Linguistically-augmented approaches to G2P conversion

B139453

August 2019

Master of Science

Speech and Language Processing

School of Philosophy, Psychology and Language Sciences

The University of Edinburgh

Word count: 7999

Abstract

This dissertation explores the approach of automatically generating morphological boundaries to improve models trained for grapheme-to-phoneme conversion. This work first compares the output of grapheme-to-phoneme and morpheme-to-phoneme models, establishing that morphological information improves the output phoneme transcription accuracy. Then, this work investigates automatically generating a morphological segmentation with various grapheme-to-morpheme models: Byte-Pair Encodings (BPE) (Sennrich 2015), LMVR (Ataman 2017), and seeded BPE, new to this paper. The results of models trained with automatically segmented input are compared to expert transcriptions from CSTR's *Unisyn.* These predicted transcriptions are evaluated for G2P improvement in WER and PER. The findings show that morphological information improves G2P significantly, and that unsupervised methods of data collection still lead to an increased G2P accuracy. These methods will be tested on three data-sets, two in English - *Unisyn* and *CMUDict* as well as a novel Cornish-language machine-readable dictionary introduced in this paper: *LexiCorn.*

Keywords: ***grapheme-to-phoneme conversion; morphological segmentation; sequence-to-sequence models; neural machine translation; Cornish-language***

Acknowledgements

I would like to express my sincere thanks to Jason, Caela and Korin, without whom this paper would still be on the chopping floor.
Also to Tom, for his endless good company and friendship throughout the year - making many impossible tasks not only possible but even enjoyable.
Finally, to Danielle, for her patience and love while I have been dedicating my time to this most fulfilling of challenges.

Contents

1 Introduction

1.1 Grapheme-to-phoneme conversion

A problem in speech synthesis is that the relationship between the letters of a word (graphemes) and the sounds of which it is comprised (phonemes) is not monotonic. Letter to sound is not fully mappable. For example, the word `cough` contains five graphemes - but only three phonemes: /kofː/. Additionally, a sequence of graphemes can map to different phonemes, depending on the word. The `cou` of `cougar`: /ku/ contrasts with that of `cough`. Correct word pronunciation is a fundamental part of a text-to-speech (TTS) system, and commonly, a static pronunciation dictionary which maps from grapheme-to-phoneme (G2P) is employed. However, these dictionaries are not all-encompassing, and when an out-of-vocabulary (OOV) word is encountered, a G2P model is required to make a prediction. A robust and accurate G2P model, which is able to deal with the inconsistencies of natural language, is therefore desirable - in order to ensure that pronunciations of OOV words are correct.

This paper explores methods for improving G2P accuracy by introducing linguistic information, including part-of-speech (POS), and morphological (Mo) boundary markings to the input. allowing a bi-directional LSTM (BiLSTM) to disambiguate examples that are otherwise confusing.

1.2 Motivation

For augmented G2P conversion to be considered as a viable and consistent improvement over standard G2P, across languages and without requiring expensive data-collection; three questions need to be answered.

- First, can it be shown that including linguistic information improves the accuracy of G2P?
- Second, if including linguistic information improves G2P, the question remains, can this information be generated automatically; or with a small amount of human supervision? Or does it need to be gold-standard hand-labelled data to be useful?
- Third, if linguistic information can be automatically generated, does it provide a consistent improvement over G2P for different data-sets? Or across languages? If the answer is yes, then a cheap and robust preprocessing stage to consistently improve G2P has been revealed.

1.3 Structure

This dissertation addresses these issues in six following chapters. It begins with Chapter 2, a section on the background information necessary to understand the data, (Chapter 3), hypotheses (Chapter 4) and methodology (Chapter 5) of the subsequent experiments performed in Chapter 6. Chapter 7 analyses the results of the experiments and Chapter 8 concludes the paper, positing further areas of study and outlining a direction for further exploration.

2 Background

2.1 Approaches to G2P

G2P is most commonly handled by a look-up dictionary, which contains a hand-written pronunciation for a word to be synthesised. When a look-up dictionary fails to find the desired word, due to it being OOV, there are a variety of possible solutions. The most simple of these is to have handwritten letter-to-sound rules which dictate how a given sequence of graphemes should be pronounced. This can be effective, especially for languages like Italian, where the relationship between letter and sound is monotonic, and therefore mappable. For example, the Italian digraph `ch` is always pronounced /k/ whereas in English it can vary from /tʃ/ as in `chap` and /k/ as in `chasm`. As a result, English requires a more complex system for automatic G2P conversion. The earliest G2P approaches used finite state transducers (FST) based on hand-written rules (Kaplan and Kay 1994). However, knowledge-based systems such as these remain limited, and struggle to handle exceptions. Subsequently, data-driven approaches to the FST solution, such as *Phonetisaurus* (Novak et al. 2012) which uses Weighted FSTs. As these are data-driven, they can be generated for languages without the need for expert rules, given enough training data. Modern data-driven approaches remain close to state-of-the-art, for example joint-sequence models (Bisani and Ney, 2008) and subsequently various neural network (NN) models such as Rao et al's BDLSTM (2015), and Yolchuyeva et al.'s CNN-BDLSTM (2019).

2.2 Linguistic information for ambiguity resolution

We identify two kinds of ambiguity that are common in G2P. The first is at character-level, where a given character sequence may map to different phoneme sequences in different contexts, as the earlier `chap/chasm` distinction showed. The second is word-level ambiguity - homographs can have different pronunciations. For example `desert` has different pronunciations depending on whether it is referring to an arid climate /dezət/, or the act of leaving /dɪzəːt/. In order to resolve ambiguities from G2P, it is necessary to source other information which distinguishes otherwise identical words. POS can be useful to distinguish homographs. In our `desert`, case 1 is a noun, case 2, a verb.

However in the case of `unionise`, both versions are verbs, and so other information must be used. Morphology considers the smallest meaningful units which comprise words; morphemes. These can be defined as either free or bound. Free morphemes are able to exist as entire words alone, and are often roots. A root morpheme expresses a self-standing concept, such as `cat`. Bound morphemes are combined with a free morpheme as an affix, to make a morphologically complex word, and cannot stand alone.

By marking morphology of words, `union-ise` /ju.niən.ʌɪz/ and `un-ion-ise` /ən.ʌɪən.ʌɪz/ can be distinguished. Crucially, no other information such as POS or stress-marking disambiguate this case. Demberg et al. (2007) use morphological prediction to improve German-language G2P. They found that it was useful not only for disambiguation at word level, but also at character level, for compound words (word with more than one root morpheme) such as `loophole`, where the correct German pronunciation is /luːphəʊl/ rather than /lufəʊl/. We reason that the same logic applies to English, given the word `pothole`. A morphological parse of the word would reveal that there are two root morphemes, and that the word is therefore a compound, necessitating that the syllabic and phonetic transcription generated is not /[pɒ.θəʊl]/ but /pɒt.həʊl/. Due to the frequent co-occurrence of `th` and the transcription /θ/, it is feasible that data-driven approaches, trained only on a standard G2P lexicon, might not make this distinction at test time.

The question remains, if morphological information indeed is useful, what sort of morphological information is sufficient to improve G2P accuracy? Richmond et al. (2010) used a toolkit implementing an FST: PC-KIMMO (Antworth 1991) that transduced a word's derivation by finding the most probable path through a lattice. Bikmetova (2018) found this solution to be problematic as when there are no legal paths through the lattice (as the root morpheme is OOV) it is impossible to segment the word using an FST-based approach.

In its place, Bikmetova found that morphological decomposition could be predicted by a sequence-to-sequence neural network trained on hand-labelled data, substantially outperforming the FST-based approach used by Richmond et al. with PC-KIMMO by 16% WER. By framing the relationship between source and target as a machine translation

(MT) task, and by training a neural network to map between graphemes and the same grapheme with labelled morphological boundaries - a greater level of accuracy was attained without requiring new entries to be created and hand-written for OOV words.

2.3 Sub-word segmentation

Morphological information can be used for grapheme disambiguation, but is not included within most lexica. Therefore it needs to be gathered automatically. Here we discuss three methods which can generate quasi-morphological boundaries.

2.3.1 BPE

Sennrich et al. (2015) proposed a novel approach to tackling OOV words with sub-word segmentation, allowing for open-vocabulary machine translation. By implementing Gage's (1994) system of byte-pair encoding (BPE).

The motivation behind BPE is straightforward: That some new words in a source language that do not have known translations can be translated in composite parts. For example, Sennrich et al. found that by segmenting the German compound word `Sonnesystem` into `Sonne+system`, they could successfully predict the English:`Solar system`. For the task of morphological segmentation, the same logic applies. If an OOV word is encountered, it is hypothesised that by breaking it down into sub-word segments, the transformations for segments can be learned and applied - potentially allowing for more accurate segmentation of words that would otherwise remain unseen in training data. This provides some of the same benefit as morphological segmentation, but at a fraction of the cost. Demberg et al. (2007) propose that morphological information is particularly useful when data is scarcer, and as BPE is robust to small data quantities, is appears promising. Additionally, Sennrich found that BPE segmentation sometimes mirrored a word's morphology, as in the case of `Sonnesystem` above. As BPE segments purely on frequency; common morphemes, particularly affixes such as `anti-, de-, -able` and `-ed` (Honig 2000) are themselves therefore more likely to be compressed together as a segment by the algorithm. Of course, there are no constraints on BPE, so this is not always the case, but even some sub-word structure may lead to benefit. Critically, BPE is agnostic about linguistic structure at sub-word level, any structure generated that mirrors morphology remains solely due to frequency.

The process of the BPE algorithm is simple. For a given data-set, the most common pair of bytes within that data is replaced with a byte that does not occur in the data-set. This is applied iteratively; until either a compression threshold hyperparameter (minimum number of unique tokens) or a transformation threshold hyperparameter (number of replacement operations) is reached.

Given a toy corpus of data, comprising three words: `bananas, gonads, diagonals` BPE applies as follows: First, a special end-of-word token ('\w') is appended where appropriate, then the algorithm iterates, as shown in the table below.

Table 1: 4 merge combinations for BPE vocabulary reduction

Step	Input	Output	Corpus
1	n+a	na	b+a+na+na+s+\w, g+o+na+d+s+\w, d+i+a+g+o+na+l+s+\w
2	s+\w	s\w	b+a+na+na+s\w, g+o+na+d+s\w, d+i+a+g+o+na+l+s\w
3	g+o	go	b+a+na+na+\w, go+na+d+\w, d+i+a+go+na+l
4	go+na	gona	b+a+na+na+s\w, gona+d+s\w, d+i+a+gona+l+s\w

Here, after four steps of the algorithm, we already see that the plural-marking suffix, `-s` has been separated by a boundary. Morphology is approximated. Other segmentations which do not respect the morphology of the words are also made, such as the splitting of `gonads` into `gona+d+s`.

2.3.2 Seeded BPE

BPE was designed as a data compression algorithm, not for morphological segmentation. We propose that one way to improve the likelihood of BPE segmentations matching the morphology of a given language is to create a seeding corpus. Normally, the BPE algorithm would be trained on the corpus it segments, as this allows for optimal segmentation of the data for compression. By creating a corpus made up solely of affixes, and training the BPE algorithm on it, when the algorithm is applied to the main corpus, the segmentations will reflect the frequency counts of the characters in the seeding corpus. This is a very cheap, and easy way to force BPE to reflect a language's morphology, without needing annotation - only a small collection of common morphemes. For example, if our corpus for segmentation were:

[`catty\w, catatonic\w, catastrophic\w, muscat\w, catches\w, chico\w`]

and BPE was trained on the same set of data, after five iterations of the algorithm would lead to segmentations of:

[`m+u+s+cat+\w, cat+at+o+n+ic\w, cat+a+s+t+r+o+p+hic\w, cat+t+y+\w, cat+c+h+e+s+\w, c+hic+o+\w`.]

This fails to respect the suffixes `-ic, -es, -ty` on `catastrophic, catches` and `catty` respectively. However, training BPE on this set:

[`aty\w, ity\w, hic\w, bic\w, nes\w, zes\w`]

(giving final merges of `ty\w, ic\w, s`) and applying this trained algorithm on the original corpus for segmentation would lead to the more accurate segmentations:

[`m+u+s+c+a+t, c+a+t+a+t+o+n+ic, c+a+t+a+s+t+r+o+p+h+ic, c+a+t+ty, c+a+t+c+h+e+s, c+h+i+c+o`]

This is closer to the true morphology of the words. In terms of input, only a small collection of carefully chosen seeding words to deliver determined improvements is required. However, there is the risk that if a work contains a sequence of characters that typically forms an affix, such as `pretty`, that it would be incorrectly segmented into `pret+ty`.

2.3.3 LMVR

Ataman et al. (2017) further resolved this issue, presenting a refined version of BPE - Linguistically Motivated Vocabulary Reduction (LMVR). LMVR avoids breaking morphological structures during segmentation by combining an unsupervised HMM-based morphological parser; *Morfessor FlatCat* (Grönroos et al., 2014) with the BPE segmentation algorithm. The end result is vocabulary reduction that retained linguistic knowledge, and in turn, better performance than BPE for MT tasks. This was especially apparent when dealing with morphologically complex OOV tokens. In Ataman's experiments translating Turkish to English, they highlight two Turkish words: `ağlarını` (*the nets*) and `ağlamayacak` (*would not be crying*). BPE segments the words into `ağ+larını` and `ağ+lamayacak`. In reality, the two words have different root morphemes, `ağ` (*net*) and `ağla` (*cry*). LMVR does not bisect the root `ağla`, and thus the predictions match the reference translation. For G2P conversion then, we posit that LMVR may be a solution that allows the suffix `-ty` to be predicted in cases like `catty`, and yet not segmented incorrectly for words such as `pretty`. This preprocessing stage is more complex than BPE, and requires more data to train the FlatCat stage accurately, so there is a computational trade-off. Performing LMVR segmentation on a corpus of 100000 words takes 800 seconds, compared to 1.7 seconds needed to BPE segment an equivalent corpus.

2.4 Neural network architectures

2.4.1 Feed-forward Neural Networks

To address neural G2P, first the role of a NN must be explained. The fundamental function of a NN is to approximate a relationship between an input x and an output y. A single unit of a NN applies a single affine transform on its input value to return a changed output (see Rosenblatt 1958). By combining many of these units into layers, and stacking layers, a NN becomes a universal function approximator; thereby able to learn and model the relationship between any input and subsequent output. Given an input of characters for which we desire a phonetic transcription:

$$(g_1 ... g_i ..., g_n)$$

the probability of the transcription of a character p can be expressed as

$$P(p_i | p_{i-1}, g_i)$$

as it is a function of the source grapheme (g) and a previous n-character window (in this case, n=1). However, basic feed-forward NNs are limited. They are only able to look at an arbitrary window of input. When dealing with natural language tasks, it is sometimes necessary to consider a larger window of context over the whole word, Therefore a fixed window is insufficient.

An example of this can be found by comparing the phonetic transcription of the final syllable of `Ossian` /siən/ with that of `Russian` /ʃən/.

To this end, recurrent systems, such as the RNN and LSTM were developed.

2.4.2 Recurrent neural networks

With a recurrent neural network (RNN), at a given time step, the RNN unit looks not only at the new input, but also the hidden state of the network at the previous time step. As this is performed iteratively, this additional input is effectively a compressed representation of the entire history of the input sequence; which resolves the issue of long-range dependencies highlighted earlier.

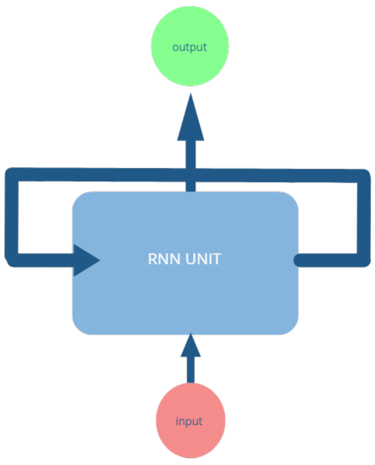

Figure 1: an RNN

When the RNN is unfolded, it becomes clear how information from each time step is 'passed along' from one hidden state, $h_{(t-1)}$ to the next, $h_{(t)}$, and therefore how these long range dependencies can be modelled.

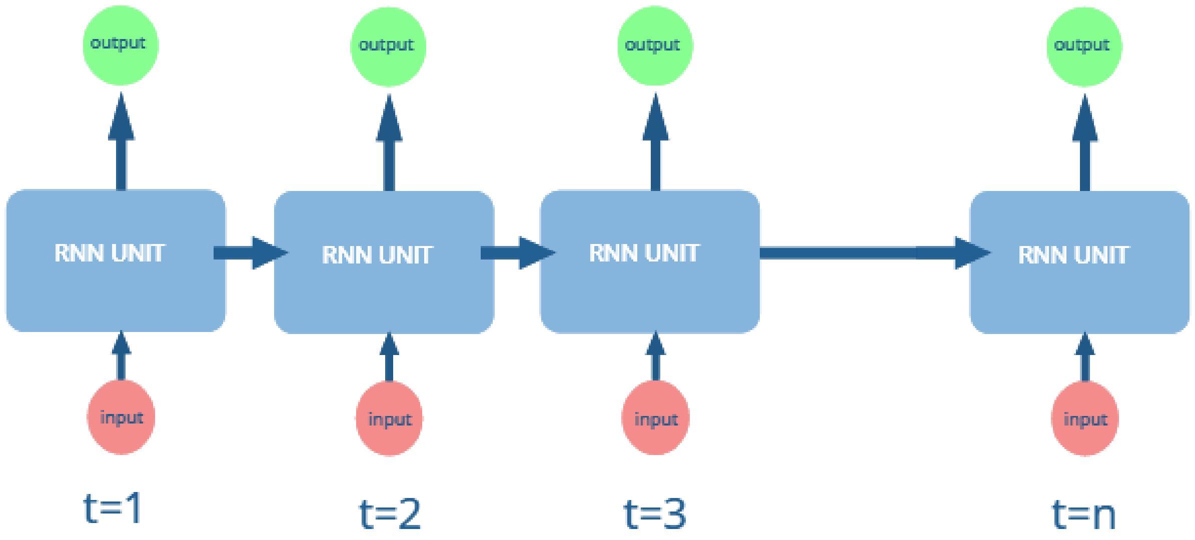

Figure 2: an unrolled RNN

However, for an RNN, the first element of an n-length sequence by necessity is compressed more times than the element immediately previous. If we are at the sixth character of a sequence, information for the first character has been compressed five times, rather than only the once for the fifth character. As compression leads to data loss, this still a potential problem for the Russian/Ossian example. Additionally, RNNs suffer from the *vanishing gradient problem*. As the number of layers in a network increases, it becomes harder to train the early layers of the network using back-propagation, due to the diminishing size of the returned loss. This is problematic as early layers are responsible for basic pattern identification, and so if they are inaccurate, error is amplified by further layers.

To resolve these issues the Long Short Term Memory cell (LSTM) was developed (Hochreiter and Schmidhuber, 1997). These networks retain the recurrent connections of an RNN, but for each cell of the LSTM, there are gates within the recurrent unit which allow for stored information to be selectively carried over while it remains useful, or forgotten if is no longer needed.

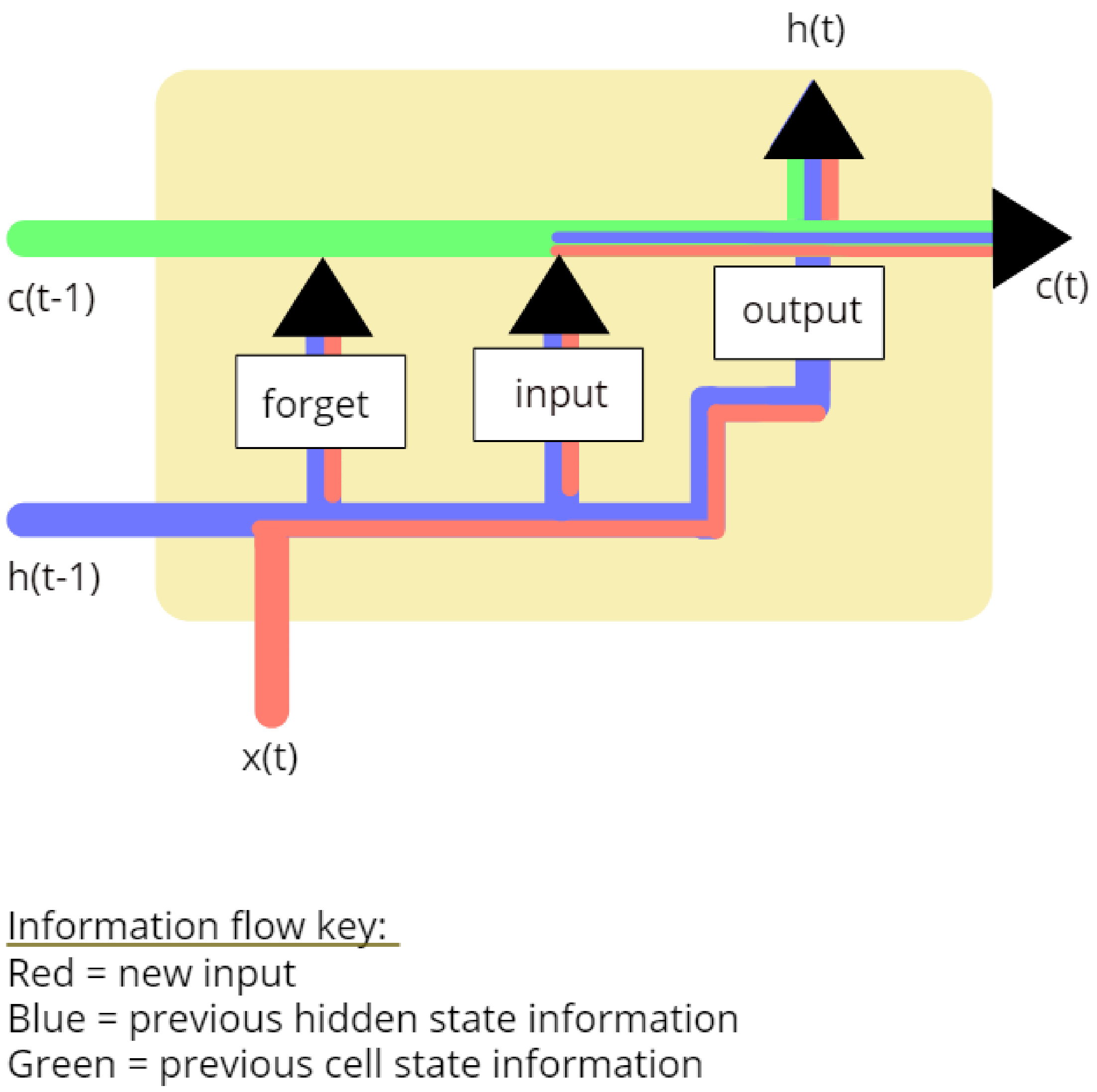
h(t)
c(t-1)
c(t)
forget
input
output
h(t-1)
x(t)
Information flow key:
Red = new input
Blue = previous hidden state information
Green = previous cell state information

Figure 3: The interior workings of an LSTM cell

The above figure demonstrates three key gates in the LSTM cell, (adapted from Olah, 2015)

- The forget gate looks at the previous hidden state, $h_{(t-1)}$ and the new input, $x_{(t)}$, and uses this information to make a decision about which elements of the cell state $c_{(t-1)}$are no longer needed.

- The input gate looks at information from the previous hidden state and new input and updates the cell state with relevant information. This is also where information within the cell state that was earmarked for deletion in the forget gate is actually removed, giving us the current cell state $c_{(t)}$.
- The output gate decides what information should be output by the LSTM to become the new hidden state $h_{(t)}$. The cell state is moderated by the information from the input gate, so that only the desired information is passed forward to the next unit.

The key innovation that sets LSTMs apart from RNNs is that of the cell state. Because the cell state is only ever linearly transformed, it does not get affected by the vanishing gradient problem, and therefore early layers can be trained more accurately. Additionally, the cell state is able to selectively store relevant information without the loss that comes from multiple-compression; thus resolving the Ossian/Russian problem.

2.5 Mapping sequences of arbitrary length

2.5.1 Sequence-to-sequence networks

Sequence-to-sequence models (seq2seq), introduced by Sutskever et. al (2014) to resolve an issue common to all models discussed above. These models can only map between an input and output of fixed sequence length. As NMT needs to map between sequences of variable length (and this is true for G2P as well, there is not a one-to-one relationship between grapheme and phoneme), fixed sequence length is problematic. The solution introduced by Sutskever involves using one NN with recurrent units as an *encoder* and a second as a *decoder*. The encoder takes the input string and encodes it as the *hidden vector*. This is a vector of the final hidden states from the encoder, and is an abstract representation of the entire input; encoded in such a way that it helps the decoder make useful and accurate predictions. The decoder takes the hidden vector as its own initial hidden state - before then making a prediction on the first element of the translation. Crucially, because the entire encoded representation of the input is accessible, any number of input characters (and in any position) can be used in prediction of the output. Allowing for arbitrary length sequences is particularly useful for languages like English and French where letter-to-sound is less mappable, however, it introduces a new issue. It is difficult to establish exactly which elements of the embedded input align to segments of the embedded decoding. All of the encoded input is represented as a single context vector, which has to provide all the information needed for decoding. In the French: `oiseaux` mapping to /wazo/, to make a correct G2P prediction, the model needs to learn which graphemes align to a given phone, and it has to do this from a highly compressed input sequence.

2.5.2 Attention

To resolve this issue, Bahdanhau et al. (2015) introduce the concept of attention. Rather than creating a single context vector, they proposed an additional attentional layer which calculates how much of a contribution an embedded unit makes in decoding the output unit via a softmax distribution. In the case of translating from English to French they show how attention provides a soft alignment between the two sequences.

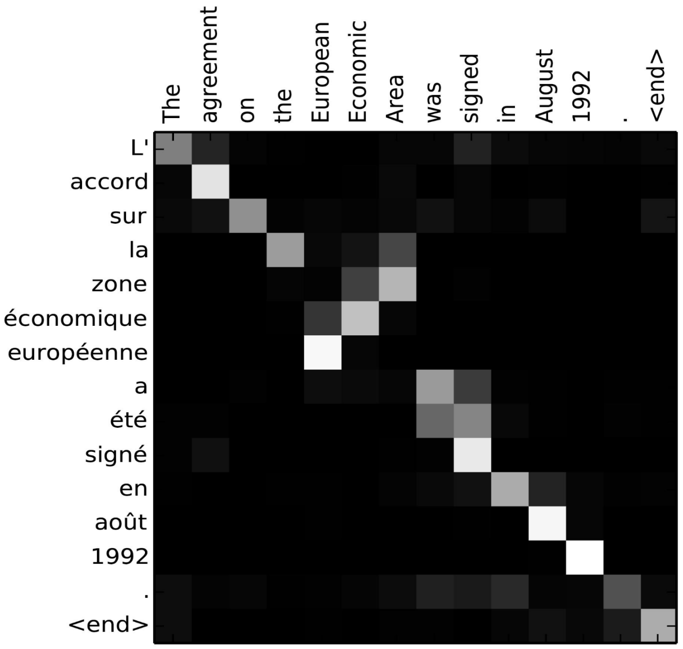

Figure 4: Figure 3(a) from Bahdanau et al. (2015, pp.6)

Despite `European Economic` and `économique européen` being in reversed order, the attentional map shows how both words of English are used to translate the French. For the task of G2P conversion, we can see how letters map to sounds with the French word `oiseaux` in the figure below.

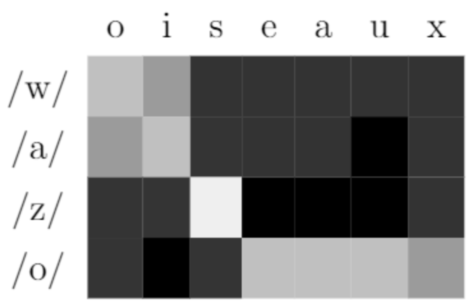

Figure 5: Attentional alignments between oiseaux and /wazo/

We can see how attention creates a soft alignment between phoneme /o/ and the graphemes `eaux`.

3 Data

By using different lexica in our experiments, we can assess the effect of segmentation methods on a wider variety of corpora of different qualities and languages. This section details three sources from which we gather data.

3.1 Unisyn

Unisyn (Fitt 2000), developed at the University of Edinburgh's Centre for Speech and Technology Research, is a database which contains lexical entries along with other linguistic information, such as POS, morpheme boundaries and stress markings. It is also dialect-independent, with vowels represented by key symbols (metaphones) based on Wells' (1982) lexical sets. Vowels are classified into sets such as the BATH vowel (GenAm /(æ)/, RP /ɑ/. The phonetic transcriptions can then be converted to a chosen dialect via simple transformations. An entry within *Unisyn* is formatted as such:

```
undo :  :  VB / VBP : < uh n <.  d * uu  :  <un<do :  2015
```

By including information such as part of speech (POS), morphological segmentation, and a generalised phonetic transcription - there exists the potential to augment a model trained for G2P with this information in order to improve its accuracy, and therefore the database provides an ideal source for our experiments. In this paper, as we are not aiming to look at G2P in a particular dialect of English, we keep *Unisyn* in its base form. This lexicon provides the baseline for the experiments in this paper - allowing for evaluation of automatic segmentation methods' performance against both G2P and gold-standard Mo2P.

The *Unisyn* lexicon contains 119356 entries. These are split into training, test and validation subsets with a ratio of 75:20:5. Accordingly, the number of entries in each set is seen in Table 2.

Table 2: Dataset splits for *Unisyn*

Subset	**Entry count**
Training	89516
Test	22872
Validation	5968

This splitting into sets was performed twice for *Unisyn*. For the first set - 'random' there is no supervision in the splitting and a true random number generator was used to separate the data. The second set - 'disjoint' used a copy of the original lexicon but in this case, the set separation was supervised in such a way that no subset shared root morphemes with another subset. In terms of set theory, they are disjoint from each other. The motivation for creating a disjoint set is to investigate the effects on model prediction when it is required to 'translate' root morphemes not previously encountered during training. For example, in our experiments, the root morpheme `{post}` is only found in the training set, as such all derivations such as `{lamp}-{post}, {post}-{man}, and {post}>ing>` are also only observed in the training set. However, `{compost}` does not share the same root morpheme, and so is not bound by this restriction, and is found in the test set.

From these split entries, multiple preprocessing steps were performed (see Appendix for scripts). Relevant information was extracted from entry fields and create index-linked lists of the POS, graphemes and phonemes, both grapheme and phoneme lists exist with and without morphological annotations.

3.2 CMUDict

CMUDict 0.7b (CMU Pronunciation Dictionary, 2014) is an American English lexicon, and one of the largest available for free use. It is more traditional than Unisyn insofar as it maps only between graphemes and phonemes but contains a greater number of entries. However, the lexicon is inconsistent - for example in the following excerpt:

```
ATHEISM AH0 TH AY1 S AH0 M
ATHEIST EY1 TH IY0 AH0 S T
```

At train time, this means the model is faced with competing versions to learn. If a transcription is variable in natural language, this is less of an issue, but if the model learns from incorrect data; then even a model with otherwise perfect mapping of G2P would make transcription errors. In this case, the error is likely due to a team of lexicographers working together to compile the dictionary, who may have had different interpretations of how to transcribe a given word into General American English. Additionally, it contains a larger number of names than *Unisyn*, many of which originate for languages other than English. This in turn means that they are more likely to have more unpredictable pronunciations. However, as an open-source pronunciation dictionary, it is widely used within other research papers on G2P, therefore, we include it in order to have our models and results be comparable to other papers. *CMUDict* (133779 entries) was processed in the same way as *Unisyn* with the same data-set split ratio, but due to the nature of this lexicon, only grapheme and phoneme lists were created and it was not possible to create a disjoint set, as that required morphological labels. The number of entries in train, test and validations sets are as follows:

Table 3: Data-set splits for *CMUDict*

Subset	**Entry count**
Training	100334
Test	26756
Validation	6689

3.3 LexiCorn

Cornish (Kernewek) is a Brythonic language, closely related to Welsh (from which it diverged after the Norman Conquest) spoken by approximately 3500 speakers worldwide. with 2000 speakers considered to have Level C proficiency. (Kominek 2009). According to the 2011 UK census, there are 600 people who use it as their main language in the UK (Office for National Statistics, 2011). In spite of low usage there is a defined written standard; Furv Skrifys Savonek (FSS). As a result, it is one of the least-spoken languages with defined orthographic conventions, and is the least-spoken official language of the UK.

Despite this, there is a growing movement for the usage and revival of the language in Cornwall. However, there is, at time of writing, no machine-readable lexicon suitable for Cornish-language G2P conversion. As language technology becomes more prevalent, this acts as a restriction as Cornish speakers are unable to use their language of choice. Introduced here, *LexiCorn* is precisely that - a machine-readable G2P lexicon encoded in UTF-8, with phonetic transcriptions in X-SAMPA (Wells 1995) and IPA. Graphemes follow the FSS standard. We have created this lexicon for two purposes. First and foremost, *LexiCorn* provides a low-resource dataset for G2P conversion in a different language, in order to compare to our experiments in English. Secondly, in publishing this corpus free-to-use, we hope to allow future work to be done on creating speech technology for Cornish.

The lexicon we have created comprises of 8931 Lexical entries which are based on the most recent dictionary from Akademi Kernewek (2018) with automatically generated X-SAMPA transcriptions derived from the provided IPA. A link to this author's public-access GitHub repository containing *LexiCorn* can be found in the Appendix, and we will release it on an MIT Open-Source License. Further revisions and expansions remain as a task for the future, however, due to time constraints. The *LexiCorn* data was processed with a data-split ratio of 75:10:15 in order to maximise the amount of data available for training and validation. This decision was made due to the small size of the corpus. Again, only grapheme and phoneme lists were created, with no disjoint sets, following the same reasoning as with *CMUDict*. The number of entries in the subsets of *LexiCorn* are detailed below:

Table 4: Data-set splits for LexiCorn

Subset	**Entry count**
Training	6697
Test	894
Validation	1340

Additionally, a subset of *Unisyn* was made - *Uni_0.1* with the same number of entries and split in the same ratio as *LexiCorn* in order to compare Cornish and English G2P on comparable data-sets.

4 Hypotheses

Based on the above research, we draw the following hypotheses:

H1: Training a BiLSTM with gold-standard morphologically segmented graphemes from *Unisyn* will lead to an increase in accuracy compared to an identical model trained only on corresponding graphemes.

H2: Augmenting grapheme input with POS will improve G2P accuracy, but as it is less useful for disambiguation than morphological boundaries, this improvement will be less than than G2P with gold-standard morphological labels. Combining both will lead to optimal performance.

H3: When looking at a disjoint set of root morphemes, where roots in the test data are unobserved in the training data, phoneme transcription accuracy will be lower than in randomly-split data-sets, but the impact of this reduction will be mitigated by the provision of segmentation. Again, we predict gold-standard segmentation will be most effective, with LMVR the best automatic segmentation method.

H4: Training a BiLSTM to predict morphologically-segmented phonemes from graphemes from *Unisyn* will also increase phonetic transcription accuracy compared to baseline G2P.

H5: With the approaches used in NMT as a replacement for gold-standard segmentation will lead to an improvement in accuracy over G2P, but to a lesser extent than gold-standard labels. As seeded BPE is less agnostic about the linguistic structure of the graphemes, it should lead to better performance than BPE. LMVR segmentation should outperform both seeded and standard BPE as it provides closer-to-gold segmentation.

5 Methodology

5.1 Model nomenclature

The models we have trained are named with the following pattern with keywords:
`DATASET_ADDITIONAL_INPUT2OUTPUT`
These keywords have options, listed in the table below.

Table 5: Keyword options for model names

DATA	**ADDITIONAL**	**INPUT**	**OUTPUT**
Uni (Unisyn)	d (disjoint)	G (grapheme)	P (phonemes)
CMU (CMUDict)	s (simultaneous)	Mo (gold morph boundaries)	MoP (morph+phones)
LC (LexiCorn)	r (reduced-size dataset)	BPE (on graphemes)	
		SeedBPE	
		LMVR	

5.2 BPE and LMVR formatting

To investigate the effectiveness of providing morphological information, and to test the fruitfulness of using automatic segmentation as a preprocessing stage, the BPE algorithm and LMVR algorithm are used in the following experiments. However, by default, after BPE and LMVR algorithms are applied on a corpus of plain-text data of whitespace-separated words; they produce a segmented version of the same corpus. Post-algorithm, a typical entry appeared like so:

Table 6: segmentation algorithm output, compared to *Unisyn* entries

Process	**Entry example**
Unisyn grapheme	v i s c o s i t y
Unisyn grapheme+morph	{ v i s c = = o u s } > i t y >
BPE-segmented grapheme	vis@@ co@@ s@@ ity
LMVR-segmented grapheme	visc osity

As a result, if left like this, the neural network would process each cluster of characters as a single input, making the source vocabulary far larger, the data points far sparser, and the relationship between input and output even less monotonic than it was before. To combat this, and increase similarity to the morpheme-labelled data in *Unisyn*, a script was written to replace the segmentation character in BPE/LMVR output with "|", and then whitespace-separate each character. As a result, input to the model for LMVR and BPE experiments appeared as follows:

Table 7: Inputs to the NN after transforming LMVR/BPE output

NN model input	**Entry example**
BPE	v i s | c o | s | i t y
LMVR	v i s c | o s i t y

A link to the GitHub repositories for the BPE and algorithms is included in the Appendix.

When running the BPE and LMVR algorithms to preprocess the training data, the segmentation algorithms themselves were trained on that same training set. However, when aiming to bias BPE toward morphology for English and Cornish for use in SeededBPE experiments, a corpus of common affixes needed to be created, in order to train with the BPE algorithm. The resource provided by Honig (2000) was used for training English-language seeded BPE, and the equivalent Cornish SeededBPE corpus was adapted from *Lesson 2 - Common affixes and clitics* (Kernewegva 2009, online).

5.3 Model specifications

All experiments performed in this study were performed using OpenNMT-py (2016). In order to fairly test the effect of preprocessing the input to the network, it was essential to keep the parameters of the model itself identical. In order to allow for any results from this paper to be reproduced or used as a helpful baseline for future work, the specifications of the model are as follows:

The model itself was configured as a Bi-directional LSTM recurrent neural network (BDLSTM), with a dot-product attention (Luong et al. 2015) between the final embedded encoder hidden layer and the final decoder hidden layer prior to the soft-max classification. The dropout probability was 0.3, and was used to avoid over-fitting on the training set. The encoder and decoder both contained 2 layers of LSTM units. Training occurred for 100000 training steps in 2000 epochs of 50 training steps, with models saved every 5000 steps. From that set of models, the best performing model (based on validation set loss) was kept and used for the results published in the following experiments. The average training time of a model for a full 2000 epochs was 14 hours. The 35 models trained in this paper therefore required a cumulative 490 hours of training time. The hyper-parameters were optimised by performing a parameter sweep on a smaller 'toy set' of 25% of data. Time limitations prevented a full hyperparameter sweep on a full-size data-set.

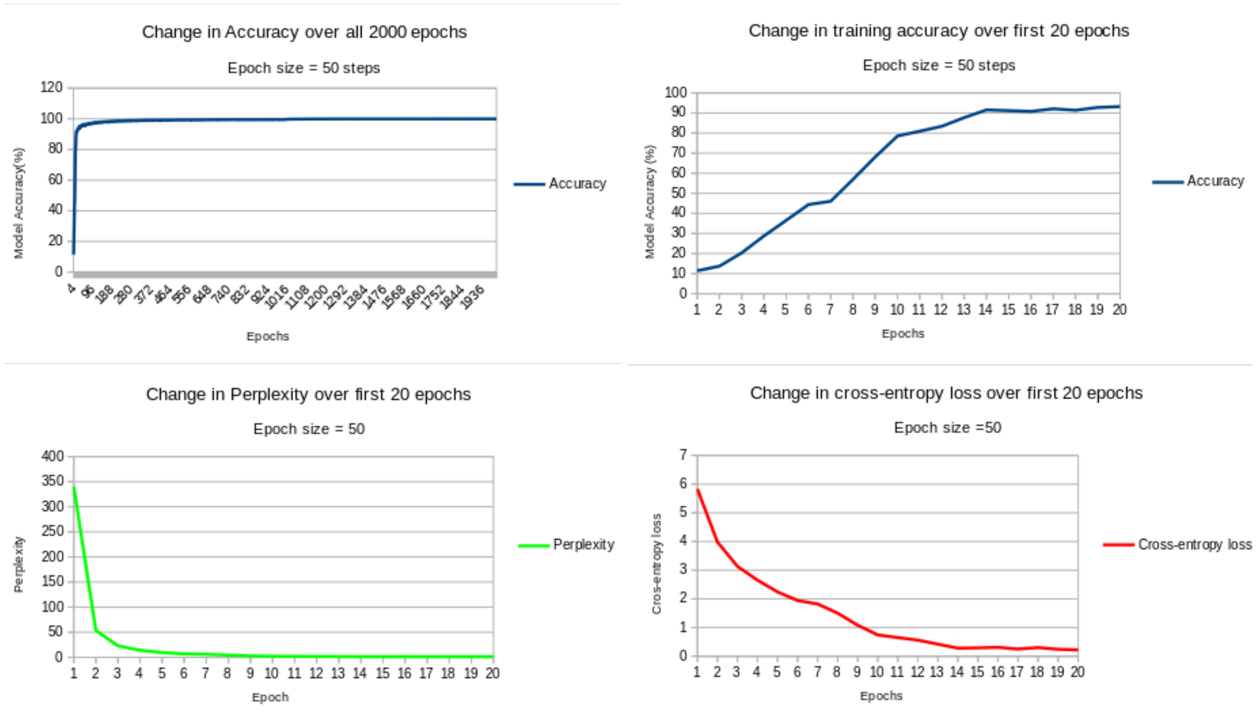

Figure 6: Graphs illustrating how training improved accuracy, cross-entropy loss and perplexity values on model Uni_G2P

Figure 6 illustrates the measures by which model training was measured. It is clear that improving the models a small amount is the most time consuming, and expensive part of the process. By 20 epochs, accuracy on training set data = 93.19%. with cross-entropy loss of 0.22 and perplexity of 1.24. The best model in this case was saved after 1900 epochs with accuracy = 99.98%, cross-entropy loss 0.01 and perplexity of 1.

5.4 Evaluation metrics

Performance is evaluated using Word Error Rate (WER), calculated by counting the number of incorrect phonetic transcriptions, and using that to calculate the percentage of incorrect transcriptions in total.

$$\text{WER} = \frac{Incorrect\ Transcriptions}{Total\ Transcriptions} \times 100$$

WER is useful to assess the efficacy of a model as it provides an accuracy score for the total number of correct transcriptions, however, if a single transcription contains more than one error, it only counts as a single failure, and additionally, there is no way to distinguish between a minor mistake in transcription and an entirely incorrect and nonsensical transcription. The table below shows two entries, each of which are considered 'equally bad' when evaluating with WER.

Table 8: Two predicted outputs, each equally wrong with WER

Class	**Entry**
Correct	f l *o w rr p -o t s
Minor error	f l *o w rr p -o d s
Major errors	t l a p rr p a p t

The single minor error here would not be problematic for a human to hear, and so is perhaps unfairly penalised by WER. Similarly the prediction with many major errors would be unintelligible if synthesised for a human to respond to, and therefore is perhaps not penalised enough by WER. A resolution to this is to evaluate with Phone Error Rate (PER).

$$\text{PER} = \frac{Insertions + Deletions + Substitutions}{All\ phones\ in\ corpus} \times 100$$

PER looks on a phone-by-phone basis, and is calculated by dividing the sum of all inserted, deleted and replaced phones in a prediction, by the total number of phones in the reference corpus. This metric resolves the issue of over-stating a minor error, but as a result, a model that gets 8/10 predictions correct, but the incorrect two **very** wrong, may have lower PER reported than a model that makes a single error on 9/10 transcriptions. By reporting both, we aim to balance the two and give a clearer overall picture of the efficacy of each pre-processing stage. In addition to this quantitative analysis, we found it useful to look at the type of errors that models were making, allowing us to see *how* these models are changing with a given pre-processing stage. A script was written to consider transcriptions with errors, which were categorised into 4 groups: vowel errors, consonant errors, ordering errors, and gross errors. Examples of each are shown in Table 9 below.

Table 9: Examples of error categories for qualitative analysis

Class of Error	**Example**	**Reference**
Vowel	f l I w rr p -o t s	f l *o w rr p -o t s
Consonant	f l *o w rr p -o g s	f l *o w rr p -o t s
Ordering	f l *o w rr t -o p s	f l *o w rr p -o t s
Gross	f l *o w *o w *o w *o w	f l *o w rr p -o t s

Vowel and consonant errors are defined as any situation wherein a vowel or consonant is incorrectly predicted, respectively. Ordering errors occur when the correct phones are predicted but in the wrong order. Gross errors are defined as when more than 30% of the phones predicted for a given transcription are incorrect. If an error is marked as gross, it is not counted as another class of error. We highlight these as particularly costly mistakes, where if synthesised, the result would be unintelligible and unrecognisable from the source.

6 Results

6.1 *Unisyn*

6.1.1 *Unisyn* baseline

Table 10: Results of modifying input for phoneme prediction for *Unisyn* with random data-sets

Data=Unisyn	**WER (%)**	**PER (%)**
G2P	14.94	3.08
Mo2P	7.76	2.20
BPE2P	14.53	3.50
SeedBPE2P	13.80	3.27
LMVR2P	12.43	3.01

Immediately, the results show that, as predicted, providing morphological information to an otherwise identical model improves its accuracy, with WER decreasing to 7.76% (Δ7.18%). and PER decreasing to 2.20% (Δ0.88%). Automatic segmentation methods also improve the accuracy of the model over baseline G2P, with seeded BPE improving over standard BPE, again as predicted for WER, however PER increases for BPE based models. LMVR proved to be the best preprocessing stage, with a reduction in both WER and PER over the baseline G2P model. However, auto-generated segmentation still did not improve over the gold-standard labelling. Table 11 shows how the distribution of errors vary with models.

Table 11: Error distribution across models trained on *Unisyn* with random data-sets

data= Uni	**Vowel (%)**	**Consonant (%)**	**Ordering (%)**	**Gross (%)**
_G2P	51.02	31.31	16.47	1.20
_Mo2P	54.11	30.07	15.12	0.69
_BPE2P	50.77	31.86	15.16	2.21
_SeedBPE2P	52.07	31.52	14.45	1.96
_LMVR2P	51.36	31.14	16.20	1.30

Here, the effect of using BPE and seeded BPE for segmentation on gross errors becomes clear. The increase makes a strong case for **not** using BPE or seeded BPE in a real-use scenario. However, the proportion of errors for LMVR mirrors baseline G2P, yet reduces the total WER and PER over baseline graphemes (as well as over either BPE variant). Therefore, LMVR is shown to be the most effective pre-processing stage for this data-set.

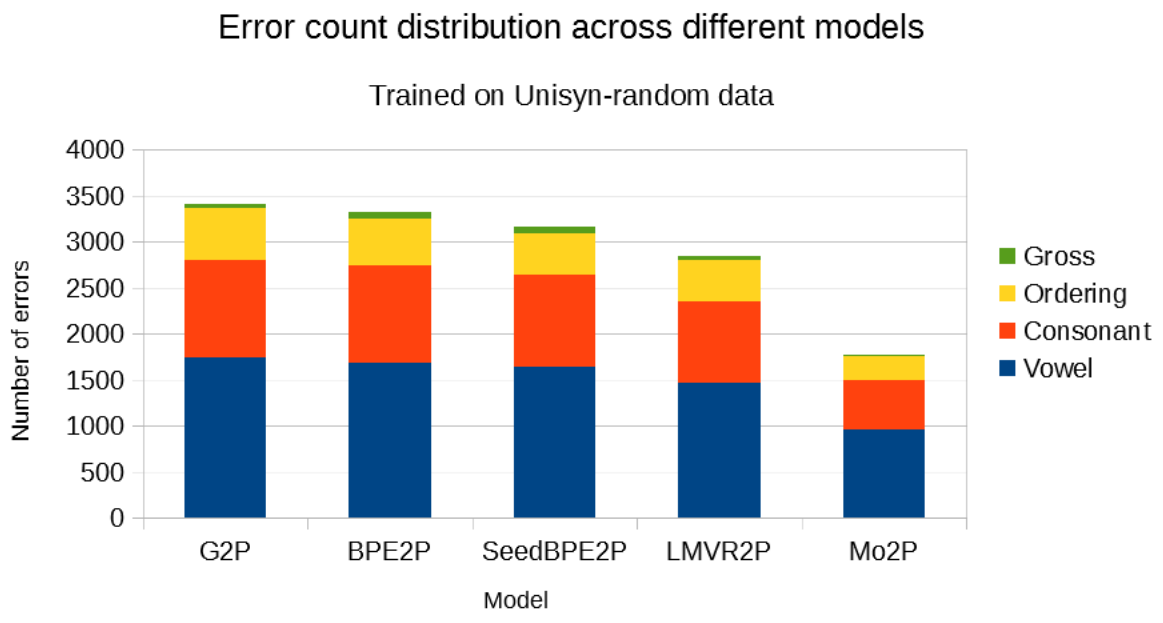

Figure 7: Number and distribution of errors for *Unisyn*-random trained models

We propose that some gross errors may be due to the recurrent nature of the model. In the Uni_G2P model, the grapheme `v i s c o s i t y` is incorrectly predicted as `v i s k o s o s o s`, rather than the reference: `v i s k o s @ t iy`. This suggests that the model may have over-estimated the importance of the input sequence `o s` and aligned it to all remaining decoder embeddings. These gross errors are common to all models (as it is caused by the model architecture, not the preprocessing stage). However additional gross errors which are not obviously caused by the recurrent nature of the model are more common to seeded/standard BPE preprocessing.

6.1.2 *Unisyn*-POS

To encode POS information, we create an additional feature vector which is appended to the grapheme as input to the encoder.

Table 12: Results of modifying input for phoneme prediction for *Unisyn*+POS

data=Unisyn	**WER (%)**	**PER (%)**
G2P	14.94	3.08
Mo2P	7.76	2.20
POS	11.42	2.91
Mo+POS2P	6.98	2.12

Including POS with the grapheme improves WER and PER over base G2P, but not to the same extent as gold-standard morphological labels. Time constraints limited us to only running a preliminary experiment here. However, it serves as another example of linguistic information improving model accuracy. Using an automatic POS tagger such as that included in NLTK (Bird, 2009) may be a further avenue to automatically collecting information to improve G2P. The model which uses gold-standard morphology and POS outperforms both models augmented separately, and with word accuracy of 93.02% and phone accuracy of 97.88% is the best performing model trained on *Unisyn* data. This shows that the improvements made by POS and morphology over G2P alone are different error subsets, and therefore this merits using both to improve model performance.

6.1.3 *Unisyn* - disjoint

It is clear that when training and testing on a disjoint set of roots that the base G2P model gets significantly worse for WER (Δ+23.48%) and PER (+3.97%) compared to training on randomly-split data. Similar Bikmetova's findings, adding morphological information led to Δ-24.13% WER and Δ-3.36% PER over the base disjoint G2P, and therefore we can confirm that adding morphological information is a suitable avenue for resolving the issue of creating phonetic transcriptions for unknown words.

However, a surprising result from this experiment is that using all automatic segmentation

Table 13: Results of modifying input for phoneme prediction for *Uni* with disjoint data-sets

data=Unisyn	**WER (%)**	**PER (%)**
_d_G2P	38.42	7.05
_d_Mo2P	14.29	3.39
_d_BPE2P	40.06	7.35
_d_SeedBPE2P	39.87	7.30
_d_LMVR2P	45.91	8.15

methods, lead to increased WER over the disjoint G2P model (Δ+1.64% , Δ+1.45% and Δ+7.49%respectively), as well as increased PER (Δ+0.30%, Δ+0.25% and Δ+1.10%respectively). When there is a disjoint between training and test data, only hand-labelled morphological boundaries make a notable improvement over standard G2P, and automatic segmentation trained on the training data causes an overall degradation in transcription accuracy. We propose that because the segmentation methods can only learn from the training data in terms of predicting morphological boundaries, they are unable to adapt on test data. There is an explanation for higher error when using LMVR. The algorithm may be over-fitting to the training set segmentations, and is therefore least generalisable to the test set. One solution to this would be to retrain BPE/LMVR on the test data at test time, and then re-segment the training data and retrain. However, this is expensive and time-consuming; and therefore not practical for reasonable use.

6.1.4 *Unisyn* - simultaneous prediction

Table 14: Results of modifying input for simultaneous morpheme and phoneme prediction for *Unisyn* with random data-sets

data = Unisyn	**WER(%)**	**PER(%)**	**Δvs Uni_*x* WER(%)**	**Δvs Uni_*x* PER(%)**
_s_G2MoP	12.27	3.18	-2.27	+0.10
_s_Mo2MoP	9.49	2.53	+1.73	+0.33
_s_BPE2MoP	25.90	5.35	+11.37	+1.85
_s_SeedBPE2MoP	44.16	8.73	+30.36	+5.46
_s_LMVR2MoP	19.49	4.01	+11.73	+1.81

Model performance deteriorates when models are trained to predict both morphological and phonetic output. The only metric by which any improvement is found is for WER on a G2MP model. We posit that the substantial increase in WER for models which take in pre-processed input is due to the mismatch between the quasi-morphological segmentations that result from applying BPE or LMVR and the gold-standard boundaries that the model is required to learn. Forcing simultaneous learning of morphology may separately be an alternative route for improving G2P, but when combined with our preprocessing stages the net result is negative.

6.2 CMUDict

Table 15: Results of modifying input for phoneme prediction for *CMUDict*

data = CMU	**WER(%)**	**PER(%)**	**Δvs *Uni_x* WER(%)**	**Δ vs *Uni_x* PER(%)**
_G2P	29.57	6.03	+14.63	+2.95
_BPE2P	30.49	6.22	+15.96	+2.72
_SeedBPE2P	30.68	6.21	+16.88	+2.94
_LMVR2P	29.06	6.08	+16.63	+3.07
Joint-sequence [1]	24.53	5.88	-	-
BDLSTM-CTC [2]	25.80	-	-	-
CNN-BDLSTM [3]	25.13	-	-	-

Examining the results of the *CMUDict* experiments, we can conclude the following: All models' performance on *CMUDict* is substantially worse than on *Unisyn*, as WER and PER both increase (these models are 2x as likely to make an error than models trained and tested on Unisyn data.)

This was predicted, due to the fact that *CMUDict* is of lower quality than *Unisyn*, with more rare words, non-English words and less consistent transcription. More surprising is that the automatic segmentation preprocessing stages which led to improvements on *Unisyn* data, degrade the quality of the model when training on *CMUDict*. We propose that because of the lower-quality data in *CMUDict*, the system is noisier - additional segmentation errors introduced by the automatic methods increase this noise even further and accordingly, phone and word error rates increase. LMVR still proves to be a useful pre-processing stage, reducing WER by 0.51%, however, with an increase in PER by 0.05% when compared to the base G2P model. It is worth noting that with a test set word size of 26756, the 0.51% increase is equivalent to an improvement on 136 words, whereas the 0.05% reduction from a test set phone count of 170322 is equivalent to 85 phones. This again suggests that LMVR is a suitable pre-processing stage which can consistently improve G2P accuracy. We also see that by comparing our WER to Bisani and Ney [1] (2008), Rao et al [2] (2015) and Yolchuyeva et al. [3] (2019) that more complex models can achieve better results on this dataset, which then raises the question: How would our preprocessing stages affect these results?

6.3 LexiCorn

Table 16: Results of modifying input for phoneme prediction for *LexiCorn*

data = LC	**WER (%)**	**PER (%)**
_G2P	11.74	2.08
_BPE2P	12.55	2.55
_SeedBPE2P	11.63	1.88
_LMVR2P	10.40	1.73

Experimental results show that automatic segmentation methods improve over baseline G2P for Cornish language data, and that error rates for Cornish in general are fairly low, especially given the small quantity of training data. One reason for this is that the relationship between letter and sound is more monotonic and consistent in Cornish than English, due to a written standard having been determined relatively recently and low usage rates. Therefore there are fewer opportunities for novel innovations or language change. Because of this more monotonic relationship, G2P prediction is easier. Additionally, the surface form of an affix is generally consistent, and therefore statistical count-based methods may perform better in correctly identifying boundaries than on equivalent English data. However, BPE alone does not improve G2P, and so a small amount of linguistic knowledge is still required. Once again, performing LMVR as a pre-processing stage improves WER and PER the most over the baseline G2P model.

6.4 Low-resource simulation with *Unisyn* data

The good performance of preprocessing stages on *LexiCorn* (with the smallest quantity of data), combined with the poor performance on *CMUDict* (with the largest quantity of data) raised an additional question: Are these preprocessing stages (and more generally, adding quasi-morphological segmentation) for G2P prediction more effective when data is scarcer?

Table 17: Results of modifying input for phoneme prediction for low-resource *Unisyn*

Model	**WER (%)**	**PER (%)**
Uni_0.1_G2P	53.36	10.76
Uni_0.1_M2P	37.91	7.37
Uni_0.1_BPE2P	58.50	13.31
Uni_0.1_SeedBPE2P	57.49	13.12
Uni_0.1_LMVR2P	53.25	10.14

Here we see a major degradation in both WER and PER, compared to the models trained on the full *Unisyn* corpus. Both Seeded BPE and LMVR were suitable preprocessing steps on *LexiCorn* However, seeded BPE is not supported as a solution for low-quantity English data. The general discrepancy in error rates between these experiments and the *LexiCorn* ones suggest that Cornish is far more mappable. This is expected, as the written standard was devised recently, and designed to be consistent. The Uni_0.1 experiments show that only LMVR segmentation improves over baseline G2P, and the improvement is small (an improvement of 0.11% WER is equivalent to a single entry). However, LMVR segmentation does improve PER to greater degree (0.62% PER). Gold standard morphological boundaries far out-perform any preprocessing stage in reducing both phone and word error rate.

7 Analysis

The combined results of these experiments show that neural G2P transcription accuracy is highly sensitive to the quantity and quality of training data, but that when available, high-quality morphological segmentations improve G2P accuracy. Gold-standard labels on *Unisyn*-trained models universally outperform G2P, or any automatic segmentation method. Therefore, Hypothesis 1 is supported. Including gold-standard POS also improves G2P, but to a lesser extent than gold standard morphological boundaries. Hypothesis 2 is also supported. Creating a disjoint between training and test sets make the preprocessing stages redundant, as they reduce performance. This is problematic, as words that are required to be predicted by a G2P engine, rather than lookup are likely to have entirely new roots. One solution would be to manually update a lexicon with root morphemes, but leave derivation prediction to the G2P model.

Hypothesis 3 is partially supported, as augmentation with gold-standard morphology improves WER and PER over G2P alone; however, automatic segmentation causes model performance to deteriorate.

Hypothesis 4 is not supported, as forcing models to simultaneously predict morphology and phonetic transcription leads to lower word and phone accuracy over equivalent models predicting only phonetic transcription.

In terms of using automatic segmentation methods to augment G2P, only LMVR provides consistent improvement over G2P on *Unisyn*-trained models. Therefore Hypothesis 5 is partially supported.

Results when training with CMUDict show that the quality of data in general has a large effect on G2P accuracy that is not able to be mitigated by a preprocessing stage; however LMVR proved to again led to a systematic improvement over baseline G2P - mirroring the improvement found when used on Unisyn. However, other work on CMUDict shows that the choice of model used for predictions is a large factor on overall accuracy. Higher quality lexica such as *Combilex* (Fitt and Richmond 2006) could also be used. Further research could involve employing our preprocessing stages to input for other models with different architectures.

LMVR segmentation again led to improvements in word and phone transcription accuracy when tested on Cornish data, showing that it is robust to language variation and quantity and quality of training data. BPE was also effective in this case, and seeding the algorithm with a small selection of affixes from the target language improves its performance over basic BPE, and can lead to an overall reduction in WER when compared to the baseline G2P models. Seeded BPE may be an alternative preprocessing stage when computational resources are at an absolute premium. However, improvement is inconsistent, and our experiments show it to be inferior to LMVR.

8 Conclusions

The findings of this paper open many avenues for further study. It has been shown that including morphological information when performing G2P conversion has a systematic improvement when that data is of a gold standard. LMVR is shown to provide a robust and systematic, (albeit smaller) improvement over G2P. However, this has only been tested when using a BiLSTM for G2P conversion. As discussed above, NNs are particularly effective at predicting abstract and non-linear relationships between input and output; but at significant computational cost. Because the motivation behind preprocessing the grapheme data is to reduce the amount of training data required; the preprocessing stages presented here could be combined with non-neural G2P alternatives which are better at dealing with yet smaller quantities of data, or alternatively, with architectures that lead to an improved performance over a BiLSTM.

Additionally, the experiments performed in this paper should be performed on a greater selection of corpora to evaluate LMVR's impact on G2P more thoroughly. The clear improvement on phonetic prediction accuracy for *LexiCorn* suggests that it may be more beneficial to employ these segmentation algorithms when presented with data scarcity, but that as the quantity of training data increases, these benefits are lost, particularly if using BPE or Seeded BPE. LMVR also led to improvement with our low-resource simulation on English, a less mappable language than Cornish.

The creation of *LexiCorn* allows Cornish-language speech technology development, and provides an option for low-resource language processing. When released, *LexiCorn 1.0* will be bilingual, allowing for other tasks including MT to be performed.

Finally, our experiment with including POS for phonetic prediction showed that gold-standard labels were beneficial. Automatic collection of POS for G2P augmentation on *CMUDict*, and more challengingly, *LexiCorn* (as there is no toolkit for parsing and tagging Cornish) could lead to a set of experiments mirroring those performed in this paper on morphology.

9 References

Antworth, E. L. 1991. *PC-KIMMO: a two-level processor for morphological analysis.*

Ataman, D., M. Federico, M. Negri and M. Turchi. 2017. 'Linguistically Motivated Vocabulary Reduction for Neural Machine Translation from Turkish to English' in *The Prague Bulletin of Mathematical Linguistics. No. 108.* 331-342

Bahdanau, D., K.H. Cho and Y. Bengio. 2015. 'Neural Machine Translation by jointly learning to align and translate'. in *The proceedings of ICLR 2015.*

Bikmetova, M. 2018. *Grapheme-to-Metaphoneme Conversion for Unisyn and Combilex Baseform Transcriptions.* Unpublished Manuscript. University of Edinburgh, Edinburgh, UK.

Bird, S., E. Loper and E. Klein. 2009. *Natural Language Processing with Python.* O'Reilly Media Inc.

Bisani, M. and H. Ney. 2008. 'Joint-sequence models for grapheme-to-phoneme conversion.' In: Speech communication. 50.5: 434–451.

CMUDict 0.7b. 2014. *The CMU Pronouncing Dictionary.* Retrieved from: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Demberg, V., G. Moehler and H. Schmid. 2007. 'Phonological Constraints and Morphological Preprocessing for Grapheme-to-Phoneme Conversion.' in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.* 96–103.

Fitt, S. 2000. *Documentation and user guide to UNISYN lexicon and post-lexical rules.* Center for Speech Technology Research, University of Edinburgh, Tech. Rep.

Fitt, S. and K. Richmond. 2006. 'Redundancy and productivity in the speech technology lexicon-can we do better?' in *Ninth International Conference on Spoken Language Processing.*

Gage, P. 1994. 'A New Algorithm for Data Compression'. C Users J., 12(2):23–38.

Gerlyver Kernewek. 2018. Truro, UK. Akademi Kernewek.

Grönroos, S.A., M. Kurimo, P. Smit and S. Virpioja. 2014. ‘Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology’ in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 1177-1185.

Hochreiter, S and J. Schmidhuber. 1997. ‘Long short term memory’. in *Neural Computation* 9(8):1735.

Honig, B., L. Diamond and L. Gutlohn. 2000. *Teaching Reading Sourcebook: For Kindergarten Through Eighth Grade.*

Kaplan, R. M. and M. Kay, 1994. ’Regular models of phonological rule systems.’ in *Computational linguistics* 20.3: 331–378.

Kernewegva. 2009. *Lesson 2 - Common affixes and clitics*. Retreived from: https://www.kernewegva.com/PDFs/NebGer/NebGer_02_affixes_clitics.pdf.

Kominek, J. 2009. *TTS from Zero: Building synthetic voices for new languages.* Unpublished doctoral thesis. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Luong, M.T., C. D. Manning and H. Pham. 2015. ‘Effective approaches to Attention-based Neural Machine Translation.’ in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

Novak, J.R., K. Hirose and N. Minematsu. 2012. ‘WFST-based Grapheme-to-Phoneme Conversion: Open Source Tools for Alignment, Model-Building and Decoding’ in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing* 45–49.

Olah, C. 2015. *Understanding LSTM Networks.* Retrieved from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

OpenNMT. 2016. *Open-source neural machine translation.* Retrieved from: http://opennmt.net/

Richmond, K., R. Clark and S. Fitt. 2010. ‘On generating combilex pronunciations via morphological analysis.' in *INTERSPEECH 2010*. 1974-1977.

Rao. K., F. Peng, H. Sak and F. Beaufays. 'Grapheme to phoneme conversion using long short term memory recurrent neural networks' in Proceeding of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

Rosenblatt, F. 1958. 'The perceptron: A probabilistic model for information storage and organization in the brain'. in *Psychological Review* 65.6: 386-408.

Sennrich, R., B. Haddow and A. Birch. 2015. 'Neural Machine Translation of Rare Words with Subword Units' in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 1715–1725.

Sutskever, I., O. Vinyals and Q. V. Le. 2014. ‘Sequence to sequence learning with neural networks’ in *NIPS’14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2.* 3104-3112.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin. 2017. ‘Attention is all you need’. in *Advances in Neural Information Processing Systems.* 5998-6008.

Wells, J.C. 1982. *The Accents of English.* Cambridge, UK.

Wells, J. C. 1995. ‘Computer-coding the IPA: a proposed extension of SAMPA’ Retrieved 07/08/2019 from: https://www.phon.ucl.ac.uk/home/sampa/ipasam-x.pdf.

Yolchuyeva, S., G. Nemeth and B. Gyires-Toth. 2019. ‘Grapheme-to-Phoneme Conversion with Convolutional Neural Networks’ in *Appl. Sci* 9.6: 1143-1160.

Appendix

LexiCorn is publicly accessible from:
`https://www.github.com/MadDanWithABox/lexicorn`

OpenNMT-py is available from: `https://github.com/OpenNMT/OpenNMT-py`

Corpus processing scripts are available at
`https://www.github.com/MadDanWithABox/msc`

Sennrich's implementation of BPE can be found at:
`https://www.github.com/rsennrich/subword-nmt`

Ataman's implementation of LMVR can be found at:
`https://www.github.com/d-ataman/lmvr`