected by replacing all inconsistencies by the tie-bar-version. It is also possible to exclude the tie bars at all.
- **Errors in the transcriptions**: ? noticed many errors in the WikiPron English data. They identified errors by looking at the least frequent phonemes and then check the word-pronunciation pairs where those phones occurred in. As the phoneme inventory of a language is often known, it can be used to check the phonemes in the datasets and identify uncommon ones.

Especially the task of finding errors in the transcriptions is quite tricky. It requires a lot of knowledge about the phonology and phonetics of a specific language.

Low-resource setting Apart from a few well-studied examples, for most languages there is only little data available. It is therefore highly interesting and important to find solutions of how to deal with lack of data. ? presented a system focusing on data augmentation methods. The primary goal of their approach was to test how successful a minimalist data augmentation model would be, knowing it would most probably not outperform any of the other models. They identified two approaches that might improve low-resource models. The first one is to use as much as possible of the development set for training. The second is to train all languages together differentiating the languages only by a tag added to the word representations. The results for these strategies were not every clear as both of these strategies were successful for some languages but did not improve the results for others.

? propose a data augmentation model for low-resource settings. The methodology applied in their approach is ensemble learning combined with a self-learning strategy.

They use their ensemble to make predictions on unlabelled data. This newly created data is then added to the training data and the models are trained for another epoch on the extended data. This strategy worked well and produced good results.

Results in a low-resource setting are still bad when only using 800 samples for training. More research needs to be done in data augmentation techniques and improving the systems to cope with only little available data.

Reduce vocabulary size Some syllabary languages like Korean allow the decomposition into smaller units that make up the signs. Many other languages that do not use the Latin alphabet allow to be written with Latin letters. If a reduction of the character vocabulary size is possible in one of these ways, it almost always improves performance as smaller vocabulary sizes are easier to handle for models [?].

3.5.3 Error and result analysis

In this section I will list different types of analyses that have been performed by different authors on a trained model to improve future research.

Broad and narrow transcriptions In the SIGMORPHON tasks, there are great differences in the performance of models for different languages. One possible explanation is that the datasets were a mix between broad and narrow transcriptions. As narrow transcriptions are a lot more detailed, it can be argued that this is more difficult for any system [?]. This assumption still needs to be analysed more closely. This differing performance for various languages calls for the question what makes a language hard to pronounce. Especially as for Georgian all models from the SIGMORPHON task reached a WER of 0.0. For Georgian the provided training set only contained 10,000 samples. Interestingly enough, the WER for English which was trained on 42,000 samples reached one of the highest WERs. This suggests that English is more irregular to pronounce than Georgian. That a model trained on considerably less data for one language compared to a model trained on much more data for another language performs a lot better is a strong indication that one language is easier to pronounce.

Linguistic error analysis ? chose to perform an error analysis and try to minimize the frequent errors of a model in a multilingual low-resource setting. The analysis showed that often the model gets vowels and diacritics wrong. They extended

the model in such a way that wrong vowel and diacritics predictions are punished more than other errors. Compared with the unchanged model, this extended model reached a better performance for some languages. The predictions with their model shows an improvement in vowel prediction. A further analysis showed that many errors still happen with vowels. Vowels get often confused with similar vowels. Their conclusion is that many of these errors make sense in a linguistic sense.

Another type of linguistic analysis that can be performed is to analyse the data and check for uncommon pronunciations or language internal ambiguities. If a model produces a lot of wrong output because of ambiguities or uncommon data, then this is not necessarily the model's fault but just a language inherent inconsistency. As languages are generally ambiguous, this type of analysis is very insightful to find out about *real* errors of the model. If a model makes a lot of mistakes although there is no underlying language inherent ambiguity this is a real mistake of the model. It should be possible for a model to abstract a transcription rule from the data if there are no ambiguities. **?** did such an analysis for the SIGMORPHON 2021 task which showed that many errors are due to language internal ambiguities.

Include linguistic information What has been suggested by **?**, is to make use of phonetic resources or rule-based systems to improve the quality of current models. The advantage of such an approach is that it is specifically tailored to the problem at hand and not at all task-agnostic. **?** confirm this suggestion as they performed an error analysis which showed that including linguistic information such as PoS-tags might be useful.

As is always the case with such research there are many different aspects that can be tuned in order to improve model results. For my thesis I will have a closer look at how we can use phonetic features to improve models.

ISO396-3	**BS20**						**DeepSPIN20**		**IMS20**		**BS21**	**CL21**	**UBC21**		**DP21**
	LSTM		**transformer**		**pair n-gram**							CL	UBC-1	UBC-2	
	WER	PER	WER	PER	WER	PER	WER	PER	WER	PER	WER	WER	WER	WER	
ady^{B}	28.00	*6.53*	28.44	*6.49*	32.00	*7.56*	24.67[3]		25.00	*5.79*	**22.00**	**22.00**[23]	25.00	**22.00**	
bul^{A}	31.11	*5.94*	34.00	*7.89*	41.33	*9.05*	-		22.22	*4.85*	**18.30**	18.80[6]			
cym (wel_sw)^{B}											**10.00**	**10.00**[1]	13.00	12.00	
ell (gre)^{B}	18.89	*3.30*	18.89	*3.06*	21.78	*4.05*	-		**18.67**	*2.97*	21.00	20.00[13]	22.00	22.00	
eng(_us)											41.94				**37.43**
fra (fre)^{A}	6.22	*1.32*	6.89	*1.72*	13.56	*3.12*	**5.11**[3]		6.89	*1.60*	8.50	7.50[456]			
hbs^{A}											**32.10**	35.3[7]			
hin	6.67	*1.47*	9.56	*2.40*	12.67	*4.05*	-		**5.11**	*1.20*					
hun^{A}	5.33	*1.18*	5.33	*1.28*	6.67	*1.51*	-		5.11	*1.12*	1.80	**1.00**[67]			
hye (arm_e)^{A}	14.67	*3.49*	14.22	*3.29*	18.00	*3.90*	-		12.67	*2.94*	7.00	**6.40**[7]			
ice^{B}	10.00	*2.36*	10.22	*2.21*	17.56	*3.62*	-		**9.33**	*2.04*	12.00	10.00[13]	13.00	11.00	
ita^{B}											**19.00**	31.00[3]	20.00	22.00	
jpn(_hira)^{A}	7.56	*1.79*	7.33	*1.86*	9.56	*2.07*	**4.89**[4]		5.33	*1.26*	5.20	5.00[7]			
kat (geo)^{A}	26.44	*5.14*	28.00	*5.43*	37.78	*6.48*	-		24.89	*4.57*	**0.00**	**0.00**[4567]			
khm^{B}											34.00	32.00[13]	31.00	**28.00**	
kor^{A}	46.89	*16.78*	43.78	*17.50*	52.22	*15.88*	24.00[13]		26.22	*4.38*	16.30	**16.20**[4]			
lav^{B}											55.00	**49.00**[23]	58.00	**49.00**	
lit	**19.11**	*3.55*	20.67	*3.65*	23.11	*4.43*	-		20.00	*3.63*					
mlt(_ltn)^{B}											19.00	12.00[1]	19.00	18.00	
nld (dut)^{A}	16.44	*2.94*	15.78	*2.89*	23.78	*3.97*	-		**13.56**	*2.36*	14.70	14.70[7]			
rum^{B}	10.67	*2.53*	12.00	*3.62*	11.56	*3.55*	**9.78**[3]		10.22	*2.23*	10.00	12.00[3]	14.00	10.00	
slv^{B}											49.00	50.00[1]	56.00	**47.00**	
vie^{A}	4.67	*1.52*	7.56	*2.27*	8.44	*1.79*	**0.89**[2]		1.56	*0.48*	2.50	2.00[57]			

Table 3: The table lists the SOTA models from the SIGMORPHON tasks in 2020 and 2021. Superscript: model numbers. Numbers in bold: best WER. Languages in bold are in the 100LC corpus. ^{A}: 10,000 training samples in 2021. ^{B}: 800 training samples in 2021. Model explanations: table 4.

Model name	Authors	Model Architecture
CL21	SIG21: ?	These are seven models in total. LSTM-based neural transducer with pointer network-like monotonic hard attention trained with imitation learning. All models 1-7 are majority-vote ensembles with different number of models (5-30) and different inputs (characters or segments).
UBC21	SIG21: ?	UBC-2 outperforms the baseline. They analysed the errors of the baseline and extend it by adding penalties for wrong vowels and wrong diacritics. Errors on vowels actually decreased. UBC-2 achieved the best macro average for low-resource languages in 2021. UBC-1 included syllable prediction which did not improve the results.
DP21	SIG21: ?	Dialpad-1: Majority-vote ensemble consisting of three different public models (weighted FST, joint-sequence model trained with EM and a neural seq2seq), two seq2seq variants (LSTM and transformer) and two baseline variations.
BS21	?	The baseline in 2021 is a neural transducer trained with imitation learning similar to a model submitted in 2020 [?]. As they are so similar, I did not include the model twice, but only the newer one.
DeepSPIN20	SIG20: ?	DeepSPIN-2,-3,-4: Transformer- or LSTM-based seq2seq models with sparse attention. Add language embedding to encoder or decoder states instead of language token.
IMS20	SIG20: ?	IMS: Self training ensemble of one n-gram-based FST and three seq2seq (vanilla with attention, hard monotonic attention with pointer, hybrid of hard monotonic attention and tagging model).
BS20	?	The baseline for the task in 2020 consisted of three different model types. An LSTM, an transformer and a pair-n-gram model that is based on a weighted FST. All of them were outperformed by other models except for Lithuanian. The LSTM performed best among these three.

Table 4: This table presents the state-of-the-art G2P models which will be important to compare my own results to. The results for these models can be found in table 3

4 Experiments: Data Collection and Preprocessing

The first practical part of this thesis is concerned with data collection. Although phonetics is an important subarea in linguistics, phonetic transcriptions in the form of continuous text are hard to find. If there are any transcriptions available, it is not always possible to use them as-is. In the following, I outline the different data types which are available and the different strategies that are used to convert the data into one well-formatted corpus. All the data that I collected and used for this thesis can be found on my public GitHub repository[1].

There is quite a famous set of phonetic texts which is a collection of short stories called *The North Wind and the Sun*. Those stories were the starting point for my search for data. The reason for this is that the short stories are available in many different languages. As I aim at creating a multilingual phonetic corpus, the availability of data in many languages was one of the key criteria to choose the datasets. Not all languages are relevant for my thesis. In section 4.2, I will present the languages that are relevant for my thesis and why those languages are relevant.

One of the first things I have noticed while collecting my datasets is the abundance of different codes and names to refer to the languages. I have decided to use the ISO 396-3[2] convention, not to be confused with the ISO 396-2 convention which uses sometimes completely different codes and does not cover all of my languages. ISO 396-3 distinguishes some more dialects and variants. This also means that throughout my research I paid close attention to what language variant was actually referred to. For some cases, it seems that there exists a dataset for a language which is relevant for my thesis, but actually it is not the same variant that I am looking for. To make sure I pick the same code and language, I used the Glottolog database as reference[3].

[1] `https://github.com/theDebbister/masterThesis`

[2] `https://iso639-3.sil.org/code_tables/639/data`

[3] `https://glottolog.org/glottolog`

4.1 Transcription sources & formats

Phonetic transcriptions of various languages are available from different sources in different formats. From what I have found out in my research phonetic transcriptions are available as either full texts or word lists.

Full Text For the task of G2P conversion, phonetic transcriptions in the form of fully transcribed texts would be ideal. As became clear, it is hardly possible to find full text phonetic transcriptions. There is plenty of material describing how different languages can be transcribed but those rarely contain fully transcribed text. If they do, it is mostly limited to one or a few sentences. The only fully transcribed texts are *The North Wind and the Sun* (NWS) short stories which are described in section 4.3.2.

Pronunciation Dictionaries Another data type that is found quite often are word lists. Those are sometimes referred to as pronunciation dictionaries. However, pronunciation dictionary often means that there are words mapped to an audio representation of that word which is not what is meant in this present case. Pronunciation dictionary in this present case refers to the mapping of an orthographic word to its pronunciation using phonetic symbols. Although such lists are very handy, especially as they can easily be used to train a transcription model, transcriptions of individual words and of entire texts are not exactly the same. There are two major problems:

- Pronunciation depends on the context of the word in question. Word forms are ambiguous and sometimes their pronunciation differs given on their specific context.
- Phonetic boundaries are not always equivalent with word boundaries. Spoken language sometimes merges certain words which leads to one phonetic unit.

Data in the form of word lists can be found quite a lot when compared to the availability of full text. An example is the Nidaba[4] website that presents a few word lists in different languages. What becomes clear when looking at this data is that there are two important limitations: First, there are only very few languages available in the Nidaba corpus that are relevant for my thesis. Second, there are different transcription conventions used.

[4] `http://nidaba.co.uk/Contents/OriginalWordList`

Transcription conventions No matter if the transcriptions are available as full text or word list, a key factor for me to choose a particular dataset is the transcription convention. The IPAlpha is a well-known convention but there is a lot of data that is available using different conventions. Although it might be possible to convert one convention into another convention, this is very tedious and often those conventions are made specifically for one language or a group of related languages and are not cross-linguistically applicable. This is why for this present thesis, I decided to only look at datasets that make use of the IPAlpha transcription convention.

4.2 The 100 language corpus

The basis of the data used in this thesis is a corpus provided by the SPUR lab at the University of Zurich (UZH). The text group of the Language and Space lab at the University of Zurich maintains a project that provides a multilingual corpus consisting of 100 language text samples [?] which is referred to as 100LC. Those 100 languages are meant to be representative for all the world's languages. It is therefore meant to give insight on relations, similarities, differences or properties of individual languages or language families. Specifically, the goal of the team at UZH is to use quantitative methods like statistical modelling, machine learning and information theory to study language variation and compare languages. The corpus contains 100 languages which are proposed by **?**. This is an online book that contains different chapters each of which shows a different linguistic feature including a map which shows the distribution of that feature over the world's languages. While the number of languages presented on the individual maps depends on the amount of research done in a specific area, the sum of all maps gives quite an impressive overview on the structure of nearly half of the world's languages. Out of the 2676 languages a sample of 100 languages was proposed. This sample does not contain too many languages from one area, neither does it contain too many languages from one family. Those are the 100 languages that are in the 100LC. Not considering the aforementioned criteria of maximizing genealogical and areal diversity can lead to misleading results in multilingual analysis. Figure 6 shows the distribution of the languages in the corpus on a world map. The different icons show the genus of the languages which is a classification of languages defined by the World Atlas of Language Structures (WALS) that maintains the language description collection. The interactive map can be viewed online [?]. Table 21 in the appendix A shows all languages that are in the 100 language corpus. None of the text samples are provided by WALS. The entire corpus is provided by the SPUR team at UZH that

Iso 639-3	Language name	Type WikiPron	# Words WikiPron	Type NWS
cmn	Chinese	broad	133,686	unk
deu	German	broad	34,145	broad
deu	German	narrow	10,984	narrow
ell	Greek (Modern)	broad	10,547	unk
eng	English US	broad	57,230	broad
eng	English US	narrow	1,633	narrow
eng	English UK	broad	60,422	-
eng	English UK	narrow	1,284	-
eus	Basque (Goizueta)	broad	1,742	broad
fin	Finnish	broad	69,015	-
fin	Finnish	narrow	69,008	-
fra	French	broad	56,911	unk
hin	Hindi	narrow	9,563	-
hin	Hindi	broad	10,812	unk
ind	Indonesian	broad	1,555	unk
ind	Indonesian	narrow	2,637	-
jpn	Japanese (Hiragana)	narrow	19,689	-
kat	Georgian	broad	15,123	broad
kor	Korean	narrow	14,141	unk
mya	Burmese	broad	4,631	broad
rus	Russian	narrow	402,586	unk
spa	Spanish (Castilian)	broad	60,677	broad
spa	Spanish (Castilian)	narrow	52,190	narrow
spa	Spanish (Latin America)	broad	48,649	-
spa	Spanish (Latin America)	narrow	41,845	-
tgl	Tagalog	broad	3,321	-
tgl	Tagalog	narrow	1,915	-
tha	Thai	broad	15,050	unk
tur	Turkish	broad	1,789	unk
tur	Turkish	narrow	1,812	-
vie	Vietnamese	narrow	15,240	unk
zul	Zulu	broad	1,677	-

Table 5: These are all languages I will use for my experiments as they are in the 100LC (section 4.2) and they are available as a WikiPron word list (section 4.3.1). For some of those languages there is an NWS short story available as well. If not, the 'Type NWS' column does not have an entry.

collected the corpus over the last few years and is continuously working on and with it.

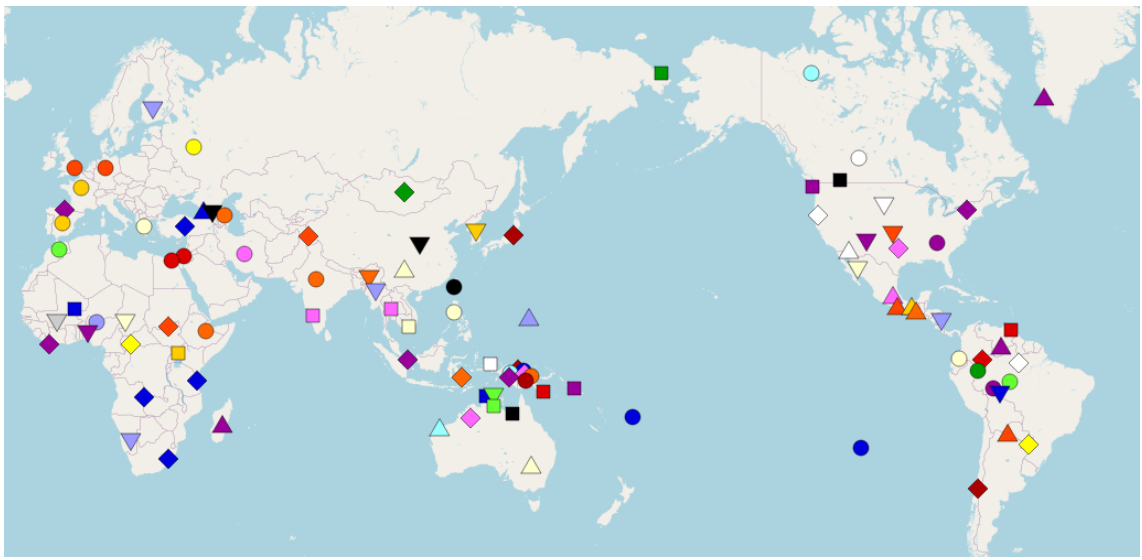

Figure 6: WALS - Map that shows the 100 Languages

4.3 Data used for this thesis

Based on the outcome of my research on the availability of phonetic data, I decided to use two datasets for my experiments in this thesis. I will introduce both of them below.

4.3.1 WikiPron

A very recent project that publishes pronunciation lists is WikiPron. The WikiPron project [?] is an open-source Python mining tool to retrieve pronunciation data from Wiktionary. Their database contains 1.7 million word/pronunciation pairs in 165 languages. Both, the database and the mining tool, are freely available online. The WikiPron data for one language is always structured the same. It is a tsv-file that has graphemes as a first column and corresponding phonemes as the second column. While the graphemes are just listed as-is, the phonemes are split using the segments library, which I shortly presented in section 3.1. The phoneme segments are separated by white spaces. In table 6, I show an example of how the WikiPron data looks like.

The WikiPron data has already been used in shared tasks which means that results

on this data can easily be compared to other results. For a shared task in 2021 organized by SIGMORPHON, the WikiPron data was improved and additional scripts were added based on feedback and findings from a similar task in 2020. One major improvement was concerned with languages written in different scripts. WikiPron supports now the detection of different scripts and languages can be sorted according to those scripts. For some languages, there is a filtered version of either the broad or the narrow transcription available. Whenever WikiPron made a filtered word list available for one of my languages, I used this filtered version as a starting point.

Grapheme	**Phoneme**
a	ʔ aː
aa	aː
aachen	aː x ə n
aachener	aː x ə n ɐ
aachenerin	aː x ə n ə ʁ ɪ n
...	...
übungen	ʔ yː b ʊ ŋ ə n
übungsbuch	yː b ʊ ŋ s b uː x
üppig	ʏ p ɪ ç
üsselig	ʏ z ə l ɪ ç
œuvre	œː v ʁ ə

Table 6: This table shows an extract from the German broad WikiPron word list. Instead of using two columns like in this example, the grapheme and the phoneme sequence are separated by a 'tab' character in the original file.

4.3.2 *The North Wind and the Sun*

As I have already mentioned, the NWS short stories are quite a well known corpus of phonetic transcriptions. The Journal of the International Phonetic Association (JIPA) continuously published different phonetic transcriptions of this short story. The story is a fable said to be written by Aesop and has been translated into many languages. Additionally, for many languages there exits a phonetic transcription. Many of these texts are available already transcribed in a text file format and free to use[5]. Table 7 shows the languages for which the short story is available and which are also in the 100LC. **?** performed an analysis on these texts to find out how many phonetic tokens we need to cover a phonetic inventory of a language adequately. What they have found out in their study is that those short stories are

[5] `https://github.com/SimonGreenhill/jipa`

by far not long enough to give a good picture of the phonetics of a language. This is of course problematic if those texts are used to demonstrate how a language works phonetically. That said, I will use these stories with care and only as an additional dataset to have more variety.

Iso 639-3	Type	Variation	Language
arn	broad and narrow		Mapudungun
cmn		Pekinese	Mandarin Chinese
deu	broad and narrow	North German	German
ell			Modern Greek
eng	broad and narrow	Americsn	English
eus	broad and narrow	Goizueta	Basque
hau	narrow		Hausa
heb			Modern Hebrew
hin	narrow		Hindi
ind			Indonesian
kat	broad and narrow		Georgian
kor			Korean
mya			Burmese
pes			Western Farsi
spa	broad and narrow	Castilian	Spanish
tha			Thai
tur	broad	Istanbul	Turkish

Table 7: The table shows a list of all the short stories *The North Wind and the Sun* that are available as phonetic text and whose languages are in the 100 language corpus (see section 4.2)

Below, I include the English version of the short story including its phonetic transcription (examples 4.1 and 4.2).

(4.1) The North Wind and the Sun were disputing which was the stronger, when a traveler came along wrapped in a warm cloak. They agreed that the one who first succeeded in making the traveler take his cloak off should be considered stronger than the other. Then the North Wind blew as hard as he could, but the more he blew the more closely did the traveler fold his cloak around him; and at last the North Wind gave up the attempt. Then the Sun shined out warmly, and immediately the traveler took off his cloak. And so the North Wind was obliged to confess that the Sun was the stronger of the two.

(4.2) ðə ˈnoɹθ ˌwɪnd ən (ð)ə ˈsʌn wəˑ dɪsˈpjutɪŋ ˈwɪtʃ wəz ðə ˈstɹɑŋgəˑ, wɛn ə

ˈrɹævələˇ ˌkəm əˈlɑŋ ˈɹæpt ɪn ə ˈwoɹm ˈklok. ðe əˈgɹid ðət ðə ˈwʌn hu ˈfəˇst səkˈsidəd ɪn ˈmekɪŋ ðə ˈtɹævələˇ ˈtek ɪz ˈklok ˌɑf ʃʊd bi kənˈsɪdəˇd ˈstɹʕŋgəˇ ðən ðɪ ˈəðəˇ. ðɛnðə ˈnoɹθ ˌwɪnd ˈblu əz ˈhɑɹd əz i ˈkʊd, bət ðə ˈmoɹ hi ˈblu ðə ˈmoɹ ˈklosli dɪd ðə ˈ tɹævləˇ ˈfold hɪz ˈklok əˈɹaʊnd ɪmˈ ˌæn ət ˈlæst ðə ˈnoɹθ ˌwɪnd ˌ gev ˈʌp ði əˈtɛmpt. ˈðɛn ðə ˈsʌn ˈʃaɪnd ˌaʊt ˈwoɹmli ənd ɪˈmidiətli ðə tɹævləˇ ˈtʊk ˌɑf ɪz ˈklok. ən ˈso ðə ˈnoɹ ˌwɪnd wəz əˈblaɪʒ tɪ kənˈfɛs ðət ðə ˈsʌn wəz ðə ˈstɹɑŋgəˇ əv ðə ˈtu.

Transcription of NWS stories

A collection of the NWS stories is available in a handbook of the JIPA which is only available as a pdf scan of the original book [**?**]. Luckily, most of those texts have been transcribed and made available by Simon Greenhill[6]. At least the phonetic part of it has been transcribed, the original version was still not available already transcribed. Before I knew about the availability of these texts in text file format, I did some research on optical character recognition (OCR) for IPAlpha texts and manual transcription. While OCR is technically possible it turns out to be very difficult for IPAlpha characters. There are tools that include IPAlpha character recognition like the ABBYY FineReader which can be acquired for a fee. The Computational Linguistics institute at the UZH owns a version of the ABBYY tool but this version does not include the IPA module. I ran the ABBYY version without the IPAlpha module on a JIPA pdf containing said phonetic transcriptions but the result could not be used. Mostly diacritics and special phonetic symbols were not correctly transcribed. Diacritics and special phonetic symbols are exactly those graphemes that make it difficult to transcribe IPAlpha manually, so there is not point in using this tool. There are also open source tools, one of which is called tesseract[7]. tesseract does not include the IPA alphabet. It is possible to train the model to include the IPA alphabet but this would need appropriate training data which I do not have.

As it was not possible to use OCR to transcribe the texts, a next approach is to manually transcribe them. I experimented with a software called Transkribus[8] to manually transcribe the pdf scans. The software allows to make use of neural handwritten text recognition (HTR) models. There exists no pre-trained model for transcribing IPAlpha charachters, but I trained my own while transcribing some of the documents. On the website they mention that, ideally, training needs 5,000 -

[6] https://github.com/SimonGreenhill/jipa

[7] https://github.com/tesseract-ocr/tesstrain

[8] https://readcoop.eu/de/transkribus/

10,000 words already transcribed. Although my available data is not nearly enough to train a reliable model (the short stories have around 40 - 100 tokens), it was a great help to transcribe. As the scans where not handwritten but machine typed text, the model still reached a surprisingly good quality. As an example: For the Hebrew transcription, the model reached a WER of 34.52 and a CER of 6.11. The two main mistakes were made for two characters that were not even in the training data. The quality of the scans differed quite a lot which had an influence on the performance of the model as well. After transcribing more documents I trained the model again on the newly transcribed text and let it automatically transcribed a few more documents. The transcriptions got continuously better such that in the end I did not take me nearly as much time to correct the model's automatic transcriptions as it got most of the IPAlpha characters correct. Most of the errors resulted from characters that had not been in the previously transcribed documents.

It was interesting to see that it is relatively easy to transcribe IPAlpha text when using the right software. The orthographic texts were easier to transcribe as they do not contain as many diacritics or other special characters. Still, for some texts like Chinese or Hindi, I asked different people for help that know the respective language.

4.4 PHOIBLE

A way to analyse phonetic corpora is to use phonetic features to represent each phoneme. Phonetic features are a list of properties that are overlapping with the phonetic description of each phoneme, that I have mentioned before when writing about phonetics in general in chapter 2. Phonetic features are a minimal list that can be used to describe unique phonemes. There exists an online database called PHOIBLE [**?**] that contains over 3,000 phonemes for more than 2,000 languages. PHOIBLE includes a feature system that can describe each phoneme uniquely. In total, there are 37 phonetic features that are used to describe the phonemes. Each of the features can take on three values:

- '+' (applies to this phoneme)
- '-' (does not apply to this phoneme)
- '0' (not applicable)

When we give each feature a value for each phoneme, this gives us what we can call a feature vector for each phoneme. In addition to the phonetic features, each

phoneme has other features like for example what allophones are used for it in a specific language. The entire PHOIBLE inventory is structured around languages. It lists all phonemes including features for all languages that are in the database. This means that one phoneme can be listed for more than one language, but the features will always be the same for the same phoneme no matter what language. Also, this means that we cannot only use PHOIBLE to represent each phoneme as a feature vector, but we can also get an overview of the phoneme inventory of each language that is covered in PHOIBLE. Table 8 shows an example for two feature vectors for two phonemes.

All the PHOIBLE phonetic features are listed in table 8 together with an example. Some of them might sound familiar from the linguistic background chapter. This is because the feature system was designed based on linguistic descriptions of sounds and is supposed to be cross-linguistically applicable[9] just as the IPAlpha. In order to understand better how those features work, I will explain them in a bit more detail. The features can be grouped into subsets of features. The features in one of the subsets are related to some extent. Some of the features and subgroups of features are more straightforward to understand, while others need more linguistic background. Although the PHOIBLE features are similar to the description of phonemes based on the IPAlpha, there are sometimes features that seem very different or incomplete. This is because on the one hand, the PHOIBLE features are designed to be minimal. This means that sometimes two or more features are used in combination to obtain an additional feature. Please see the description of '**STRESS, LONG, SHORT**' below where I give an example:

STRESS, LONG, SHORT: These three features are very straightforward to understand and roughly correspond to the suprasegmental chart in the IPAlpha figure 1. The special thing about the features 'long' and 'short' is that we can combine them to obtain an additional feature 'half-long' (which is also found in figure 1). In order to do that we simply mark 'long' as '+' and 'short' as '+' as well. Although at the first sight this combination does not make sense as something cannot be long and short at the same time, it is a smart way to minimize the numbers of necessary features. Many other PHOIBLE features are used in a similar way to be able to represent more different phonemes by combining already existing features.

TAP, TRILL, LATERAL, LABIODENTAL, NASAL, CONSONANTAL, LABIAL, APPROXIMANT, DORSAL, CORONAL, ANTERIOR: All of these features can be inferred from the IPAlpha consonant table (see figure 1). The PHOIBLE feature 'labial'

[9] https://github.com/phoible/dev/tree/master/raw-data/FEATURES

Phoneme	**tone**	**stress**	**syllabic**	**short**	**long**	**consonantal**
h	0	-	-	-	-	-
j	0	-	-	-	-	-

Phoneme	**sonorant**	**continuant**	**delayed Release**	**approximant**	**tap**	**trill**
h	-	+	+	-	-	-
j	+	+	0	+	-	-

Phoneme	**nasal**	**lateral**	**labial**	**round**	**labiodental**	**coronal**
h	-	-	-	0	0	-
j	-	-	-	0	0	-

Phoneme	**anterior**	**distributed**	**strident**	**dorsal**	**high**	**low**
h	0	0	0	-	0	0
j	0	0	0	+	+	-

Phoneme	**front**	**back**	**tense**	**retracted Tongue Root**	**advanced Tongue Root**	**periodic Glottal Source**
h	0	0	0	0	0	-
j	+	-	+	0	0	+

Phoneme	**epilaryngeal Source**	**spread Glottis**	**constricted Glottis**	**fortis**	**raised Larynx Ejective**	**lowered Larynx Implosive**
h	-	+	-	-	-	-
j	-	-	-	-	-	-

Phoneme	**click**
h	-
j	-

Table 8: This table shows two PHOIBLE phonetic feature vectors for the phonemes /h/ and /j/. Note that all the features would be on one line. I split them into multiple lines to show them more easily. For reasons of convenience I split some of the feature names with a white space (for example: 'delayedRelease' → 'delayed Release'). In the original PHOIBLE database, all feature names are written as one string.

is called 'bilabial' in the IPAlpha consonant table. For example: The consonant /ʙ/ has features 'trill' and 'labial' marked as '+' as we can infer from the IPAlpha table. The 'dorsal' feature summarizes palatal, velar and uvular. The same is true for the 'coronal' and 'anterior' features which each represent a different subset of consonants.

ROUND, HIGH, LOW, FRONT, BACK: Those five features are inspired by the vowel schema described in section 2.1. We can use the combination of these to describe the different vowel tongue positions. For consonants, these features are typically not applicable.

SYLLABIC, RETRACTEDTONGUEROOT, ADVANCEDTONGUEROOT, RAISEDLARYNXEJECTIVE, LOWEREDLARYNXIMPLOSIVE: In the IPAlpha we use a diacritic mark to represent each of these features. If a phoneme has a specific diacritic mark, we mark the respective PHOIBLE feature as '+'. Note that in the IPAlpha table, some features are named a bit differently (for example, 'advanced' instead of 'advancedTongueRoot').

TONE, CLICK: For tones and clicks there exists a separate table or column in figure 1. Whenever a phoneme can be considered a click or a tone, we mark the respective feature as '+'.

CONTINUANT, SONORANT: These two features are not connected directly to the IPAlpha. They are two broad phonetic categories that are used to describe a specific manner of articulation. Both are applicable to different consonants as well as vowels.

SPREADGLOTTIS, CONSTRICTEDGLOTTIS, DELAYEDRELEASE, STRIDENT, DISTRIBUTED, TENSE, FORTIS, PERIODICGLOTTALSOURCE, EPILARYNGEALSOURCE: These are features that have no direct connection to any of the linguistic phenomena or categories that I have introduced. But just as for all the other PHOIBLE features, they are well based on linguistic knowledge and how the sounds are produced.

PHOIBLE is curated very carefully and contains many different phonemes for each language. As it is based on many different studies and other phonetic databases it is very complete. This means that if there is a phoneme used in one of my datasets that cannot be found in PHOIBLE, chances are very high that it might be a mistake in my dataset. Although it is of course also possible that there is a phoneme missing in PHOIBLE. Still, comparing phonetic data to the PHOIBLE database helps to identify potential mistakes or at least very uncommon transcriptions. This is why for this present thesis, PHOIBLE serves as a phonetic reference database to clean my data. Refer to section 4.6 where I present a possible use case of using PHOIBLE as a reference database.

4.5 Pronunciation dictionary coverage

As I am using two different datasets, I want to find out how these two datasets relate. I order to do that, I am using the WikiPron lists to write the NWS stories. In a way, I used the WikiPron data as a look-up table to create phonetic transcriptions for the NWS stories. This means that for each orthographic word in the short story, I search in the WikiPron data for the respective language for that specific grapheme sequence. If the word is in the WikiPron data, I write it to a separate transcription file. If the word is not in the WikiPron data, I write a capitalized version of the orthographic word into the same transcription file. Capitalizing words that are not in the WikiPron data allows me to easily count words that occur in the short story but are not found in the WikiPron data. Examples 4.1 and 4.2 show the orthographic version of the English NWS story and its phonetic transcription. Example 4.3 shows the phonetic transcription of the same English orthographic text using the WikiPron English word list as a look-up dictionary. I produced a text similar to example 4.3 for each language and compared it to the original NWS transcription of that language.

(4.3) ðə ˈnoɹθ ˌwɪnd AND (ð)ə ˈsʌn wəˇ DISPUTING ˈwɪtʃ wəz ðə ˈstɹɑŋgəˇ, wɛn ə ˈɹɹævələˇ ˌkəm əˈlɑŋ ˈɹæpt ɪn ə ˈwoɹm ˈklok. ðe əˈgɹid ðət ðə ˈwʌn hu ˈfəˇst səkˈsidəd ɪn ˈmekɪŋ ðə ˈtɹævələˇ ˈtek ɪz ˈklok ˌɑf SHOULD bi kənˈsɪdəˇd ˈstɹʃŋgəˇ THAN ðɪ ˈəðəˇ. ðɛnðə ˈnoɹθ ˌwɪnd ˈblu əz ˈhɑɹd əz i ˈkʊd, bət ðə ˈmoɹ hi ˈblu ðə ˈmoɹ ˈklosli dɪd ðə ˈ tɹævləˇ ˈfold hɪz ˈklok əˈɹaʊnd ɪm; AND ət ˈlæst ðə ˈnoɹθ ˌwɪnd ˌ gev ˈʌp ði əˈtɛmpt. ˈðɛn ðə ˈsʌn ˈʃaɪmd ˌaʊt ˈwoɹmli AND ɪˈmidiətli ðə tɹævləˇ ˈtʊk ˌɑf ɪz ˈklok. AND ˈso ðə ˈnoɹ ˌwɪnd wəz əˈblaɪʒ tɪ kənˈfɛs ðət ðə ˈsʌn wəz ðə ˈstɹɑŋgəˇ əv ðə ˈtu.

By comparing the text produced by the WikiPron data to the reference transcriptions from the JIPA articles, I could calculate the coverage, the WER and the PER score. The coverage is calculated by simply calculating the percentage of how many words from the NWS stories are in the WikiPron data of one language. The comparison and the analysis of the metrics gives the following insights. In addition, I manually checked some errors which gave me insights about the word lists in general:

- For some NWS transcriptions it is not clear whether their transcription is narrow or broad. On the other hand, sometimes there is no broad or narrow word list available for a specific language but only one of those. For the short stories where the type was unclear, I tried both word list types if those are available. The analysis shows that NWS transcriptions of unknown type, where the broad list is used as a look-up table gives better results than if the narrow word list is used as a look-up table. See, for example, Indonesian

(ind) or Hindi (hin) in table 9. As this is the case, it makes sense to treat the unknown transcriptions as broad.

- The IPA allows to transcribe intonation segments. In German, those correspond mostly to punctuation marks like end of sentence symbols or commas. But this must not be true for every case. It needs to be decided if those should be kept or potentially deleted.
- The pronunciation dictionaries sometimes included duplicates with different pronunciations. This is not surprising but still it needs to be handled well. A solution is to simply delete duplicate words.
- In order to do this very simple experiment, it is necessary to tokenize the texts. This works well for languages using the Latin script. For languages like Chinese or Korean this is more difficult to accomplish. I used a special Python library called *polyglot*[10] to tokenize languages using other scripts than alphabets.
- Interestingly enough, for some languages the coverage is really low although there are quite a lot of words in the word list. This is the case for German (narrow) where the coverage is only 22% (see table 9) and the word list contains more than 10,000 words. A manual analysis showed that some frequent words like 'sich' and 'und' are not in the word list. It becomes clear that a few thousand words are not enough to reach reasonable coverage. 10,000 - 15,000 words seem to be a lower bound for covering around 50% of this short text.
- For most of the languages, the PER is lower than the WER. This is a good sign, as it suggests that if a word is in the list, it is at least partially spelled the same as in the reference text. Still, it is surprising that for broad Spanish, the WER is actually lower than the PER. Results like this show that even if the words are covered, their phonetic transcription might be spelled differently. This is related to the fact that the IPAlpha is not really standardized. It always depends on the person who transcribes a text.

The results from this experiment are summarized in table 9. Generally it is good to see that most texts are at least partially covered by the pronunciation dictionary. It will later be interesting to see how the neural models perform on these short texts.

[10] `https://polyglot.readthedocs.io/en/latest/`

Iso 639-3	Coverage	WER	PER	Type ref	Type list	# Words list
cmn	85.15	93.07	59.26	unk	broad	133,686
deu	75.00	72.22	52.67	broad	broad	34,145
deu	22.22	97.22	84.85	narrow	narrow	10,984
ell	26.32	84.21	87.74	unk	broad	10,547
eng	92.04	83.19	37.27	broad	broad	57,230
eng	7.08	100.00	108.35	narrow	narrow	1,633
eus	5.75	96.55	97.95	broad	broad	1,742
ind	22.22	96.30	88.04	unk	broad	1,555
ind	1.85	100.00	101.69	unk	narrow	2,637
kat	44.29	90.00	73.85	broad	broad	15,123
mya	7.50	97.50	97.70	broad	broad	4,631
spa	64.95	46.39	48.64	broad	broad	60,677
spa	35.05	98.97	65.92	narrow	narrow	52,190
tha	83.85	88.20	40.62	unk	broad	15,050
tur	18.46	100.00	88.89	unk	broad	1,789
tur	6.15	100.00	92.53	unk	narrow	1,812
hin	31.75	100.00	84.00	unk	narrow	9,563
hin	57.94	93.65	69.81	unk	broad	10,812
kor	18.64	100.00	51.97	unk	narrow	14,141
fra	86.11	51.85	40.76	unk	broad	56,911
vie	96.58	79.49	48.66	unk	narrow	15,240
rus	94.79	95.83	56.30	unk	narrow	402,586

Table 9: The table shows the coverage, WER and PER when the pronunciation dictionaries are used to write *The North Wind and the Sun*.

4.6 Language profiles

Once I collected my datasets, I wanted to find out what characters they include. I collected phoneme and grapheme profiles of the WikiPron data and the NWS stories and compared it to the PHOIBLE dataset. A profile lists all used characters or tokens (this depends on your chosen settings) for a text and lists its frequency. All profiles were collected for each language and each dataset separately. For each language and each dataset I got three lists:

- **Grapheme list**: contains all graphemes in that language. Characters that need a base character like diacritics are shown together with their base character.

- **Phoneme list**: contains all phonemes in that language. Again, diacritics are shown with their base characters.
- **Phoneme cluster list**: Phonemes can be clustered into bigger sound groups. How to do this is an ongoing discussion, but I used the segments library to get the clusters (compare **?**).

Having this overview for the characters for each language allowed me to compare the character vocabulary to the characters or character clusters available in the PHOIBLE database. This comparison showed that quite a few characters are included in the WikiPron data and the short stories which cannot be found in the PHOIBLE database. My observations are listed below.:

- **Characters that are not included in the official IPAlpha chart**: Sometimes there are characters included in the phonetic version of the WikiPron data or the NWS short stories that are not a part of the IPAlpha. There might be a reason why the authors of the transcriptions decided to use a specific character that is not in the IPAlpha to denote a particular sound, but the reason is not always known. A possibility is to try and map it to a character that is available in PHOIBLE and that represents a similar sound (or even the same sound actually).
- **Tie bars**: The creators of PHOIBLE decided to exclude tie bars because they add no real value to the transcriptions[11].
- **Stress marks and other suprasegmentals**: Stress marks are not represented in PHOIBLE as they do not represent a sound. They are included in the NWS short stories. The same is true for other suprasegmental elements.
- **Tones**: Even within the IPAlpha there exist different conventions of how to represent tones. Some are better suited for different languages. Apart from different ways of representing tones, it is not always sensible to have tones represented at all. Mostly, tones are not written in orthographic versions as speakers of a language know how to pronounce the tones. I am going to explain more about tones in my datasets at a later stage.

I conducted this preliminary analysis at an early stage of my thesis. In chapter 6 I will clean the datasets to use them for training and testing. At this point, I will give a more detailed list of what characters are in my datasets but not included in PHOIBLE and what I cleaned in the datasets.

[11] `https://phoible.github.io/conventions/`

5 Experiments: Phonetic & Orthographic Word Length Correlation

Multilingual corpora, like the one I am using for the present thesis, are used more and more in NLP research. Having a multilingual corpus allows to perform multilingual analyses as introduced in chapter 2. Such analyses are more useful if our corpus represents linguistic diversity well. Having a *linguistically diverse* multilingual corpus instead of only having a multilingual corpus is important as multilingual does not necessarily mean that the languages in the corpus represent a broad spectrum of different languages. Sometimes, having a large number of languages in a corpus is referred to as linguistic diversity. But the number of languages is not enough to account for linguistic diversity. A linguistically diverse multilingual corpus ideally includes languages that are different from each other in terms of linguistic properties. More on linguistic diversity is presented in section 5.1.

Someone might ask why we need linguistically diverse corpora. One reason why we should use linguistically diverse corpora is that we want to be able to generalize well to as many languages as possible. If we can generalize well, it allows us to make statements about languages in general even if those languages are not in the corpus. If in our corpus there are only languages with the same properties, we cannot make assumptions about languages with very different properties. If we observe something for a subset of similar languages, we do not know if our observations are true for a different subset of languages with very different properties. If we make an observation based on a linguistically diverse corpus, we can assume that our observations generalize to other languages as, ideally, there is at least on language in our corpus that is similar to any other language.

As the corpus my work is based on is already designed to be linguistically diverse (see section 4.2), I will not go into more detail about the linguistic diversity of my corpus. The goal of the experiment presented in this chapter is to find out if phonetic versions of languages need to be treated as a completely different way of representing

a language. Or if they represent the same linguistic properties of a language as the orthographic version of that same language represents. This means: if we study a language, do we need both orthographic and phonetic texts of that language in order to fully represent the language in question. If both orthographic and phonetic texts represent one language equally well, it suffices to use only orthographic text to study a language. If both orthographic and phonetic texts represent different aspects of a language, we need samples of orthographic *and* phonetic text to properly study the language in question. Only if we can study *one* language properly, we can study multiple languages and compare them. More on the experiment is found in section 5.3.

5.1 Measuring linguistic diversity

If we would like to create linguistically diverse corpora, we need to measure linguistic diversity somehow to judge whether a corpus is linguistically diverse or not. While linguistic diversity in general is often neglected in present NLP research there are approaches to measure linguistic diversity. One very simple part of such a diversity score is the number of languages in our corpus as I have mentioned before. Although the number of languages on its own is not sufficient to measure linguistic diversity, a corpus with only two languages cannot be linguistically diverse either.

No matter how we intend to create such a diversity score, we will eventually need to compare languages and tell whether they are similar or not. This is needed as a multilingual corpus can only be diverse if the languages it contains are not all very similar. But they need to be similar to languages that are not in the corpus as those need to be represented as well. In order to compare languages, we need to be able to describe them properly. There are many different characteristics like language family, script, grammatical features and other characteristics of languages that can be used to describe a language. The entire process of describing languages and comparing them would take up another entire thesis. But with the help of my corpus I can make a contribution to push work on linguistic diversity. In the next section, I will have a look at one property of languages that can be used to compare languages to each other.

5.2 Mean word length as linguistic property of a language

A very simple characteristic to describe a language is to calculate the mean word length of a language. Although this task sounds very simple there are in fact multiple challenges to tackle before it is possible to calculate the mean word length of a language. Many of these challenges break down to a set of questions that need to be answered:

- **Tokenization**: In order to calculate word lengths, the texts need to be tokenized properly. This sounds extremely trivial, but in reality this can be a problem. In order to tokenize a text we need to define the notion of a token before.
- **Preprocessing**: Different scripts use different punctuation symbols. Is it necessary to exclude those? For example hyphens between two words. Do they count as part of the word? What to do about tones in the phonetic transcription? Should tones be excluded or are they part of the phonetic word? For every language, there are many uncertainties about how to preprocess the data similar to the one in example 5.1. In the end, preprocessing often breaks down to deciding what information in a text should be kept and what can be excluded because it does not add any valuable information.

  (5.1) The example of the English word 'sub-area' shows the difficulties of preprocessing. The question is how to treat hyphens in English. Does a hyphen add any valuable information to the text? What happens to character or token counts for different preprocessing options? I can think of four versions of how to process the word. It is likely that there are more.

  (a) Leave the word as-is: 'sub-area'
  Token count: 1, character count: 8

  (b) Delete the hyphen and merge the word: 'subarea'
  Token count: 1, character count: 7

  (c) Delete the hyphen and split the word: 'sub area'
  Token count: 2, character count: 7

  (d) Leave the hyphen to preserve the character count but split the word: 'sub- area'

Token count: 2, character count: 8

- **Counting characters**: In section 3.1 I have given an introduction to the Unicode standard and that there can arise problems when counting characters of a string. While there is no clear answer how to count characters, it is important to decide for one way to count the characters and do this for every language. Otherwise it is not possible to compare the results.

All of those questions need to be answered before conducting an experiment to calculate mean word lengths. As the texts I am using for this experiment are rather short, some of those challenges are already resolved because the phenomenon is not present in the corpus. Still, it is important to be aware of these challenges especially as I will conduct more experiments on the same datasets in chapter 6.

5.3 Mean word length correlation of phonetic and orthographic texts

In this section I present the actual analysis I performed. As I have pointed out in the introduction to this chapter, the question I would like to answer is:

- Is the difference between orthographic and phonetic text of the same language small enough such that we do not need samples of both orthographic and phonetic texts to properly represent the language?

One way to quantify the difference between the phonetic and the orthographic version of a language is to use the above explained property of mean word length. There are a few more steps necessary to answer the question. In order to answer this question I performed the following steps:

1. **STEP - CORPUS:** As a corpus I used the NWS corpus that I have collected (see section 4.3). It contains parallel examples of orthographic and phonetic written full text for 21 languages. Refer to table 11 for an overview of all languages.

2. **STEP - PREPROCESSING:** As I have pointed out in the section before, it is necessary to preprocess both orthographic and phonetic texts and tokenize them before I can perform any calculations. I solved the above mentioned problems in the following way:

    - **Tokenization**: Luckily, I already knew about a tokenizer that I can use for the orthographic texts of Chinese, Japanese and Thai that are difficult

to tokenize. I used the same tokenizer *polyglot*[1] that I have already used in previous experiments. For all other languages and all phonetic texts I chose the simple strategy to tokenize the texts at white spaces.

- **Preprocessing**: I performed a minimal cleaning of both orthographic and phonetic texts. The characters I excluded are:
  - Orthographic text: all punctuation symbols
  - Phonetic text: segment marker symbols and tones
- **Counting characters**: In order to count the characters, I used the default Python string length function for the orthographic tokens. For the phonetic texts, I used the segments library to tokenize the individual phoneme tokens on segment basis (see section 3.1). I counted the number of segments that I have received for each phoneme.

Table 10 gives an example of how the word lengths can be counted for orthographic and phonetic texts.

Graphemes	Length graphemes	Phonemes	Length phonemes
t h e	3	ð ə	2
n o r t h	5	ˈn o ɹ θ	4
w i n d	4	ˌw ɪ n d	4
a n d	3	ə n	2
t h e	3	ð ə	2
s u n	3	ˈs ʌ n	3

Table 10: This table gives an example for the word length of orthographic and phonetic texts. The text is a parallel example of the first few tokens of the English NWS story. The characters are separated by white space to make manual counting easier. The phonemes are processed by the segments library such that phonetic segments are counted. We can see that, for example, the IPAlpha symbols marking stress (ˈ and ˌ) are one segment together with the subsequent character.

3. STEP - CALCULATING MEAN WORD LENGTHS: For each language, I calculated two mean word lengths:

1. the mean word length for the orthographic version of the NWS short story for one language
2. the mean word length for the phonetic version of the NWS short story

[1] `https://polyglot.readthedocs.io/en/latest/`

for one language

In order to calculate the mean, I summed up all word lengths for one text and divided it by the number of tokens of that text. All the means for both texts for all languages are found in table 11. To calculate the mean word lengths I exclusively used the narrow transcriptions if those were available for the specific language. If not, I used the broad transcription or just the one I had with the unknown transcription type.

Iso396-3	**Language name**	**Mean word length orthographic**	**Mean word length phonemes**	**Type**
aey	Amele	5.21	5.5	unk
arn	Mapudungun	4.81	4.65	narrow
cmn	Chinese	1.59	4.44	unk
deu	German	5	4.35	narrow
ell	Greek	4.62	4.23	unk
eng	English	4.19	3.46	narrow
eus	Basque	5.3	4.98	narrow
fra	French	4.55	3.18	broad
hau	Hausa	3.8	4.07	narrow
heb	Hebrew	6.62	6.57	unk
hin	Hindi	3.53	3.93	narrow
ind	Indonesian	5.92	5.25	unk
jpn	Japanese	1.59	3.77	unk
kat	Georgian	5.99	6.32	narrow
kor	Korean	2.85	6.56	unk
mya	Burmese	10.22	8.15	unk
pes	Farsi	3.99	5.03	unk
spa	Spanish	4.62	4.36	narrow
tha	Thai	3.25	3.03	unk
tur	Turkish	6.74	7.02	broad
vie	Vietnamese	3.24	3.87	unk

Table 11: This table shows the mean word lengths for the NWS phonetic and orthographic texts.

4. STEP - CALCULATING CORRELATION This last step is the most important step of this experiment. I calculated the correlation between the mean word length for phonetic texts and orthographic texts. For this step, I do not distinguish anymore between languages. The correlation is calculated on two lists, one containing all mean word lengths for all orthographic texts and the other one

containing all mean word lengths for the phonetic texts.

The correlation I used is the Spearman correlation between phonetic and orthographic mean word length:

$$\rho = 0.66$$

The correlation value is a real number between 0 and 1. If the correlation between two factors is strong (which means it is close to 1) this means that the two factors behave in a similar way. For my experiment this means that both mean word lengths for orthographic and phonetic texts behave in a similar way. A Spearman correlation of 0.66 is not extremely strong, but there definitively is some correlation.

Although this study is very small, it indicates that phonetic and orthographic texts are not representing contrary properties of a language as there is some correlation between their mean word lengths. It is up to future research to further investigate how orthographic and phonetic texts relate.

6 Experiments: Automatic Grapheme-to-Phoneme Conversion

In this chapter, I present the experiments I conducted to obtain computational models to create phonetic transcriptions. The datasets I use to train the models are presented in section 6.1. The model that I am using for my G2P experiments is explained in section 6.2. In order to compare my results to already existing models, I will use the results of the SIGMORPHON tasks 2020 and 2021. They present results for quite a few languages. Also, they use the same data type as I do. I introduced some of those models in chapter 3.

Reproducability All scripts that I used to preprocess, train and evaluate my models are found on my GitHub[1]. I installed the model on my machine as specified on the model's GitHub[2].

6.1 Datasets

As a result of my first practical part in chapter 4, I presented two datasets that I will now use to perform the experiments in this chapter. The WikiPron corpus serves as a training dataset for my model. Note that WikiPron continuously adds data to their repository. This means that there might be new languages that are in the 100 language corpus and that I did not use. Additionally, there were languages which had a WikiPron dictionary that was smaller than 1,000 grapheme-phoneme pairs. ? have found that a dataset with 1,000 samples is not enough to train a decent model for G2P conversion which is why I excluded datasets for languages with less than 1,000 samples. Their finding is relevant for my thesis as they based their conclusion on similar WikiPron data. For some languages in the WikiPron data there is a cleaned version available. Whenever a filtered version is available I

[1] https://github.com/theDebbister/masterThesis

[2] https://github.com/cmusphinx/g2p-seq2seq

used the filtered version as the basis for my experiments.

The NWS corpus is much smaller which is why I will use it as a test dataset only. As the NWS corpus is quite well known among linguists or at least among phoneticians, it will hopefully give interesting insights when testing the models on these short texts.

Table 5 shows all languages that I am using in my experiments. Generally, broad and narrow transcriptions are treated as separate languages and thus trained separately as well. The same is true for dialects if there is any information available. For example: for American and British English I will train different models.

6.2 CMUSphinx

For my personal experiments, I decided to use the CMUSphinx seq2seq G2P model. This model has been used in the SIGMORPHON 2021 task as part of the Dialpad ensemble [**?**]. It was not used on many languages but promised a good performance which is why I decided to use this model for this present thesis. The CMUSphinx model is a transformer-based seq2seq model implemented with tensorflow. Unfortunatley, there is not a lot of information available online about the model. There exists a pre-trained version of the model for G2P. However, they use a transcription format other than IPAlpha which means it cannot be used for my dataset[3].

For all my experiments, the following two settings of the model are the same:

- The CMU model splits the data automatically if not specified otherwise. The automatic split is 85% training data, 5% development data and 10% test data. I did not change this split.
- The hyperparameters are left as default if not mentioned otherwise. This means that I did not adapt the model and used it as it can be found online. More on the default settings can be found on the model's GitHub.

The model takes as input parallel examples of grapheme and phoneme sequences. This is typically a tsv (tab-separated-values) file structured as shown in table 6. Sometimes this data format is referred to as a input dictionary.

[3] `https://github.com/cmusphinx/g2p-seq2seq`

6.3 Training settings

There are different settings which I will use to train the models and analyse their performance. The settings are designed in a way that each setting introduces a new or more complex adaption of the previous setting. The first experiments are set up very simply and without much effort. Then I continuously add more complexity. Below I added a short description of each setting as an overview. I will add more details on the individual steps in separate sections.

Setting 1: Baseline short This is the most basic setting. I will train a model for each language to get a baseline result. The model is trained with the least amount of effort. The model is trained for the minimum number of steps which is 10,000 steps in this case.

Setting 2: Baseline long This setting is similar to setting 1 except that the model is trained as long as possible for each language. All models have been trained for 200,000 steps. This setting is supposed to show if training for a considerably longer time changes the results a lot or if it does not result in a better performance.

Setting 3: Baseline clean short I will train another model that is the same like the baseline short, but I will use the cleaned WikiPron data. What I will clean is described below (section 6.4).

Setting 4: Baseline clean long The same as setting 3 except that the models were trained for as long as possible like in setting 2.

Setting 5: Feature input version 1 short The final experiments will be with phonetic features as input. I used the cleaned WikiPron data from setting 3 and added phonetic features to it. In section 6.5, I explain how I encode the features.

Setting 6: Feature input version 1 long The same as setting 5, but again the models are trained for 200,000 steps like in setting 2.

Setting 7: Feature input version 2 short This setting is the same as setting 5, but with a different version of the input feature data.

Setting 8: Feature input version 2 long The same as setting 7, but again the models are trained for 200,000 steps like in setting 2.

The training time for the short model for one language with 10,000 steps is about 20 minutes. The long models took another six hours to complete the training per language.

6.4 Preprocessing

In section 2.3, I wrote about the incompleteness and difficulties of transcribing using the IPAlpha. How exactly a sound is mapped to an IPAlpha symbol also depends on whoever transcribes a particular text. The WikiPron data has been put together by many different people. There are conventions on how to add transcriptions to Wiktionary[4], but there still might be inconsistencies. Other than that, it is always possible that a transcription is correct, but computational models just cannot handle it well. That said I will carefully examine and clean the datasets. In chapter 5, example 5.1 I have already given an example of the type of questions that need to be answered to preprocess a text to perform an analysis on that text. This current section is centered around a multiple of such questions. Some of the preprocessing will be done for both of my dataset to be consistent in the preprocessing. As the datasets are very different there needs to be done individual preprocessing for each dataset as well.

The first analysis to decide on what to preprocess is a character based comparison of the datasets and the PHOIBLE database. This means that I used the Python segments library[5] to split all the phonemes into Unicode characters. Then I compared those characters to the PHOIBLE database. As I have explained in section 4.4, in can use the PHOIBLE database to find phonemes that are uncommon for a specific language. While it is possible that a correctly used phoneme is not in PHOIBLE, it still gives a good overview of uncommon phonemes in the transcription or even points out mistakes in my data. This character based cleaning is done for both datasets. A more detailed list of what needs to be cleaned is found in table 12.

There are two more character types not found in table 12 that I removed from the datasets. The first is **tones**. The reason for removing tones is that it is not possible to infer tones from graphemes. Although they are used to distinguish meaning, they are not written down, but usually just known. This means that there are grapheme

[4] `https://en.wiktionary.org/wiki/Wiktionary:Pronunciation`

[5] `https://pypi.org/project/segments/`

sequences that look exactly the same and their transcription is the same as well *except* for the tones. Example 6.1 shows the same Chinese grapheme together with its transcription. The superscript number represent the tones. For none of the three examples are the tones the same. For the exact same Chinese grapheme there exist three different possible pronunciations. But if only shown the grapheme, there is absolutely no way to find out which tones are meant:

(6.1) (a) 浸 t͡ɕʰ i n ^{214}

(b) 浸 t͡ɕʰ i n ^{51}

(c) 浸 t͡ɕʰ i n ^{55}

For the reason of the ambiguity shown in example 6.1, I decided to exclude tones. There are different ways to represent tones and I excluded all of them (refer to figure 1 for the different tone versions).

The second additional preprocessing step I performed is to remove all **punctuation symbols**. This accounts mostly for the orthographic version of the NWS short stories. For character-based modelling, punctuation does not add any valuable information as this only becomes relevant on a sentence basis.

In example 6.2 I list a few phonetic transcriptions before and after cleaning.

(6.2) This example shows strings from the WikiPron data before and after cleaning them. The left-hand side of the arrows shows the uncleaned version and the right-hand side the cleaned version. I added the language code and the transcription type and the original grapheme sequence in parentheses.

(a) deu, narrow

aː b ə n t m aː l t͡s a ɪ̯ t → aː b ə n t m aː l t s a ɪ̯ t (Abendmahlzeit)

(b) fin, broad

ɑː lː o t ɑˣ → ɑː lː o t ɑ ʔ (aallota)

(c) cmn, broad

t͡ɕʰ i n ^{214} → t ɕ ^{h} i n (浸)

(d) eng uk, narrow

ə l ɜˇ d ʒ ɪ k → ə l əˇ d ʒ ɪ k (allergic)

Phon.	Unicode name	Repl.	Explanation
ˈ	MODIFIER LETTER VERTICAL LINE	NULL	These are all IPAlpha suprasegmentals except the long and half long marker and the extra short (ː ˑ ˘). The reason why these were excluded is that they are not meaningful on the character level. The vertical lines, for example, mark intonation groups which only matter in a larger sentence or text context. There are a few rare occurrences of COMBINING VERTICAL LINE ABOVE which is probably meant to be MODIFIER LETTER VERTICAL LINE as they look similar. It is excluded as well.
ˌ	MODIFIER LETTER LOW VERTICAL LINE	NULL	
|	VERTICAL LINE	NULL	
‖	DOUBLE VERTICAL LINE	NULL	
.	FULL STOP	NULL	
‿	UNDERTIE	NULL	
͡	COMBINING DOUBLE INVERTED BREVE	NULL	Both tie bars below and above are excluded in PHOIBLE[6] which is why I am excluding it as well. Put plainly, those do not add any additional information that cannot be derived otherwise.
͜	COMBINING DOUBLE BREVE BELOW	NULL	
ɜ̯	LATIN SMALL LETTER REVERSED OPEN E WITH HOOK	ə̯	Both base letters are very similar vowels. It is just more common to use the latter than the former.
g	LATIN SMALL LETTER G	ɡ	The IPAlpha 'g' has a different code point and is a different character than the typical keyboard small Latin 'g'. This is just an IPAlpha decision. For some fonts the two characters do not look different, for some they do.
∼	SWUNG DASH	NULL	All of these characters make out less than 1% of their respective dataset, most of the time it is less than 0.1%. A close examination of the dataset and the Wiktionary transcription conventions for the respective language did not show any reason why to keep the phonemes. Note that the 'v' for the tilde is only there for correct representation.
,	COMMA	NULL	
ṽ	TILDE	NULL	
ᵊ	MODIFIER LETTER SMALL SCHWA	NULL	
ˣ	MODIFIER LETTER SMALL X	ʔ	The ˣ only occurred in the broad Finnish transcription and is used to denote possible gemination. In the narrow transcriptions there is a glottal stop instead. The occurrence of glottal stops and gemination follows the same rules. Therefore, for consistency, the gemination ˣ is mapped to a LATIN LETTER GLOTTAL STOP.
( )	(SUPERSCRIPT) [LEFT | RIGHT] PARENTHESIS	NULL	Parentheses are used to denote optionality for phonemes or tones. WikiPron actually discards those but keeps the content[7]. I will do the same for all parenthesis found.

Table 12: The table shows what phonemes where changed or excluded and what the reason is for this preprocessing. All characters that were excluded are replaced by a NULL value.

NWS corpus

I had to transform the NWS stories into dictionary format in order to be able to use them as testing data. It was necessary to tokenize both orthographic and phonetic texts and then align orthographic tokens and phonetic tokens. As the number of tokens (when split naively at white space) is not always the same for orthographic and phonetic text, it was necessary to align the orthographic and phonetic texts for some languages manually. While this is not a problem for languages that I know how to pronounce, it is a bit tricky for languages completely unknown to me. Luckily there are tools online that provide a rough pronunciation of a word in a given language or even a phonetic transcription (although rarely in IPAlpha)[8]. For my experiments described in section 4.5 and chapter 5 I had to tokenize both phonetic and orthographic texts for most of my languages. Please refer to the before mentioned section and chapter to find more details on tokenization.

Apart form the characters listed in table 12 that I excluded from the NWS texts, there were a few more characters that needed to be excluded. As the models cannot handle characters that are not in their vocabulary, I needed to remove the unknown characters to be able to evaluate the models on the NWS stories. The vocabulary of the neural models I use contains all characters that were in the training data. The models I train consequently contain all characters that are in the training split of the WikiPron data. However, for some characters, they were in the vocabulary, but the input data was in the wrong Unicode normalization form as described in section 3.1. For example, some characters were precomposed characters in the WikiPron training data while they were not precomposed in the NWS stories I used to test the models. Both characters in the model's vocabulary and the NWS stories look exactly the same but the model treats them as two completely different characters because they have a different code point. I could easily solve that problem by converting the NWS short stories to the same normalization form as the model's vocabulary. See example 3.3 where I illustrate the difference between precomposed and non-precomposed characters.

WikiPron

Just as for the NWS corpus, I needed to do some additional preprocessing for the WikiPron data. As the CMU model does not expect the orthographic input se-

[8]For most languages I could use Google translator (`https://translate.google.com/?hl=de&sl=ko&tl=de&op=translate`), for languages like Hebrew, I could ask someone who knows the language to help me out.

quences to contain white space, I replaced any white space in the input grapheme sequences with underscores. This was necessary only for Vietnamese. In a first run, I did not replace white space by underscores. When running the evaluation on the Vietnamese model, it did not work but showed an error message. As the model creates a vocabulary file, I had a look at it which revealed the following:

- The model does not actually split the input file at tab characters, but splits it at spaces and then uses the first item of the resulting list as input grapheme and the rest as a list containing the output phoneme segments. Consequently, if the input grapheme contains a white space, everything following it will be in the phoneme part. Part of the input will be treated as a phoneme segment in the training which results in a huge vocabulary and wrong predictions.

In Japanese there is the superscript phoneme /$^{\beta}$/. This phoneme actually means compression, which is a special type of rounding[9]. This special case of rounding in Japanese is not reflected in PHOIBLE nor in the official IPAlpha but this does not mean it is wrong. As it is quite common in the Japanese data, I decided to keep the phoneme /$^{\beta}$/. This is one of those cases where the PHOBILE data helped to find a uncommon transcription that is not necessarily wrong.

A last thing I did to clean the WikiPron datasets was to exclude duplicate grapheme sequences with different pronunciations. Although ambiguities and multiple possible pronunciations for one word are linguistically speaking very common, it is not possible for a neural model to distinguish such cases without any context. As G2P modelling happens on character basis and not on word basis there is no context available in our case that could account for at least some ambiguities. Also, in the WikiPron data that was preprocessed for the SIGMORPHON task, duplicates were excluded as well. It makes sense to clean all datasets in a similar way to allow for better comparability.

6.5 Feature encoding

In order to incorporate the phonetic features into the dataset, we decided to add what we refer to as 'flags' to the phonemes. Generally, the idea is to encode the phonetic features that PHOIBLE provides for each phoneme and add them to the WikiPron dataset. I will list the steps of my experiments in the following. For all feature experiments I used the WikiPron data that I had cleaned according to what is explained in section 6.4.

[9] `https://en.wikipedia.org/wiki/Roundedness`

Version 1

The first idea to incorporate the phonetic features into the datasets is to use the PHOIBLE feature set for each of the phonemes and encode the information that is found in those features. As there are 37 features for each phoneme they allow to encode much more and more finegrained information than one single character can. By encoding the phonetic features we can add more information that the model can use to learn how graphemes are mapped to phonemes. For example: in some languages, there are vowel-consonant patterns. In English a word typically needs at least one vowel and it is uncommon to have consonant sequences longer than three letters. If it was possible to explicitly encode for each phoneme if it was a vowel or a consonant, the model could abstract that information more easily an learn how sound patterns work.

In order to add phonetic information to the datasets, I performed the following steps:

1. **STEP** I split the list of all 37 PHOIBLE features (see section 4.4) into two sets of PHOIBLE features. After this step, I had two disjoint sets of features. One containing the first half of all features, the other one containing the second half of all features.

2. **STEP** For each of the two sets, I used two different capital letters to encode all possible feature combinations that can be obtained by combining all features in the set with all other features in the same set. If there had been two features in the set, 'syllabic' and 'consonantal', there would have been nine possible combinations. To each of these combinations I assigned one string of two capital letters. Table 13 shows how this step looks like for a small example.

3. **STEP** Once I completed the above step, I had two sets of strings that I could combine to encode all possible combinations of features for each of the phonemes. Each string represents a different feature combination of one of the two subsets of the features.

4. **STEP** Then, I could assign two strings to each phoneme in my data. The first string represents the first half of the phoneme's features, the second string represents the second half of the phoneme's features.

After completing the above steps, I got an output that looks similar to the examples below (examples 6.3 and 6.4). The combination of the two flags is unique for each phoneme. The model will interpret each two-letter feature string as one phoneme segment.

syllabic	**consonantal**	**Feature encoding**
+	-	AA
+	0	AB
+	+	AC
-	-	AD
-	0	BA
-	+	BB
0	-	BC
0	0	BD
0	+	CA

Table 13: This table shows an example of how the feature encoding works. In this case there are only two features involved. Nonetheless, the procedure works the same for more features. Refer to section 4.4 to understand the meaning of the values for each feature.

(6.3) p AB CD k^h AA BC

(6.4) k^h AA BC ʊ BF CC

A problem with this approach is that the sequences get really long if all phoneme segments are now represented by three segments. Examples 6.3 and 6.4 show that the phoneme sequence gets three times longer. Originally there were only two segments separated by a white space while after the feature flags are added there are six segments. However, the CMU model separates the phoneme sequences at white spaces and cuts off every segment after 30 segments. This means that only phoneme sequences that are no longer than 10 segments will be represented in full when for each phoneme two additional feature flags are added as in the example above. Example 6.5 shows how the CMU model cuts off the phoneme sequences.

(6.5) The CMU model cuts off phoneme sequences that are longer than 30 segments. Sub-example (a) shows the original phoneme sequence without features and the corresponding grapheme sequence. Sub-example (b) shows the full length phoneme sequence with features and sub-example (c) shows the shortened version with features. It is easily understandable that there is a lot of information lost if a sequence is shortened like that.

(a) yː b ʊ ŋ s b uː x z a ɪ̯ t ə n
Übungsbuchseiten

(b) yː BA EF b AH FG ʊ AN GE ŋ AG EC s AJ FE b AH EE uː BA FF x AJ GH z DE HG a BG FG ɪ̯ DC EH t AB HH ə AA FE n BB EE

(c) yː BA EF b AH FG ʊ AN GE ŋ AG EC s AJ FE b AH EE uː BA FF x AJ GH z DE HG a BG FG

Many original phoneme sequences are longer than 10 segments. It is possible to increase the limit of the model after how many segments to cut off segments. However, 30 segments is already quite long. There is no point in setting this limit too high as neural models like the CMU model still experience difficulties in processing sequences that are too long. Not only the sequence length is influenced but also the vocabulary size. Each flag will be added to the vocabulary which increases the vocabulary by a lot. This makes it harder for the model to pick one character from the vocabulary as output as there are more options available.

It is therefore not surprising that the model did not perform well. In fact, the WER of the model trained on the German data with feature version 1 was about double the WER of the long model trained on the data with no features. Given this results, I decided not to train any more models for this feature version and look for another way of how to encode the features.

Version 2

For the second attempt at encoding phonetic features I tried a different approach. Instead of adding two flags, I added only one flag after each phoneme. Also, this time the idea was not to add a unique flag to each phoneme but to encode only some features for all phonemes. This means that for some phonemes that have overlapping features the same flag was added. As the phonemes were not replaced by the flags, but the phonemes were kept in the text, there is no information lost.

The intuition behind this second approach is to encode high-level patterns in phoneme sequences. For example, as vowels typically have some overlapping features, very similar vowels will be encoded using the same flag. Again, in order to enrich my data with the features, I followed specific steps which I will describe in more detail below:

1. STEP As a very first step, I collected a list of all unique phonemes in PHOIBLE together with their features. I will later use this set to encode each phoneme.

2. STEP In contrast to the first feature version, for the second feature version I com-

puted the features based on the phoneme inventory of each language. Consequently, I collected additional sets all PHOIBLE phonemes for *each language separately*. All of the following steps are conducted for each language separately.

3. STEP As a next step, I collected a set of all phonemes for each language that are found in the training data for the models which is the WikiPron data in my case. After this step, I have two sets of phonemes for each language:

1. one set of all phonemes in PHOIBLE for the respective language
2. one set of all phonemes in the WikiPron training data for the respective language

Taking the intersection of these two sets gives me the set of phonemes which are found in PHOIBLE and in the WikiPron data for that language.

4. STEP For each phoneme in the WikiPron data, I got the PHOIBLE feature vector for that phoneme. If a phoneme is not found in PHOIBLE but in the WikiPron data, I just ignore that phoneme for the next steps.

5. STEP This step is the key step of the current feature version. I decided what features to encode for each language performing this step. In order to choose appropriate features for each language, I calculated the Pearson correlation between the features of all phonemes for each language. The idea behind calculating the correlation between the features is to exclude features for one language that are not important for that language. If I find two features for the phonemes of a language that have a strong correlation, it means that those two features behave in a similar way. For example:

- There is a phoneme /ŋ/ that has feature 'sonorant' marked as '+' and feature 'continuant' as '-'.
- Then there is another phoneme /n/ that also has feature 'sonorant' marked as '+' and feature 'continuant' as '-'.

Based on the above very small example, we could argue that whenever the feature 'sonorant' is '+', then the feature 'continuant' is '-'. If we observe this pattern for those two features for many phonemes of a language, then there is a correlation between the two features 'sonorant' and 'continuant'. It means that when we only observe the feature 'sonorant' to be '+', we can infer that the feature 'continuant' is '-'. For all features of all phonemes of a language, the calculation is a bit more complex but the intuition is the same. When we

study table 15 more closely, we can actually see that for some languages (for example Chinese or German) only 'sonorant' was encoded but not 'continuant'. This means, that there actually is a correlation between 'continuant' and at least one other feature in those languages. Having two strongly correlating features in a dataset means to encode the same information twice. And this is exactly what we would like to avoid.

I only encoded those features for a language whose correlation with no other feature within that language is higher than a certain threshold. For this experiment I set the correlation threshold to be 0.5. If the threshold was too high, for many languages, no correlating features were found. This is not surprising as features in general do not make a lot of sense if they are all encoding the same information. This is why I set the threshold relatively low which means I did not encode a lot of features.

The reason why I suggest that this feature encoding strategy can be used to find important features for a language is that all languages only use a relatively small subset of phonemes. This means that many features are not important for a language as those features are not used to distinguish meaningful sounds in that language. In order to recognize high-level patterns, only the most distinctive features are important. In table 15, I list all features that I encoded for each language.

6. STEP Once I have all the features for one language that I want to encode, I can actually encode them. This step is similar to the encoding of the features of the first version. I get all combinations of the set of features to encode for each language (similar as in table 13) and assign it a different capitalized two-letter string. Table 14 lists a few example strings for how the input data changed after enriching it with features.

There are still a few phonemes that are not in the PHOIBLE set that are in the WikiPron data. These will just not be encoded. An exception is the length marker. There are a few long vowels that are not found in the PHOIBLE data. As there is the 'long' feature in PHOIBLE, it is easily possible to use the base character feature vector and mark the 'long' as '+'. This is the only thing I changed for the following experiments. All other features vectors are just used as they are retrieved from PHOIBLE. I encoded the features in a jupyter notebook which is in my GitHub repository[10].

[10] `https://github.com/theDebbister/masterThesis/blob/master/data/MA_Encode_features.ipynb`

Grapheme	Phoneme	Phonemes with features
a	ʔ aː	ʔ AK aː AM
aa	aː	aː AM
aachen	aː x ə n	aː AM x AJ ə AL n AG
aachener	aː x ə n ɐ	aː AM x AJ ə AL n AG ɐ AN
aachenerin	aː x ə n ə ʁ ɪ n	aː AM x AJ ə AL n AG ə AL ʁ AJ ɪ AL n AG
...	...	...
übungen	ʔ yː b ʊ ŋ ə n	ʔ AK yː BA b AH ʊ AN ŋ AG ə AL n AG
übungsbuch	yː b ʊ ŋ s b uː x	yː BA b AH ʊ AN ŋ AG s AJ b AH uː BA x AJ
üppig	ʏ p ɪ ç	ʏ AN p AH ɪ AL ç
üsselig	ʏ z ə l ɪ ç	ʏ AN z AJ ə AL l AE ɪ AL ç
œuvre	œː v ʁ ə	œː BA v BC ʁ AJ ə AL

Table 14: This table shows parallel examples for the first few and the last few entries of the German broad WikiPron data file for the feature encoding version 2. It shows the graphemes, the unchanged phoneme sequence and the phoneme sequence with features. In the original file, the phoneme column is omitted. The feature encodings are not unique for each phoneme. I marked an example for the same encoding for two phonemes in red. Note that the two phonemes are very similar (see figure 1), thus have similar features and consequently the same feature encoding.

While encoding these features, I noticed that some phonemes are not listed as individual phonemes in PHOIBLE but are listed as allophones of one phoneme that is in PHOIBLE. This means that some broad transcriptions contain allophones which, ideally, should not be the case (see section 2.3). Also, it means that I did not have features for these phonemes which means that those were not encoded. It would be possible to replace all allophones for each language with the respective phoneme. However, this would have gone beyond the scope of this present thesis.

6.6 Results

For the evaluation of my models, I first used the WikiPron test sets. The test set is a subset of the entire WikiPron dataset for one language as explained in section 3.4. In addition, I tested each model on the respective NWS story if available in that language. Below, I first discuss the results separately for the short (setting 1, setting 3 and setting 7) and the long (setting 2, setting 4 and setting 8) models. I will mostly compare the WER scores of the model, as this is common in research.

ISO 396-3	Type	Features
all langs	-	tone, stress, syllabic, short, consonantal, tap, trill, lateral, labial, epilaryngealSource, spreadGlottis, loweredLarynxImplosive
cmn	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
deu	narrow	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
ell	broad	long, sonorant, continuant, delayedRelease, nasal, coronal, dorsal, constrictedGlottis, raisedLarynxEjective
eng uk	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
eng uk	narrow	long, continuant, nasal, constrictedGlottis
eng us	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
eng us	narrow	long, continuant, nasal, constrictedGlottis
eus	broad	long, continuant, delayedRelease, nasal, constrictedGlottis, raisedLarynxEjective
fin	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
fin	narrow	long, continuant, delayedRelease, nasal, constrictedGlottis, raisedLarynxEjective
fra	broad	long, nasal, constrictedGlottis, raisedLarynxEjective
hin	broad	constrictedGlottis, raisedLarynxEjective
hin	narrow	nasal, constrictedGlottis, raisedLarynxEjective
ind	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
ind	narrow	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
jpn	narrow	long, delayedRelease, coronal, dorsal, constrictedGlottis, raisedLarynxEjective
kat	broad	long, continuant, delayedRelease, nasal, coronal, constrictedGlottis
kor	narrow	continuant, nasal, constrictedGlottis, raisedLarynxEjective
mya	broad	long, delayedRelease, nasal, raisedLarynxEjective
rus	narrow	long, continuant, nasal, coronal, dorsal, constrictedGlottis, raisedLarynxEjective
spa ca	broad	long, sonorant, continuant, delayedRelease, nasal, dorsal, constrictedGlottis, raisedLarynxEjective
spa ca	narrow	long, sonorant, continuant, nasal, constrictedGlottis, raisedLarynxEjective
spa la	broad	long, sonorant, continuant, nasal, coronal, dorsal, constrictedGlottis, raisedLarynxEjective
spa la	narrow	long, sonorant, continuant, nasal, constrictedGlottis, raisedLarynxEjective
tgl	broad	long, continuant, delayedRelease, nasal, constrictedGlottis, raisedLarynxEjective
tgl	narrow	long, sonorant, continuant, nasal, dorsal, constrictedGlottis, raisedLarynxEjective
tha	broad	nasal, constrictedGlottis, raisedLarynxEjective
tur	broad	long, continuant, nasal, constrictedGlottis, raisedLarynxEjective
tur	narrow	long, nasal, constrictedGlottis, raisedLarynxEjective
vie	narrow	long, nasal, coronal, constrictedGlottis, raisedLarynxEjective
zul	broad	continuant, delayedRelease, nasal, coronal, constrictedGlottis, raisedLarynxEjective

Table 15: The table shows all features that were encoded for each language for feature version 2. Some of them are encoded for all languages, those are at the top of the table. All features that are listed for each language have a correlation lower than 0.5 with all other features listed for that language.

Also, the PER scores are a bit harder to interpret, as the edit distance it involves is not very intuitively understandable as there are many different ways of how to get to the same result.

To evaluate the models that were trained on data containing feature flags, I cleaned out the features to calculate the scores on the predicted phonemes only. Leaving in the feature flags would make it difficult to compare it to the other models' results.

6.6.1 Results: short models

Table 16 shows the results for all models that I trained for 10,000 steps. It is interesting that the results for all three model types are actually very similar. I was surprised by the fact that for a vast majority of the languages the version trained on the cleaned data did not improve the performance although the cleaning was supposed to make the data more consistent. For example, excluding duplicate words with different phoneme sequences should remove some ambiguities. However, it is also possible that it actually introduced more ambiguities as I randomly took one version of the duplicates which might be an uncommon one. An interesting result is the one for Zulu [zul]. The cleaned model performed a lot better than the other two. As this result seemed rather strange, I had a close look at the data again. Manually comparing the test prediction for Zulu to the reference phonemes (there are only 167 words in the test set) reveals that the model almost exclusively got the tones wrong. Those were excluded in the cleaned version which might explain the wide gap. Finnish [fin] had a very similar result but the gap was not as large. This finding shows how important appropriate preprocessing is.

Table 17 shows a comparison between my short models and the SIGMORPHON models. The results for the shared task are the same no matter to which of my models I compare them. As we will see in the next section, my long models performed better than the short ones presented in the current section. Therefore, I will not discuss the results at this point for the short models compared with the shared task. I will only discuss my results of the long models compared with the SIGMORPHON models.

6.6.2 Results: long models

Table 18 shows the results for all models that I trained for 200,000 steps. The results and the differences between the models are very similar to the short models. The observation for Zulu [zul] is still the same. In fact, the long Zulu model trained on

Iso 639-3	Type WikiPron	WER BS	PER BS	WER BS-clean	PER BS-clean	WER F2	PER F2
cmn	broad	19.6	4.5	**18.1**	5.0	20.3	6.0
deu	broad	40.3	5.4	**38.1**	5.3	40.2	6.6
deu	narrow	**52.1**	7.5	59.9	9.4	59.9	10.4
ell	broad	**9.8**	0.9	10.4	1.0	11.5	1.1
eng us	broad	54.4	10.6	**53.0**	10.8	57.7	12.1
eng us	narrow	84.6	32.0	**84.2**	31.8	86.6	32.2
eng uk	broad	**48.6**	9.5	50.0	10.1	51.8	11.1
eng uk	narrow	**88.8**	32.2	93.8	34.5	90.5	31.2
eus	broad	**19.4**	2.8	22.8	2.8	23.6	4.0
fin	broad	**5.8**	0.4	11.3	1.3	11.5	3.1
fin	narrow	14.3	1.0	**7.1**	0.7	17.1	4.3
fra	broad	**7.2**	1.0	8.8	1.3	9.1	1.6
hin	narrow	8.4	1.4	10.1	1.6	**8.3**	1.3
hin	broad	**5.6**	1.2	7.3	1.2	6.9	1.2
ind	broad	**35.3**	5.3	36.3	4.8	38.6	5.7
ind	narrow	**43.5**	5.5	44.1	6.0	44.3	6.1
jpn	narrow	**6.6**	0.9	6.7	0.9	6.8	1.1
kat	broad	**0.3**	0.2	1.1	0.2	1.7	0.9
kor	narrow	28.7	4.6	27.0	4.4	**26.7**	4.9
mya	broad	34.2	7.4	**34.1**	7.1	87.5	18.8
rus	narrow	**14.8**	1.7	16.4	1.9	22.0	3.6
spa ca	broad	**2.3**	0.3	2.7	0.4	5.3	1.5
spa ca	narrow	**3.6**	0.5	9.1	1.5	8.4	1.6
spa la	broad	**2.6**	0.3	2.8	0.5	4.4	1.0
spa la	narrow	3.3	0.4	**3.2**	0.3	5.4	0.8
tgl	broad	**33.3**	5.9	33.4	5.1	34.8	5.6
tgl	narrow	**42.9**	6.6	51.1	7.6	50.3	7.2
tha	broad	14.1	2.9	**13.0**	2.8	15.3	3.7
tur	broad	**52.8**	8.0	53.4	7.9	54.3	8.6
tur	narrow	57.9	8.4	53.9	7.9	**52.0**	8.5
vie	narrow	**3.0**	1.3	**3.0**	1.5	4.3	2.0
zul	broad	61.7	11.5	**10.4**	1.1	95.1	13.7

Table 16: This table shows the results for setting 1 (BS), setting 3 (BS-clean) and setting 7 (F2). These are the short models.

ISO396-3	BS WER	SIG WER	Transcription type	Model name SIG
eng (us)	53.00	**37.43**	broad	DP21
fra	7.20	**5.11**	broad	DeepSPIN20
ell	**9.80**	18.67	broad	IMS20
kat	0.30	**0.00**	broad	BS21/CL21
hin	5.60	**5.11**	broad	IMS20
jpn	6.60	**4.89**	narrow	DeepSPIN20
kor	26.70	**16.20**	narrow	CL21
vie	3.00	**0.89**	narrow	DeepSPIN20

Table 17: The table shows the WER results for the best of my short models compared with the SIGMORPHON 2020 and 2021 results. For each language the best score is reported no matter what year or what model from the SIGMORPHON task. My models and the SIGMORPHON models were trained on WikiPron data. However, the SIGMORPHON data was preprocessed in a slightly different way which means that the results are not directly comparable. All references for the SIGMORPHON models can be found in tables 4 and 3.

the uncleaned data performed even worse than its short equivalent.

The largest improvement from the short to the long models was for Russian [rus]. Russian had by far the largest dataset with more than 400,000 samples. It makes sense that for this large dataset, the model takes more time to extract all the relevant information.

Table 19 shows a comparison of my best long model compared with the best SIGMORPHON shared task result for different languages. For three of eight languages in table 19, my model reached a better or the same score. The SIGMORPHON Korean [kor] model performed quite a lot better than mine which can be explained as the SIGMORPHON model was not trained on the Korean logograms but those were converted to hangul characters which is a Latin alphabet representation of Korean. As explained in section 3.5, converting logographic scripts to alphabets can lead to better results as it reduces the vocabulary size of the model. My result for English is a lot worse as well. Generally, English (as well as German) seems to be a language that is relatively difficult to pronounce as the results are often a lot worse than those of other languages. In the present case, the difference might result from the fact that the SIGMORPHON model was an ensemble and it was only trained for English and therefore optimized to work for the English language.

ISO 639-3	Type WikiPron	WER BS	PER BS	WER BS-clean	PER BS-clean	WER F2	PER F2
cmn	broad	17.6	*3.9*	**17.0**	*4.0*	18.1	*4.4*
deu	broad	**37.1**	*4.8*	38.1	*5.0*	38.6	*5.9*
deu	narrow	**52.2**	*7.1*	56.9	*8.4*	55.9	*9.4*
ell	broad	**7.1**	*0.6*	9.0	*0.7*	9.1	*0.8*
eng us	broad	**50.7**	*9.5*	51.2	*10.2*	51.3	*10.4*
eng us	narrow	84.6	*31.4*	**84.2**	*33.1*	84.9	*32.3*
eng uk	broad	**45.5**	*8.6*	47.5	*9.2*	47.6	*9.7*
eng uk	narrow	**90.3**	*30.2*	94.8	*35.3*	93.7	*34.2*
eus	broad	21.2	*2.7*	**19.6**	*2.3*	21.7	*3.2*
fin	broad	**2.8**	*0.2*	3.3	*0.3*	8.9	*3.1*
fin	narrow	**3.2**	*0.3*	3.9	*0.4*	9.0	*3.5*
fra	broad	**5.3**	*0.7*	5.8	*0.8*	5.9	*0.8*
hin	narrow	**7.7**	*1.2*	8.4	*1.4*	8.4	*1.3*
hin	broad	**4.4**	*0.7*	6.4	*1.0*	6.3	*1.0*
ind	broad	37.9	*5.3*	**34.9**	*5.3*	39.3	*6.1*
ind	narrow	43.1	*5.4*	**43.0**	*5.6*	43.1	*5.6*
jpn	narrow	**6.5**	*0.6*	6.8	*0.6*	6.6	*0.8*
kat	broad	**0.0**	*0.0*	**0.0**	*0.0*	1.0	*0.8*
kor	narrow	**23.4**	*4.1*	25.3	*4.4*	25.8	*4.6*
mya	broad	**35.1**	*6.5*	36.0	*6.9*	88.0	*17.4*
rus	narrow	**1.9**	*0.2*	2.4	*0.3*	5.0	*1.5*
spa ca	broad	**1.1**	*0.1*	1.3	*0.1*	2.2	*0.7*
spa ca	narrow	2.3	*0.3*	**2.2**	*0.3*	2.8	*0.6*
spa la	broad	**1.4**	*0.1*	1.5	*0.2*	1.9	*0.6*
spa la	narrow	**2.6**	*0.3*	2.7	*0.3*	2.7	*0.4*
tgl	broad	**28.4**	*4.6*	31.2	*5.2*	33.9	*5.0*
tgl	narrow	**45.5**	*6.4*	47.3	*7.0*	48.6	*6.9*
tha	broad	12.5	*2.6*	**11.1**	*2.5*	12.1	*3.1*
tur	broad	50.6	*7.8*	52.3	*7.5*	**49.1**	*7.2*
tur	narrow	55.1	*7.6*	55.6	*8.3*	**54.8**	*8.0*
vie	narrow	**1.5**	*0.8*	1.6	*0.8*	2.6	*1.5*
zul	broad	65.9	*10.7*	**9.8**	*0.9*	91.4	*12.0*

Table 18: This table shows the results for setting 2 (BS), setting 4 (BS-clean) and setting 8 (F2). These are the long models

My model performs better on Modern Greek than the SIGMORPHON model. The SIGMORPHON model was trained on only 800 samples. I had more than 10,000 samples. The result aligns with cutting edge research, as **?** have found that 800 samples are not enough to train a decent G2P model. My results confirm this finding as my model trained on 10,000 samples of the same data was *considerably* better than the model trained on only 800 samples. The only difference between the training sets of the models was that I applied different preprocessing (see section 6.4).

ISO396-3	BS WER	SIG WER	Transcription type	Model name SIG
eng (us)	50.70	**37.43**	broad	DP21
fra	5.30	**5.11**	broad	DeepSPIN20
ell	**7.10**	18.67	broad	IMS20
kat	**0.00**	**0.00**	broad	BS21/CL21
hin	**4.40**	5.11	broad	IMS2
jpn	6.50	**4.89**	narrow	DeepSPIN2
kor	23.40	**16.20**	narrow	CL21
vie	1.50	**0.89**	narrow	DeepSPIN20

Table 19: The table shows the WER results for the best of my long models compared with the SIGMORPHON 2020 and 2021 results. For each language the best score is reported no matter what year or what model from the SIGMORPHON task. My models and the SIGMORPHON models were trained on WikiPron data. However, the SIGMORPHON data was preprocessed in a slightly different way which means that the results are not directly comparable. All references for the SIGMORPHON models can be found in tables 4 and 3.

6.6.3 Results: NWS tests

Table 20 presents the results for the tests on the NWS stories. The models do not perform very well on these stories in general. Concering the model types, there is no real pattern in how they perform. The clean models seem to perform a bit better than the other two types. Also, as the texts are really short, the performance of all models is often very similar.

Still, when compared to the coverage experiment in section 4.5, many G2P models perform quite a lot better than if we just used the WikiPron data as a look-up table. Although this is not surprising, it is good that we can confirm that those models are able to extract a lot of information that is not found in a simple look-up

table. What is surprsing is that there are a few languages link broad Hindi [hin] or Vietnamese [vie] where the WER of the G2P models is higher or very similar to the coverage experiment. Neither of these models performed particularly bad when tested on the WikiPron test set (WER long model Hindi: 4.4, Vietnamese: 1.5), in fact the results were very good. In addition, both training sets contain more than 10,000 samples. The most plausible explanation is probably that the transcription conventions used in the WikiPron data and the NWS stories are just very different.

ISO 396-3	**Type**	**Long**		**Short**		**Long clean**		**Short clean**		**F2 long**		**F2 short**	
		WER	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**
cmn	broad	95.0	72.5	100	73.8	94.1	45.2	86.1	40.5	93.1	39.2	**84.2**	36.4
deu	broad	74.1	26.0	76.9	28.0	72.2	24.5	**63.0**	23.4	72.0	20.8	65.4	19.8
deu	narrow	**50.9**	16.2	54.6	16.4	63.9	19.6	63.9	20	57.0	16.6	59.8	17.9
ell	broad	78.9	32.8	**75.4**	31.1	**75.4**	29.2	**75.4**	29.5	77.0	17.3	77.0	18.3
eng us	broad	**66.4**	22.3	87.6	32.5	84.1	27	87.6	30.4	83.9	20.9	92.0	24.7
eng us	narrow	**94.7**	47.5	97.3	44.5	97.3	51.8	97.3	47.9	97.3	45.1	97.3	44.1
eus	broad	36.8	4.2	48.3	5.9	36.8	4.8	**34.5**	4.7	34.9	5.1	34.9	4.8
fra	broad	37.0	15.6	45.4	17.7	**34.3**	14.3	48.1	19.3	34.6	11.6	48.6	15.5
hin	narrow	91.1	62.6	91.1	62.6	90.3	58.5	90.3	58.5	**90.2**	29.9	**90.2**	30.4
hin	broad	96.0	56.1	96.0	55.8	95.2	57.8	95.2	57.8	**95.1**	47.0	**95.1**	47.0
ind	broad	**81.5**	30.8	**81.5**	27.9	**81.5**	29.7	**81.5**	30.1	82.2	17.8	82.2	19.4
ind	narrow	85.2	33.0	**82.4**	32.2	**82.4**	31.3	**82.4**	31.0	86.0	23.8	86.0	23.6
kat	broad	64.2	14.6	64.2	14.6	**58.2**	11.5	**58.2**	11.5	59.1	7.5	59.1	7.8
kor	narrow	100.0	52.5	100.0	52.7	100.0	52.5	100.0	51.4	100.0	100.0	100.0	100.0
mya	broad	97.5	34.4	97.5	36.5	**95.0**	21.3	**95.0**	23.2	97.4	29.4	97.4	35.8
rus	narrow	91.6	37.2	91.6	37.3	**89.5**	36.3	91.6	37.6	91.5	35.7	91.5	37.1
spa ca	broad	**27.8**	5.1	28.9	5.4	**27.8**	5.1	**27.8**	5.1	32.3	5.4	32.3	5.4
spa ca	narrow	56.7	17.3	**53.6**	16.5	58.8	16.2	54.6	16.2	65.6	17.3	54.2	16.1
tha	broad	99.4	62.3	99.4	62.2	56.9	19.0	56.9	19.1	57.2	16.3	**56.6**	16.4
tur	broad	53.8	7.8	63.1	10.1	**46.2**	6.7	55.4	8.7	56.2	9.1	64.1	9.2
tur	narrow	76.9	15.6	**66.2**	11.6	76.9	11.9	75.4	13.7	75.0	11.9	68.8	12.1
vie	narrow	100.0	99.8	100.0	99.8	**79.5**	44.2	**79.5**	44.2	80.2	36.1	80.2	36.1

Table 20: In this table I present the results for all models when they predicted the NWS stories. The models are explained in section 6.3 where I explain the different training settings. 'Long' corresponds to setting 2, 'short' to setting 1, 'Long clean' to setting 4, 'Short clean' to setting 3, 'F2 long' to setting 8 and 'F2 short' to setting 7.

6.6.4 Results: overall

All models of one type show a similar performance no matter how long they were trained. The most notable observation is that the difference between the performance of the long and the short models is rather small. For some languages, the best short model performed even better than the long one or the results were the same. This was the case for narrow German [deu], narrow US English [eng us], narrow UK English [eng uk], broad Goizueta Basque [eus], broad Burmese [mya], narrow Tagalog [tgl] and narrow Turkish [tur]. I find it surprising that most of these languages are the narrow variants as those are more detailed. I would expect a model trained for longer to catch more details. However, none of the scores were really low, the lowest being 19.4 for Basque. Additionally, the differences between the short and long models for these languages are never larger than 2.8 percentage points.

The feature models did not show any relevant improvement compared to the models trained without features. They did not worsen the results but performed very similarly. When the long feature models are compared to the short ones, the models trained for longer reached a better performance for almost all languages. As the input data is more complex because I added the features, it might be worth exploring even longer training times. However, also the long trained models without features reached a better performance than the short models without features. This means that it could be possible that the feature models would still not outperform the models trained without features if both model types were trained for longer.

7 Conclusion

As a first step of this thesis I created an overview of what phonetic data is available in which languages. The overwhelming majority of available data is word lists. But in general, there is quite a lot of data that I could use for my experiments. The WikiPron data keeps growing and even if it is not cleaned, the models were able to produce good results for quite many languages. One thing I noticed about the data is that phonetic text (given it is written in IPAlpha) is more complex and has more characters than most alphabets. The case for Zulu also showed that the tones (which are represented as diacritics in Zulu) add to that complexity. The comparison of the NWS transcriptions and the WikiPron test sets showed that phonetic transcriptions are not as standardized as typical writing systems. It might be worth looking into why exactly the models performance was so bad on the NWS short stories.

As G2P conversion is a well-known seq2seq task, there exist a lot of models and architectures that can be used. My experiments with the feature models showed that it is possible to manipulate the input and the model does still perform well. All in all, there were some good results produced for a large collection of many languages. Some of my models outperformed existing state-of-the-art models, for some there was no comparison available. It would certainly be good to train more G2P models on more languages such that my results can be compared.

An important lesson I learnt from this thesis is the overwhelming complexity of computational representations of different scripts and the IPAlpha. While I did not read a lot about this topic in present research, it was very important for my multilingual processing. Multilingualism in general presents a huge challenge. Starting from transcription conventions that are not always cross-linguistically applicable to machine learning models that are very carefully designed to work for one language only, it seems that many steps of creating G2P models are not made for working with a huge multilingual datasets. Research about linguistic diversity and language comparisons in NLP datasets is certainly a good start to pave the way for more work with multilingual corpora.

Future work

Non surprisingly, apart from many exciting things I *could* do, there are many others that would have gone beyond the scope of this thesis. I would like to list a few entry points on where future research could start.

- **Data preprocessing**: ? cleaned broad transcriptions for Bulgarian and replaced all allophones by their standard phoneme. I also detected allophones in some broad transcriptions of some language. Replacing allophones by their respective phoneme could further improve model quality by having consistent broad transcriptions.
- **Phonetic features**: Although my feature models did not outperform the other models on many languages, they did not perform a lot worse. In this thesis, I tried to add the features by manipulating the input to the model. It might also be possible to chose a different model architecture such that the features can be passed more explicitly to the model. I am sure that it is worth digging deeper into phonetic features and how we can use them to train better G2P models.
- **Low resource languages**: There are a lot of languages where only very little data is available. I did not deal with those in this thesis but it is technically possible to use transfer learning or other techniques to make use of those languages. Low resource languages still are a challenge.

References

A Tables

Iso 639-3	Name-WALS	Family-WALS
abk	Abkhaz	Northwest Caucasian
amp	Alamblak	Sepik
aey	Amele	Trans-New Guinea
apu	Apurinã	Arawakan
bmi	Bagirmi	Central Sudanic
bsn	Barasano	Tucanoan
gry	Grebo	Niger-Congo
eus	Basque	Basque
ape	Arapesh (Mountain)	Torricelli
bsk	Burushaski	Burushaski
ram	Canela-Krahô	Macro-Ge
tzm	Berber (Middle Atlas)	Afro-Asiatic
cha	Chamorro	Austronesian
ckt	Chukchi	Chukotko-Kamchatkan
zoc	Zoque (Copainalá)	Mixe-Zoque
dgz	Daga	Dagan
hae	Oromo (Harar)	Afro-Asiatic
arz	Arabic (Egyptian)	Afro-Asiatic
fij	Fijian	Austronesian
fin	Finnish	Uralic
gni	Gooniyandi	Bunuban
khk	Khalkha	Altaic
hau	Hausa	Afro-Asiatic
heb	Hebrew (Modern)	Afro-Asiatic
hin	Hindi	Indo-European
hix	Hixkaryana	Cariban
hnj	Hmong Njua	Hmong-Mien
qvi	Quechua (Imbabura)	Quechuan
imn	Imonda	Border
ind	Indonesian	Austronesian
kal	Greenlandic (West)	Eskimo-Aleut
kyh	Karok	Karok
gyd	Kayardild	Tangkic
kio	Kiowa	Kiowa-Tanoan
cku	Koasati	Muskogean
kor	Korean	Korean
ses	Koyraboro Senni	Songhay
kgo	Krongo	Kadu
kut	Kutenai	Kutenai
lkt	Lakhota	Siouan
laj	Lango	Eastern Sudanic
lvk	Lavukaleve	Solomons East Papuan
lez	Lezgian	Nakh-Daghestanian
dni	Dani (Lower Grand Valley)	Trans-New Guinea