Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Asking for Help: Grapheme-to-Phoneme Conversion using Active Learning at Application Time

vorgelegt von

Mario Mohr

MIN-Fakultät

Fachbereich Informatik

Language Technology Group

Studiengang: Informatik

Matrikelnummer: 6246684

Erstgutachter: Prof. Dr. Chris Biemann

Zweitgutachter: Benjamin Milde

Abstract

In natural language processing (NLP), situations are common in which a machine learning system could theoretically request help from a human oracle at application time. Due to constraints like the required domain-specific proficiency of the oracle, however, such manual labeling is often costly.

In this thesis, it will be investigated whether it is possible for such a system to identify problematic inputs at application time, and which approaches are best suited to achieve this. To this end, some approaches from the field of active learning are introduced.

Active learning is a comparatively recent and currently sparsely researched field of machine learning that aims to minimize data labeling costs by allowing a partially trained model to select, from a larger set of unlabeled data, those samples it deems the most beneficial for its ongoing training, and send these to an oracle for manual labeling. The problem of identifying the most interesting samples at training time is related to that of identifying the most problematic ones at application time, and the transferability of some active learning techniques into application-time usage is discussed.

The NLP domain chosen for this thesis is grapheme-to-phoneme conversion (G2P). G2P plays an important role in several fields of NLP applications and is currently being researched intensely. Two state-of-the-art G2P approaches are implemented as a baseline to test against: joint sequence model based G2P in the Sequitur implementation, and Long Short-Term Memory based G2P, a comparatively recent, but somewhat successful approach in sequence-to-sequence learning. While some of the request decision approaches discussed are G2P-specific, others are of a more generalistic kind, rendering the results of this thesis just as relevant to other machine learning fields.

The results show that while several active learning based approaches to request decision fail, both simply due to bad performance as well as due to incompatibility with the G2P domain, others are somewhat promising. The generalistic approach of interpreting a system's output in terms of confidence towards an input sample is shown to be a robust and efficient metric to identify problematic samples at application time. Additionally, the domain-specific approach of grading test samples by the system's ignorance towards the input, i.e. the difference of the input to the bulk of the training data in terms of semantic distance, morphologic distance or similar, is shown to be a sub-optimal, but promising approach.

Acknowledgments

The author wishes to express their sincere thanks to Professor Chris Biemann and Maximilian Strake for their valuable professional advice and ongoing encouragement.

Special thanks go to Benjamin Milde for the continuously productive exchange of ideas, for late-night proofreading and valuable criticism.

Contents

1 Introduction

Grapheme-to-Phoneme Conversion (G2P) is the task of converting the ortographic representation of a word, consisting of *graphemes* such as letters, into its *phonemic* representation, i.e. a symbolic representation of its pronunciation.

The phonemic representation of a language's words plays an important role in a number of Natural Language Processing (NLP) applications. The most obvious of these is *Speech Synthesis*, where the knowledge of the pronunciation of a word - usually enriched with information about emphasis, sentence structure and similar aspects - allows an artificial agent to emulate natural speech. Another such application is its use in *Automatic Speech Recognition* (ASR), where the knowledge of a language's words typical pronunciations allows a recognition component to distinguish and recognize individual words in continuous audio signals.

Systems requiring phonemic knowledge often employ pre-built, (semi-)manually created datasets such as pronunciation dictionaries to provide relatively accurate knowledge on the pronunciation of words. However, due to the high cost of manual manual G2P conversion by experts and the constantly-evolving nature of natural languages, it is inherently impossible to generate final, future-proof complete datasets. Additionally, domain-specific words (such as medical terms) are often exceedingly difficult to completely and accurately cover.

In these cases, NLP systems often make use of G2P components. These are employed whenever a word of interest is not included in the pronunciation dataset(s) used by the system. Due to the advent of Natural Language Processing in the last decades and the rising extension of NLP to highly non-formalized natural language (such as social media), research in G2P approaches has experienced considerable interest in recent years. In addition to this, some overlap with other fields of interest (such as generalistic sequence-to-sequence mapping) has produced a number of exciting research opportunities for G2P conversion. Two of these approaches, *Joint Sequence Models* and *Long Short-Term Memory networks*, will be introduced in a later section of this work.

In this thesis, the problem of G2P in a specific environment will be explored: that of a *semi-supervised* G2P component, for which incompletely labeled pronunciation data is available and further manual labeling by experts is available, but expensive. Thus, we will explore approaches for a G2P component to efficiently request labeling only for the most critical of word pronunciation, while simultaneously minimizing the overall pronunciation error as much as possible. To this end, several approaches based on *Active Learning* algorithms will be introduced and their performance evaluated.

The language of interest for this paper will be English; however, efforts will be made to keep all approaches generalistic enough to make them adaptable with reasonable amount to other languages.

1.1 Related Work

As far as the author is aware, no attempts to utilize active learning techniques for the application-time requesting of manual labeling have been made so far.

Active learning at training time in the context of G2P systems appears to be a moderately researched field. Kominek and Black (2006) propose a simple, perplexity-based active learning approach to the training of a G2P, with results slightly better than that of a randomly trained model. Kominek (2009) follows a similar approach, producing results that are only marginally better than that of a randomly trained model and, additionally, utilizing metrics that are unfit for application-time requesting of oracle help.

Schlippe et al. (2012) discuss a problem similar to the one discussed in this thesis: identifying problematic samples from a larger set of phonemizations. However, in this case, a G2P component is in fact used to check the phonemizations already present in a pronunciation dictionary, making this approach unfit for this thesis' application.

Kim and Snyder (2013) apply a technique similar to active learning to reduce the labeling effort over unlabeled data for a G2P system; however, again, their approach is only applicable during training time and offers no benefit for the identification of problematic phonemizations at application-time.

In (Shen et al., 2011), active learning is used to reduce manual labeling for a homograph disambiguation task on mandarin natural language. While not directly related to G2P, they use metrics similar to the *confidence* and *confidence entropy* proposed in Chapter 3.2.

Active learning in general has successfully been applied to several fields, such as:

- *computer vision*, e.g. object recognition (Kapoor et al., 2007) and video tagging (Yang, 2003)
- *robotics*, e.g. inverse models for high-dimensional spaces, such as inverse kinematics (Baranes and Oudeyer, 2013), and
- *signal processing*, e.g. music retrieval (Mandel, Poliner, and Ellis, 2006).

2 Theoretical Background

2.1 Joint Sequence Models

A **Joint sequence model** is a probabilistic approach to describe the relation between the elements of two sets M_1, M_2 for which pairs of sequences $(s_1, s_2) \in {M_1}^* \times {M_2}^*$ stand in a predefined, meaningful relation R to each other. Generally, a joint sequence model may be found for any two sets M_1, M_2, as long as there is a set of sequence tuples that relates the two sets to each other; an example can be found in the field of machine translation of natural language (Durrani, Schmid, and Fraser, 2011), for which the relation R_{trans} might contain known valid translations between two word sets of different languages:

$$\begin{gathered}(\text{"}The\ soccer\ ball\ is\ red\text{"}, \text{"}Der\ Fussball\ ist\ rot\text{"})\\ \in R_{trans} \subset {M_{English}}^* \times {M_{German}}^*\end{gathered}$$

In a joint sequence model, these tuples of joint sequences are subdivided into *joint units* that pair shorter sequences from the two sets in a more fine-granular set of tuples. A possible subdivision of the English-German sentence translation above might, for example, be:

$$\{(\text{"}The\text{"}, \text{"}Der\text{"}), (\text{"}soccer\ ball\text{"}, \text{"}Fussball\text{"}), (\text{"}is\text{"}, \text{"}ist\text{"}), (\text{"}red\text{"}, \text{"}rot\text{"})\}$$

It is important to note that this set of joint units generated from the sentence above is *a*, but not *the only* possible such subdivision. An equally valid set of joint units would be:

$$\{(\text{"}The\ soccer\text{"}, \text{"}Der\text{"}), (\text{"}ball\text{"}, \text{"}Fussball\text{"}), (\text{"}is\ red\text{"}, \text{"}ist\text{"}), (\text{""}, \text{"}rot\text{"})\},$$

as there is no information inherent to the translated sentence that suggests the first of the two subdivisions to be preferable over the second. In fact, for the generation of a joint sequence model for English-German translations, *every possible subdivision* (or *co-segmentation*) of a given sentence is considered.

If applied to a sufficiently large set of English-German sentence translations, however, one of the crucial points of a joint sequence model will become apparent: while in the context of a single sentence, $(\text{"}The\ soccer\text{"}, \text{"}Der\text{"})$ and $(\text{"}The\text{"}, \text{"}Der\text{"})$ may appear to be equally valid co-segmentations, in the context of a larger number of sentence translations, the joint unit $(\text{"}The\text{"}, \text{"}Der\text{"})$ will appear significantly more often than $(\text{"}The\ soccer\text{"}, \text{"}Der\text{"})$. In other words, $(\text{"}The\text{"}, \text{"}Der\text{"})$ will have a much higher *likelihood* p than $(\text{"}The\ soccer\text{"}, \text{"}Der\text{"})$:

$$p((\text{"}The\text{"}, \text{"}Der\text{"})) \quad > \quad p((\text{"}The\ soccer\text{"}, \text{"}Der\text{"})),$$

where the likelihood p of a joint unit m is equal to its relative occurrence in the set of all possible joint units generated from all possible co-segmentation of all sentence pairs in R_{trans}:

$$p(m) = \frac{\sum\limits_{s \in S(R_{trans})} [s = m]}{|S(R_{trans})|} \tag{2.1}$$

In this way, joint sequence models capture the structural connection between short sequences of the two sets in a purely *data-driven* manner.

After some of the basic concepts of joint sequence models have been introduced, in the next subchapter, the transition to the G2P domain will be made and a more in-depth overview over the key parts of a joint sequence based G2P system will be given.

2.1.1 Joint Sequence G2P

Joint sequence models as an approach for various topics related to natural language processing have been in use since at least the 1990s (Deligne and Bimbot, 1995). One of the more recent implementations - and one that is still used as one of the state-of-the-art approaches to G2P conversion in current research (Toshniwal and Livescu, 2016; Novak et al., 2012; Milde, Schmidt, and Köhler, 2017) - is the *Sequitur* system introduced by Bisani and Ney in 2008 (Bisani and Ney, 2008).

From a joint sequence based perspective, the initial problem a G2P system attempts to solve is the following: Given are a grapheme set G, a phoneme set Φ and a relation $R_{pron} \subset G^* \times \Phi^*$ of known valid pronunciations. Desired is a function $\phi : G^* \rightarrow \Phi^*$ that maps any given word in phoneme form to its correct counterpart in phoneme form, i.e. its correct pronunciation. Bisani and Ney formalize this as follows: (Bisani and Ney, 2008)

$$\phi(g) \;=\; \underset{\phi' \in \Phi^*}{\operatorname{argmax}}\; p(g, \phi') \tag{2.2}$$

They define the likelihood $p(g, \phi')$ of a phonemization ϕ being correct for the input word g as:

$$p(g, \phi') \;=\; \sum_{q \in S(g,\phi)} p(q), \tag{2.3}$$

where $S(g, \phi)$ denotes the set of all co-segmentations of g and ϕ. The set of all co-segmentations for the word *"if"* and the (correct) phonemization "ɪf", for example, would contain the following sets of joint units (in the G2P context also referred to as *graphones*):

$$S(\text{"if", "ɪf"}) = \left\{ \begin{array}{l} \{\ (\text{"i", "ɪ"}),\ (\text{"f", "f"})\ \} \\ \{\ (\text{"i", "ɪf"}),\ (\text{"f", ""})\ \} \\ \{\ (\text{"i", ""}),\ (\text{"f", "ɪf"})\ \} \\ \{\ (\text{"if", ""}),\ (\text{"", "ɪf"})\ \} \\ \{\ (\text{"if", "ɪf"})\ \} \end{array} \right\}$$

The individual probability of a graphone sequence $p(q)$ (i.e. a co-segmentation of an input and a possible phonemization, e.g. { ("i", "ɪ"), ("f", "f") }) is approximated by the product of the individual graphones' posterior probabilities given the M previous graphones: (Bisani and Ney, 2008)

$$p(q) \cong \prod_{j=1}^{|q|+1} p(q_j | q_{j-1}, ..., q_{j-M+1}) \tag{2.4}$$

In other words, for each graphone, the probability that *this exact* graphone follows the *exact sequence* of graphones leading up to its position in the word is calculated, while limiting the window to the last M graphones. In this way, not only the statistical evidence in the training data for each individual graphone, but also the evidence for the sequence order of graphones is considered. This allows the joint sequence model to account for some of the structural properties of both the individual sets (e.g. *German* and *English*) as well as their joint properties (such as certain groups of English words translating to a single German compound word, or even grammatical properties of the languages involved).

The training of a joint sequence model occurs by maximizing the *log-likelihood* of the training set for the model. The log-likelihood of a data set N for a given model is defined as:

$$\log \mathcal{L}(N) = \sum_{i=1}^{|N|} \log \mathcal{L}(N_i) = \sum_{i=1}^{|N|} \log \left(\sum_{\mathcal{S} \in S(N_i)} p(N_i, \mathcal{S}) \right), \tag{2.5}$$

i.e. the sum of the log-likelihoods of all co-segmentations of all training input words and their correct phonemizations. To achieve this, the *expectation maximization* algorithm is employed, which will not be discussed in this thesis. For more details, Bisani and Ney (2008) give an excellent introduction into the topic.

One last noteworthy aspect of the training process is the application of two additional concepts to reduce over-fitting on the training data:

- *evidence trimming*, which only considers a graphone q_i to continue a graphone sequence if its training evidence of following the previous graphones $e(q_i, (q_{i-M+1}, ..., q_{i-1}))$ is higher than a threshold value d_M (and thus, effectively discounts extreme edge-cases with very little support in the training data), and
- *model interpolation*, which, when computing the probability of a graphone q_i following a graphone sequence $(q_{i-M+1}, ..., q_{i-1})$, not only considers the probability given by the current M-gram model, but also that of the lower-gram models $(M-1, ..., M-M+1)$.

Again, for more details, Bisani and Ney (2008) are highly suggested as a more in-depth explanation of the calculations employed.

2.2 LSTM G2P

In this subchapter, the theoretical background of the Long-Short-Term-Memory-based G2P system utilized in later chapters is introduced. Starting with the foundational basics of artificial neural networks, more complex concepts will be introduced until a sufficient overview of the most relevant components of the LSTM G2P system is achieved.

2.2.1 Feed-Forward Neural Networks

Artificial Neural Networks are an approach to computational learning that is roughly inspired the information processing capabilities of the (human) brain. The first paper relevant to the topic (although at this point, the idea of a digital *artificial* neural network was not mentioned yet) was published in 1943 (McCulloch and Pitts, 1943). In it, a first attempt at a mathematic-algorithmic description of the signal processing behaviour of neurons in the human brain was made. In the late 1950s, research into the replication of these neural approaches to machine learning began in earnest (Rosenblatt, 1958), culminating in "the first golden age of neural networks" (Yadav, Yadav, and Kumar, 2015).

While initial results of these first artificial neural networks were promising, toward the end of the 1960s, artificial neural networks had failed to live up to the high expectations set into them. Even some comparatively simple problems, such as the computation of the logical XOR operator, could not be solved (Minsky and Papert, 1969). Consequently, research interest into this field receded drastically. The reasons for the perceived shortcomings of early neural network systems are varied and will not be discussed in this thesis; however, among the more obvious ones are challenges regarding the hardware, availability of training data, and the fact that many of the approaches taken were still rather in their infancy.

In the 1980s, interest in neural networks increased again; for similar reasons as the first, and due to the advent of alternative approaches such as Support Vector Machines (Cortes and Vapnik, 1995), this second surge in neural network research experienced stagnation during the mid-1990s; however, in the 2000s, neural networks (and the research field of artificial intelligence as a whole) experienced a third surge, and neural networks have been a field of highly intense research for more than the last decade.

The basic theory behind artificial neural networks is comparatively simple:

- an vector of input features is fed into the *input layer* of the network;
- the data from the input vector is then transformed through one or multiple *hidden layers*;
- finally, an *output layer* interprets the transformed data and returns it as the output of the network

The in- and outputs of the network can represent a variety of types of data, from simple numerical data over audio or video data to more abstract types of data such as class labels of objects or even abstract vectors containing a sentence's meaning (as will be discussed later). During the transformation of the data, each layer iteratively receives the data of the previous layer, performs a series of linear as well as non-linear operations on it, and passes it on to the next layer. As layer k transforms the data, each neuron n_i^k receives data from (usually, but not always all) neurons of the previous layer $i-1$ or the input layer if $i=1$, computes a weighted sum of those values, and applies a non-linear *activation function* to it. Subsequently, the $i+1$st layer receives data from the ith, until the output layer is reached and an output is produced. Mathematically, the value that n_i^k propagates to the next layer is defined as[1]:

$$a_i^k = \sigma_k(z_i^k) \; = \; \sigma_k(\sum_{a_j^{k-1} \in A^{k-1}} w_{ji}^k \cdot a_j^{k-1} + b_j^k), \tag{2.6}$$

where

- a_i^k is the *activation* of the ith neuron in the kth layer, i.e. the value it passes on to the next layer,
- z_i^k is the weighted sum the neuron n_i^k "receives" from the previous layer,
- A^k is the set of all neurons of the kth layer,
- w_{ji}^k is the weight of the activation a_j^{k-1} for the calculation of the activation of the current neuron a_i^k
- b_j^k is the *bias* of the n_i^k, a single value that, among other things, enables individual neurons to better center the data they process, and
- σ is the *activation function* on layer k.

This activation function, for which a non-linear function such as the sigmoid or the hyperbolic tangent is chosen, is the key component that enables artificial neural networks to learn not only *linearly separable* problems (as they would if the activation function was a linear one), but non-linear ones as well. A common intuition to understand this property is to view every neuron's output as a *folding* of the previous layer's parameter space; thus, instead of having to perform a linear decision on an unprocessed data input, the output layer performs decisions on a heavily morphed version of the input data, allowing for significantly more complex decision boundaries.

In practice, this is often visible in the form of increasingly complex data features being constructed with later neuron layers. 2.1 shows such a hierarchical construction of features in the image processing domain: while lower-tier layers fold the image data in such a way that, for example, horizontal or vertical edges can be recognized, higher-tier layers use this preprocessed data to recognize elements of the input symbols, and, eventually, even the symbols themselves.

It is noteworthy that neural networks are a comparatively generalistic approach to problems like classification and prediction: a neural network can be trained for any function between two data sets, as long as they can be meaningfully encoded in numerical vectors.

[1] In the following equations, it is assumed that the layers of the network are *fully connected*, i.e. a neuron receives data from *all* neurons of the previous layer.

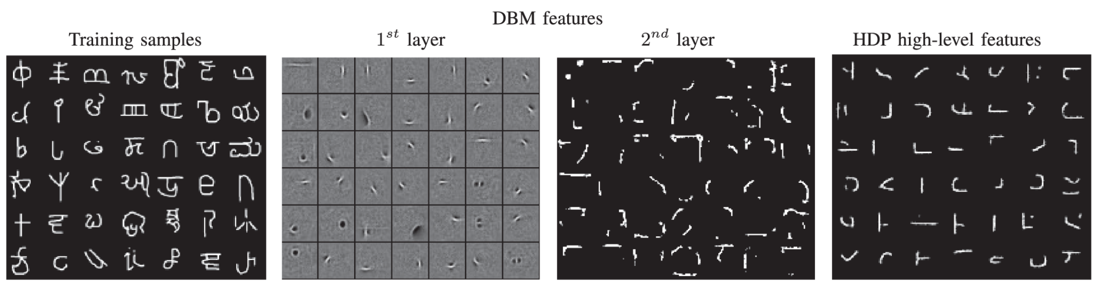


Figure 2.1: A visual representation of the features learned by neurons of different layers for a symbol recognition task. (Salakhutdinov, Tenenbaum, and Torralba, 2013)

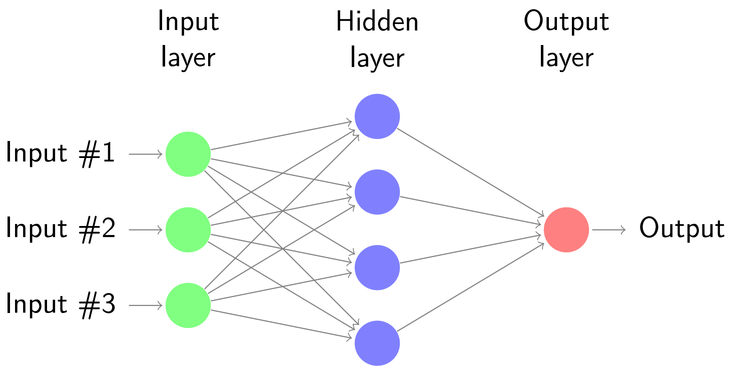


Figure 2.2: A simple feed-forward neural network.

2.2.2 Learning & Backpropagation

Similar to the joint sequence models presented in the previous chapter, neural networks are a purely data-driven approach to machine learning. From a set of training data, a system is taught - or, more accurately, teaches itself - to guess the correct output for any given valid input.

To begin with, a network's parameters (i.e. the weight vectors W_k and the bias vectors B_k) are initialized. While this often occurs by simply setting every weight factor to a small random number[2] and the biases to zero, a number of more complex initialization methods exists.

Subsequently,

1. individual data samples (or batches of samples) are fed into the network;
2. after each such training iteration, the expected outputs from the data set are compared to the outputs produced by the network;
3. a *loss* is calculated as a measure of the network's inaccuracy in producing the correct outputs;
4. the network parameters are changed in such a way as to reduce the expected error on the same samples in the future.

Different loss functions exist for different applications, from a simple metric such as the *mean squared error* for simple numeric prediction outputs, to any number of arbitrarily complex functions, as long as they adequately represent a model's performance on the task at hand. There are some requirements for a loss function to be utilized in combination with different training algorithms, such as the differentiability of the function if it is to be used in conjunction with the *gradient descent* training algorithm, but they will not be discussed here.

The above process is repeated with other samples or batches of samples until a termination condition is fulfilled, such as an acceptable loss, converging parameters or loss, or a maximum number of iterations.

There is a multitude of approaches to implement the fourth step, the update of the network parameters for a given loss. Below, one of the most common - *gradient descent* - will be introduced and on this example, the concept of *backpropagation* will be explained.

In a neural network, each neuron's *activation* a_i^k, i.e. the value it passes on to the next layer, has a calculable impact on the overall loss for the current batch of samples m:

$$\frac{\partial L(m)}{\partial a_i^k} = \sum_{z_j^{k+1} \in Z^{k+1}} w_{ij}^{k+1} \cdot \sigma_k'(z_j^{k+1}) \cdot \frac{\partial L(m)}{\partial a_j^{k+1}}, \tag{2.7}$$

[2]There are at least two important reasons to use small random numbers: *small* initial weights allow for a faster initial learning that larger, more slowly adaptable weights; and *random* initial weights prevent a learning deadlock, in which each neuron of a layer processes and outputs the exact same value for a given input.

that is, its impact is equal to the sum of the error impact of the neurons in the next layer weighted by the current neuron's influence on their activation, w_{ij}^{k+1}.

The neurons' activations itself are not an adjustable parameter of the model - they are simply the results of the inputs and the actual model parameters -, but from a neuron's impact on the batch error $\frac{\partial L(m)}{\partial a_i^k}$, the impact of its *incoming weights* from the previous layer, i.e. w_{ji}^k, can be calculated:

$$\frac{\partial L(m)}{\partial w_{w_{ji}^k}} = \frac{\partial L(m)}{\partial a_i^k} \cdot \sigma_k'(z_i^k) \cdot a_j^{k-1}, \tag{2.8}$$

that is the impact the activation of n_j^{k-1} had on the activation of the neuron n_i^k and, through this, on the overall sample batch loss $L(m)$.

With this information, it is possible to adjust the individual *weights* w_{ij} of the neural network in such a way as to reduce the expected error for future outputs for this batch's inputs:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{1}{|m|} \cdot \frac{\partial L(m)}{\partial w_{w_{ji}^k}}, \tag{2.9}$$

where $0 < \eta \leq 1$ is the *learning rate* of the system. This learning rate is one of a network's main training hyperparameters; an overly large (i.e. close to 1) learning rate may lead to issues such as a non-convergent training process (as the weights "oscillate" around individual training samples), while an overly small learning rate may lead to exceedingly slow model training speeds and convergence on *local* instead of *global* optima for the network parameters (Jacobs, 1988).

The loss gradients $\frac{\partial L(m)}{\partial a_i^k}$ and $\frac{\partial L(m)}{\partial w_{w_{ji}^k}}$ can be computed efficiently utilizing the *backpropagation* algorithm. As both loss gradients are dependent only on values from the next layer, they can be computed iteratively from the last layer of the network back to the first, starting with the output layer, the activations of which of course have a *direct* impact on the sample batch loss, as the loss function is defined on the outputs of the network, i.e. the activations of the neurons of the last layer. From these known values $\frac{\partial L(m)}{\partial a_i^{output}}$, the loss impact of the individual layers can be computed backwards up to the input layer; the loss is *propagated*.

2.2.3 Recurrent Neural Networks

The basic neural network architecture introduced in the previous subchapter, while versatile and quite effective for a considerable range of problems, has several significant drawbacks. One of them - and one that is quite relevant in the context of G2P conversion - is the poor ability of these networks, also known as *feed-forward neural networks*, to process sequential data. While a feed-forward neural network may be taught to recognise certain features (such as a human face) in the data of a *single* image, there is no effective way of feeding a *sequence*

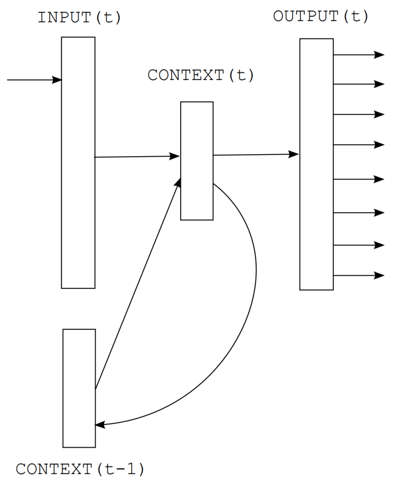


Figure 2.3: A schematic view of a simple RNN. (Mikolov et al., 2010)

of images into it and e.g. teaching it to track an object, as both the in- and output-layers are of a pre-determined size and expect the same number of input pixels and output labels for each training sample. This is obviously a massive limitation to the applicability of simple feed-forward neural networks for many problems, including sequence-to-sequence association problems such as grapheme-to-phoneme conversion.

A relatively simple extension of the basic FFNN approach allows neural networks to be applied to this kind of problem: that of *recurrence*.

A *recurrent neural network* (RNN) implements the same fundamental concepts as a FFNN, but adds recurrent connections to each of the layers. Instead of inputting exactly one input vector per sample into the network, an arbitrary number of n inputs can be inserted sequentially; after each input, an output is produced and additionally, each neuron receives *the activations of all neurons in its layer from the previous time step* as an additional input. The equation for the activation of neuron n_i^k at time t, $a_{i,t}^k$, is therefore an updated version of the equation for FFNNs:

$$a_{i,t}^k = \sigma_k(z_{i,t}^k) \;=\; \sigma_k(\sum_{a_{j,t}^{k-1} \in A^{k-1}} w_{ji}^k \cdot a_{j,t}^{k-1} + \sum_{\boldsymbol{a_{j,t-1}^k \in A^k}} \boldsymbol{w_{ji}^k \cdot a_{j,t-1}^k} + b_j^k) \tag{2.10}$$

This allows RNNs to process sequences of input vectors while retaining some latent information over multiple timesteps (such as the current position of an object in an image, or the current part of speech for NLP applications).

2.2.4 Backpropagation Through Time

The addition of recurrent connections into RNNs does, of course, require changes to be made to the backpropagation algorithm as well. These changes, however, are surprisingly small and intuitive. In fact, almost no changes are made to the loss backpropagation equations at all;

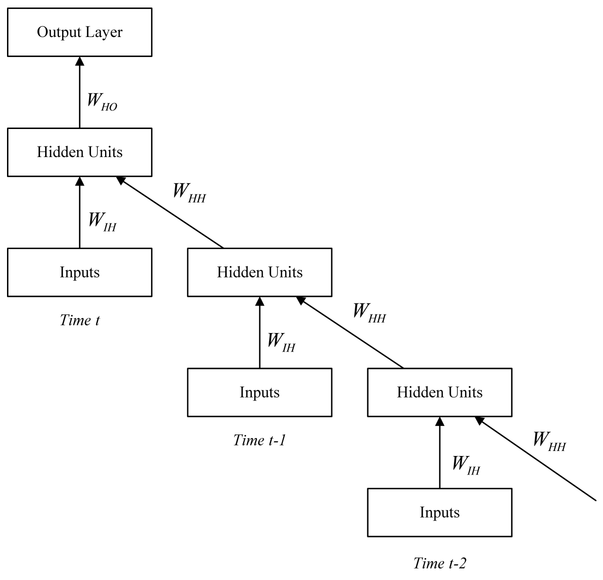


Figure 2.4: A schematic example of backpropagation through time. (Ma et al., 2015)

instead, a new preprocessing step is added: that of *unrolling* the network over the length of the current training sample. Effectively, if the training sample had a length of T individual inputs - and thus, the network went through T different internal states and produced a sequence of T outputs -, the whole network is copied T times, and the in- and output-layer of each of the T timestep-networks is connected with the previous one's output and the next one's input, respectively. Figure 2.4 shows a simple example of this process.

Since the result is fundamentally a standard FFNN that simply has T as many layers as the original RNN, the backpropagation algorithm introduced earlier can easily be applied to find the loss impact of each timestep-specific neuron, $\frac{\partial L(m)}{\partial n_{i,t}^k}$, and so, the loss impact of each timestep-specific weight $w_{ij,t}^k$. Finally, each weight w_{ij}^k of the RNN is updated by averaging the changes suggested by each time-specific edge in the unrolled network that corresponds to w_{ij}^k in the original RNN:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{1}{T} \cdot \sum_{1 \leq t \leq T} \left[\frac{\partial L(m)}{\partial w_{w_{ji,t}^k}} \right] \tag{2.11}$$

Although RNNs are fundamentally capable of processing data sequences instead of only data vectors as FFNNs do, there are still several drawbacks to their use that will be addressed in the next subchapters.

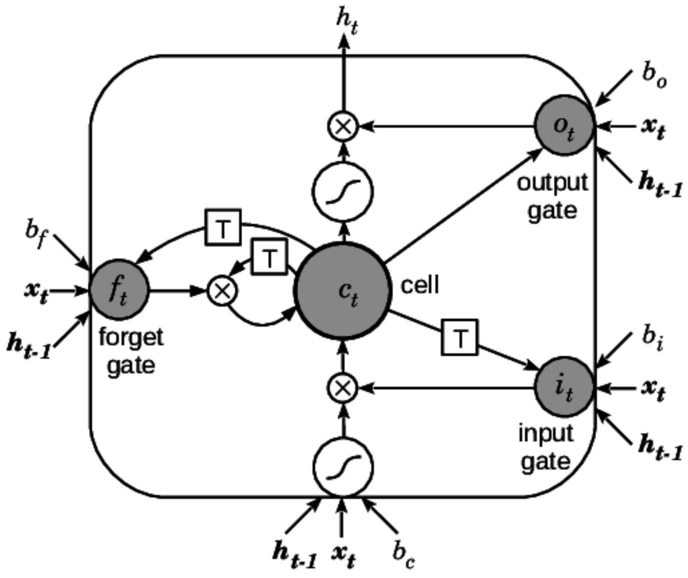


Figure 2.5: A schematic representation of a basic LSTM cell. (Marchi et al., 2017)

2.2.5 Long Short-Term Memory Networks

An issue that affects any moderately *deep* neural network, i.e. one with a sufficiently high number of layers, is the so-called *vanishing gradient problem*[3]. As many of the activation functions σ in use for neural networks have gradients in the area $(-1, 1)$ or $(0, 1)$, and this gradient is a factor in the equation for the loss impact of a neuron n_i^k, neurons in the earlier layers (i.e., closer to the input layer) are subjected to exponentially small, or *vanishing* change during each training iteration. As RNNs are "unrolled" into long chains of FFNNs during training, they are especially susceptible to this vanishing gradient problem.

One possible solution that has received massive attention over the last two decades is the concept of *Long Short-Term Memory* networks. LSTM neurons are fundamentally the same as RNN neurons; however, several significant additions have been made to the neuron's behaviour:

- an **input gate** $i_{i,t}^k$ that determines whether or not the incoming data at timestep t should influence the neuron's next state or simply be ignored,
- a **forget gate** $f_{i,t}^k$ that determines whether the neuron's last state at timestep $t-1$ should influence its current state or simply be ignored, and
- an **output gate** $o_{i,t}^k$ that determines whether the neuron's current state at timestep t or a value of zero should be used to calculate the activation $a_{i,t}^k$ propagated to the next layer.

[3]A similar, symmetrically opposed issue called *exploding gradient* exists, which will not be discussed here.

The equations for an LSTM neuron are then[4]:

$$\begin{aligned}
i_{i,t}^{k} &= \sigma_g\left(\sum_{a_{j,t}^{k-1}\in A^{k-1}}\left[w_{ji,inp}^{k}\cdot a_{j,t}^{k-1}\right] + \sum_{a_{j,t-1}^{k}\in A^{k}}\left[w_{ji,inp}^{k}\cdot a_{j,t-1}^{k}\right] + b_{i,inp}^{k}\right)\\
f_{i,t}^{k} &= \sigma_g\left(\sum_{a_{j,t}^{k-1}\in A^{k-1}}\left[w_{ji,forg}^{k}\cdot a_{j,t}^{k-1}\right] + \sum_{a_{j,t-1}^{k}\in A^{k}}\left[w_{ji,forg}^{k}\cdot a_{j,t-1}^{k}\right] + b_{i,forg}^{k}\right)\\
o_{i,t}^{k} &= \sigma_g\left(\sum_{a_{j,t}^{k-1}\in A^{k-1}}\left[w_{ji,out}^{k}\cdot a_{j,t}^{k-1}\right] + \sum_{a_{j,t-1}^{k}\in A^{k}}\left[w_{ji,out}^{k}\cdot a_{j,t-1}^{k}\right] + b_{i,out}^{k}\right),
\end{aligned} \tag{2.12}$$

where

- W_{inp}^{k}, W_{forg}^{k} and W_{out}^{k} are the weight matrices for the input, forget and output gates for layer k, respectively,
- B_{inp}^{k}, B_{forg}^{k} and B_{out}^{k} are the bias vectors for the input, forget and output gates for layer k, respectively, and
- σ_g is the sigmoid function, a function with an output range of $[0, 1]$, i.e. *closed gate* or *opened gate*.

With the values of these three gates calculated, the state of neuron n_i^k at step t is calculated as:

$$z_{i,t}^{k} = f_{i,t}^{k}\cdot z_{i,t}^{k} + i_{i,t}^{k}\cdot\sigma_c\left(\sum_{a_{j,t}^{k-1}\in A^{k-1}}\left[w_{ji}^{k}\cdot a_{j,t}^{k-1}\right] + \sum_{a_{j,t-1}^{k}\in A^{k}}\left[w_{ji}^{k}\cdot a_{j,t-1}^{k}\right] + b_{i}^{k}\right) \tag{2.13}$$

And with this, the output of the neuron n_i^k at step t is:

$$a_{i,t}^{k} = o_{i,t}^{k}\cdot\sigma_c(z_{i,t}^{k}), \tag{2.14}$$

where σ_c is the activation function of the neuron, often the hyperbolic tangent.

LSTM networks have several advantages over over traditional RNNs: due to the fact, if the unit outputs any value at all (and thus contributes to the sample loss), the value of the output gate will usually be close to 1; this largely prevents the vanishing gradient problem from occurring. Additionally, a LSTM neuron can relatively easily learn to store information *for many timesteps*. Thus, LSTMs have little problems with long-distance dependencies, which occur, among other application domains, in many NLP problems (examples of this include natural language text relationship extraction (Sutton and McCallum, 2006), natural language dependency parsing (Imamura, Kikui, and Yasuda, 2007) and machine translation (Hai Son, Allauzen, and Yvon, 2012)).

[4]The equations stem from the original work on LSTMs (Hochreiter and Schmidhuber, 1997) and were adapted to be consistent with the notations established in the previous subchapters.

2.2.6 Encoder-Decoder Architecture

While LSTMs solve some of the issues of traditional RNNs, at least one major problem remains: any RNN introduced so far produces an output sequence that is *exactly as long as the input sequence*; there is simply no formalism for the network to either stop early and ignore further inputs, or continue producing output steps after the full input sequence has been inserted. While this constriction fits some NLP problems (such as sequence labeling or Part-of-Speech-tagging),it is a gross mismatch with the reality of others, where cases in which translated sentences are of a significantly different length than the original input, or word pronunciations longer or shorter than the input word, are extremely common. While there are workaround-solutions such as padding either of the input- or output-sequences, none of them are optimal.

In 2014, a novel solution for this issue was proposed: the *encoder-decoder* architecture for sequence-to-sequence learning (Cho et al., 2014a). The fundamental idea is this: instead of employing a single RNN that produces outputs at the same time as inputs are fed into the system, *two* networks are used: an *encoder* network that receives the input sequence and encodes it into a compressed embedding vector similar to those introduced in more detail in section 2.3; and a *decoder* network that, after the full input sequence has been encoded, uses this dense representation of the input sequence's data - and, at each timestep, its own output from the previous timestep - to produce an output sequence. It is not uncommon that, instead of using an *explicit* embedding vector that is propagated from the encoder to the decoder, both encoder and decoder are constructed with the *hidden-layer architecture* and the *last hidden state* (i.e. internal state of all neurons in the encoder) is simply transferred into the decoder as *the first hidden state of the decoder* (Cho et al., 2014a).

This means that instead of being required to generate an output sequence with a length matching that of the input sequence, the system can generate an output sequence of *arbitrary length*; and, instead of producing an output from the moment the first input vector is fed into the system, the model can assess the information content of the *full* input sequence *before producing a single output*.

The training of this network architecture is, again, somewhat similar to that of traditional RNNs. The decoder is unrolled back to its first iteration (i.e. the point at which it received the embedding vector from the encoder). This is the exact same as the last iteration of the encoder (which, at this point, passed the embedding vector to the encoder); so, from this point, the encoder network is unrolled back towards the individual input vectors. Subsequently, the standard backpropagation algorithm can be employed to calculate both networks' parameter updates.

2.2.7 Attention Mechanism

The encoder-decoder-architecture's approach of using an encoding vector to transfer the encoder's interpretation of the input sequence to the decoder puts considerable pressure on the encoding mechanism: it assumes that all relevant information of a potentially arbitrarily long input sequence can be compressed into a single vector of fixed dimension. To alleviate this, a comparatively recent approach to sequence-to-sequence learning proposes an additional

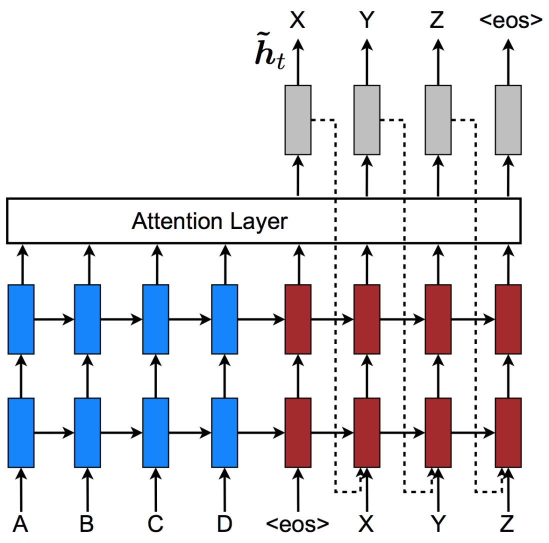


Figure 2.6: A diagram of an attention-based encoder-decoder network. In this case, both encoder states (blue) and decoder states (red) are, through an attention layer, jointly fed into an output layer (grey). (Luong, Pham, and Manning, 2015)

system component: the *attention mechanism* (Luong, Pham, and Manning, 2015; Mnih, Heess, and Graves, 2014).

The concept of the attention mechanism is this: instead of giving the decoder access to only the last output of the encoder network, it gains access to *every single output along the way*, averaged to produce a single, attention-weighted context vector. This allows the decoder to, depending on its current state and its last output, pay special attention to single vectors in the input sequence. This attention averaging mechanism itself is a trainable neural network, allowing the system to learn when to pay attention to which part of the input sequence. This allows the model to learn even more complex semantical and syntactic connections for a NLP-related task.

2.2.8 Beam Decoding

One last problem with RNNs that needs to be addressed is that of *local optima*. As, from the first timestep on, the system's output depends *on its own output from the last timestep* (for the encoder-decoder architecture, the decoder feeds its own output into itself as an input for the next timestep), standard RNNs have limited chance of producing the *optimal* output sequence: if o_1 is the first output generated by the system for the current input, any output sequence produced for this input will start with o_1; there is no possible output o' where $o'_1 \neq o_1$, even if that output o' were significantly better than the one actually produced. *Beam Decoding* using beam search partially alleviates this issue.

In classical search problems, *Beam Search* is an approach designed to reduce the amount

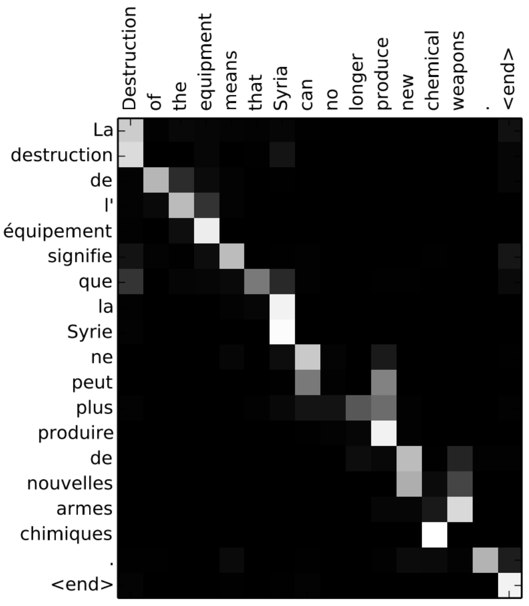


Figure 2.7: A visualization of the attention vector for a machine translation application (Bahdanau, Cho, and Bengio, 2015)

of hypotheses for possible partial solutions stored at any time. While search algorithms such as breadth-first search (BFS) or the all-present A* algorithm guarantee finding an optimal solution, they will require (or in the case of A* degenerate to require) the storage of b^d different partial solutions, with b being the *branching factor* of the graph and d being the current search depth into the graph. In other words, the number of stored partial solutions is *exponential* in the length of the solution.

Beam Search compensates for this potential problem by limiting the overall amount of tracked partial solutions at any time to a fixed number k. Each step, these k *solution prefixes* are employed to generate a subsequent set of partial solutions; from these, in turn, the algorithm selects and continues on the k best. While Beam Search loses the property to guaranteedly output an optimal solution in contrast to BFS or A* search, it allows an algorithm to track only a *constant* instead of an *exponential* number of partial solutions at each step.

In the context of Encoder-Decoder G2P conversion, this means the following: instead of selecting only the best (i.e. most likely) phonemization for an input grapheme at each step for continuation, the k most likely phonemes are selected and each used in a dedicated decoder instance to produce, in total, $k \cdot |v_p|$ phonemization hypotheses (with v_p being the phoneme vocabulary).

From these $k \cdot |v_p|$ hypotheses, only the k best partial solutions (i.e. those with the highest compound likelihood

$$\Pi_1^i \, p(\Phi_i | \Phi_{i-1}, v_c) \tag{2.15}$$

as estimated by the decoder for context vector v_c step i) are kept and, unless the last generated

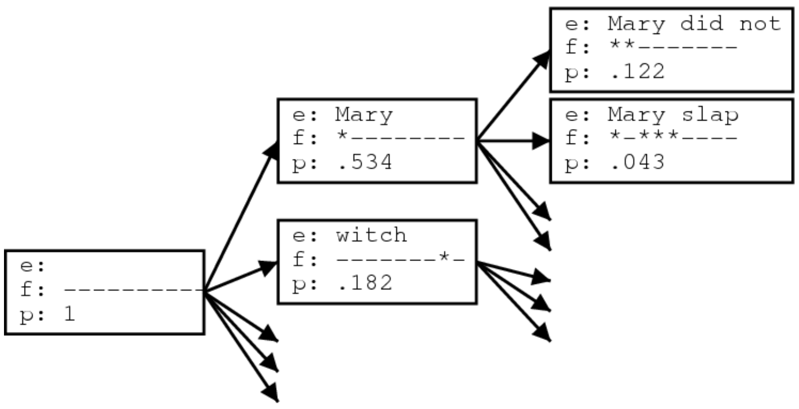


Figure 2.8: A simple example of beam search for a machine translation task with beam width 2. (Koehn, 2004)

phoneme is the end-of-sequence symbol, their respective last phonemes fed back into a new decoder each for the next decoding step). In effect, the system maintains k distinct encoder instances until a termination condition (such as maximum phonemization length, a compound likelihood threshold or termination, i.e. output of the termination symbol for all instances, is reached, after which the best overall solution is selected for output.

With individual output symbol probabilities naturally lying in the area $[0, 1]$, longer output candidate sequences are inherently at a disadvantage. To counteract this preference for shorter output sequences, the log probability of each candidate sequence is normalized by the sequence length (Cho et al., 2014b).

2.3 Dense Symbol Embeddings

Instead of plain symbol *tokens*, i.e. a unique, machine-readable representation for each input or output symbol (such as a unique integer ID or a one-hot encoded vector over the total input- or output-vocabulary, respectively), *dense symbol embeddings* will be used in the in- and output-layers of the LSTM-based G2P approaches. Dense symbol embeddings are a remarkable representation of semantical units in NLP applications that have experienced a massive growth in attention over the last decade.

The analogous word-level concept are *dense word embeddings*. The origin of the idea of word embeddings lies in the field of *distributional semantics*. Without delving too deeply in the theory of this linguistic field, the fundamental idea behind distributional semantics is this: a semantical unit (such as the word of a natural language) can to a great degree be characterized by its surrounding context, or, in the words of one of the first proponents of distributional semantics:

"You shall know a word by the company it keeps."
- J.R. Firth, 1890-1960

Classically, the first computational examples of distributional semantics calculated a *co-occurrence matrix* for a given text corpus, calculating the relative or absolute frequency of co-occurrence for each pair of distinct words in the corpus in a given frame (such as a sentence or a maximum in-sentence distance of n words). Each row of this matrix, corresponding to a single word of the corpus, represents this word's compiled contextual information in a high-dimensional space (n-dimensional, with n being the number of distinct words in the corpus). It was quickly realized that these *dense* word vectors have remarkable properties: among others, semantical word clustering based on the co-occurrence vectors can be performed to a certain degree, as pairs of words with similar meaning tend to have a low cosine distance in their respective vectors, and vice versa, and some arithmetical operations can be performed on them, one of the most famous being

$$v(king) - v(man) + v(woman) = v(queen).$$

To quote K. Lund and C. Burgess, two of the original researchers of this new field of NLP: *"The implication of these results is important. The vectors generated [...] function semantically. [...] The concepts of coffee and tea are similar and strongly associated. They tend to co-occur in natural language, and their similarity can be seen in their vector representations. These can be contrasted to road and street. Again, although these are two concepts that are highly similar, they do not tend to co-occur in usage, yet their vector representations are very similar."* (Lund and Burgess, 1996)

Despite these somewhat useful properties, occurrence-matrix-based word representations present a number of issues, two significant of which are the matrices' tendency to grow increasingly sparse with growing vocabulary, and the exponential growth of the matrices' entries in regards to the corpus size.

One suggested solution to these issues was the introduction of *dense word embeddings*, denser counterparts to the word vectors generated via co-occurrence matrices. While the exact implementation of these embeddings varies for different approaches, the general concept remains the same: the generation of a lower-dimensional, more densely populated counterpart to the classical word vectors, while retaining their rich information and interesting arithmetic properties.

The embedding approach that will be used in this thesis is that of neural network based feature compression. Made popular in no small part by the *word2vec* implementation by Mikolov et al. in 2013 (Mikolov et al., 2013), this approach utilizes a neural net with a single hidden layer that is trained to either predict the surrounding context for a given word, or predict a single word given a surrounding context. In either case, the key component of this approach is the NN's hidden layer, limited to a number of hidden units k significantly lower that the overall vocabulary word count; the idea is that, if the NN performs adequately on the word-/context-prediction task even when constricted to the use of the k hidden units, then these k units must be able to store sufficient latent information to emulate the explicit contextual information of each word's full co-occurrence vector - in other words, the activation vector of the NN's hidden layer for a word must be at least roughly equivalent to the word's full co-occurrence vector in terms of expressiveness.

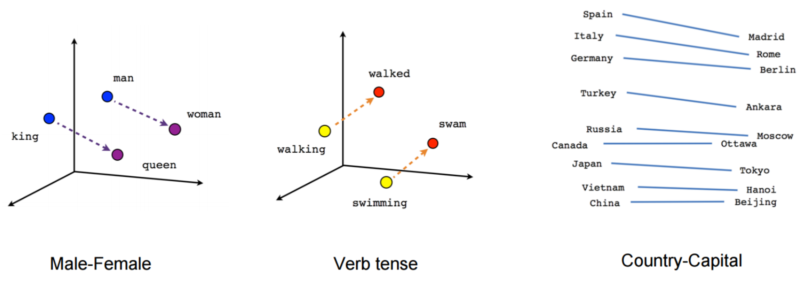


Figure 2.9: Some of the properties of word embeddings, projected onto a low-dimensional space (Collis, 2017)

This assumption of equivalence has been confirmed for various domains. In this thesis, a similar approach on a symbolic level, i.e. of *dense grapheme embeddings*, will be utilized in two ways:

- as an additional laye for the LSTM-based G2P system that is inserted between the input layer and the first hidden layer; this embedding layer is trained as part of the overall training of the system, and
- as an advanced metric of similarity of words for some active learning approaches.

A challenge in the utilization of symbol embeddings will be the compilation of a words *individual symbol embeddings* into a *word embedding*; since generating embedding vectors for the individual symbols of a word will produce embedding sequences of varied length for different words, this process is far from trivial. Again, this is a topic of sufficient size to exceed the frame of this thesis. Therefore, a simple averaging of the individual symbol embeddings will be used as a starting point. This approach "has proven to be a surprisingly successful and efficient way of [sequence] embeddings" (Kenter, Borisov, and Rijke, 2016) and should provide an adequate baseline approach. (Faruqui et al., 2014; Yu et al., 2014; Gershman and Tenenbaum, 2015; Kenter and De Rijke, 2015)

2.4 Active Learning

"The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator). Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain." - Burr Settles (Settles, 2010)

The fundamental idea behind *active learning* is to enable a machine learning algorithm to *request help* from an oracle in regards to labeling data. Traditionally, active learning is employed in applications where only *partially labeled data* or fully *non-labeled data* is available for training: instead of requiring an oracle to label all samples in a data set, the system itself decides which subset of the data to label, depending on its own estimation of factors such as the *expected gain in accuracy* or the *expected change of the system's parameters*. Active learning has been successfully applied to a broad range of problems and machine learning approaches, including natural language processing applications.

While the goal of traditional active learning applications is not identical with the main goal of this thesis - selecting samples for manual labeling to *reduce error at application time* instead of to maximize the training progress from these samples - the hypothesis of the following chapters is that for many active learning approaches, these goals are similar enough to be able to transfer some of the ideas form the latter to the former. As a consequence, any application-time requesting approaches determined to be successful in this thesis should *also* be applicable to classical train-time active learning for the G2P domain.

The approaches selected to test this hypothesis are presented in Chapter 3 and evaluated in Chapter 5.

3 Request Decision Approaches

In this chapter, the selected active learning approaches and their specific application to the two G2P systems are presented.

3.1 Random-Request Approach

An obvious choice for a naive baseline approach for the request decision process is that of *random requesting*. In this approach, a fixed probability is given for any grapheme input to be sent for manual requesting to the oracle, regardless of any potentially decision-relevant information the G2P system may provide.

The expected accuracy gain for this approach is a *linear* one in respect to the request probability (and thus, the relative number of phonemizations requested):

For a request probability $0 \leq p \leq 1$ and a test set N, the expected number of requested manual labelings R is:

$$|R| \approx p \cdot |N| \tag{3.1}$$

Furthermore, for any sufficiently large randomly selected $|R|$, the relative number of phonemization errors contained in this subset can be expected to be equal to that of the total test set N:

$$\frac{|R_{errors}|}{|R|} \approx \frac{|N_{errors}|}{|N|} \tag{3.2}$$

Since we assume that a manual labeling request returns a perfectly labeled, i.e. error-free phonemization, the number of phonemization errors remaining for a random request probability of p is:

$$\begin{aligned} Errs(p) &\approx |N_{errors}| - |R_{errors}| \\ &\approx |N_{errors}| - p \cdot N_{errors}, \end{aligned} \tag{3.3}$$

i.e. a linear function dependent on the random request probability p.

3.2 Confidence-Based Approaches

An important metric of any classification, prediction or otherwise labeling system is the system's *confidence* in any generated output. Many machine learning approaches, however, do not offer a natural measure of their confidence; rather, values like an output candidate's *probability* (as is the case for the Sequitur-based G2P system) or its *neural activation* is produced and the most likely candidate chosen for output. Therefore, the approaches chosen for Sequitur G2P and LSTM-based G2P in this thesis are to be understood as *reasonable interpretations* of the systems' outputs and output metadata rather than a true, natural confidence metric. In this section, three prospective confidence metrics for both G2P approaches and their potential usages in request decision are proposed.

The joint sequence modeling approach of the Sequitur G2P system offers a rather natural metric of confidence, proposed by the authors themselves: the *posterior probability of a phonemization ϕ given an input word g*: (Bisani and Ney, 2008)

$$p^{seq}(\phi|g) = \frac{\sum\limits_{q \in S(g,\phi)} p^{seq}(q)}{p^{seq}(g)}, \tag{3.4}$$

where the probability of an input word is approximated by the sum of the probabilities of all possible phonemizations $\phi \in \Phi^*$: (Bisani and Ney, 2008)

$$p^{seq}(g) = \sum_{\phi \in \Phi^*} p^{seq}(g,\phi) = \sum_{q \in Q^* | g(q) = g} p^{seq}(q), \tag{3.5}$$

where Q^* is the set of all possible graphone sequences and $g(q)$ is the grapheme part of a graphone sequence.

Thus, $p(\phi|g)$, while not being a true measure of confidence, is an expression of the Sequitur system's statistical support from the training data to propose the phonemization ϕ for the given word g. We therefore define the confidence of the Sequitur-based G2P system in a produced phonemization as:

$$c^{seq}(g,\phi) = p^{seq}(g,\phi) \tag{3.6}$$

For the LSTM-based G2P system, a similarly obvious choice for a confidence metric is available: the *total likelihood* of the *best beam* result. In our system, the likelihood of a single output symbol ϕ_t at timestep t is defined as the *softmax value* of this symbol's activation over all activations of the output layer (where $a^o_{i,t}$) is the activation of the ith output neuron at timestep t) :

$$p^{lstm}(g,\phi_{i,t}) = \frac{e^{(a^o_{i,t})}}{\sum\limits_{\phi_j \in \Phi} e^{(a^o_{j,t})}} \tag{3.7}$$

The total likelihood of a beam result ϕ^k is then simply the product of the individual symbol likelihoods, normalized by the length of the output sequence (as previously discussed):

$$p^{lstm}(g,\phi^k) = \frac{\prod\limits_{1 \le t \le |\phi^k|} p^{lstm}(g,\phi^k_t)}{|\phi^k|} \tag{3.8}$$

Again, we define the system's confidence in a phonemization as this probability:

$$c^{lstm}(g,\phi) = p^{lstm}(g,\phi) \tag{3.9}$$

These "confidence" metrics, $c^{seq}(g,\phi)$ and $c^{lstm}(g,\phi)$, do not take any phonemization beyond the most likely into account. Therefore, two additional measures of "confidence" will be evaluated: the *probability margin* between the two most likely phonemizations (hereafter referred to as "confidence margin"), and the *probability entropy* over all possible phonemization candidates (hereafter referred to as "confidence entropy").[1]

The *confidence margin* $p_{marg}(g,\phi)$ of a phonemization output ϕ for a word g is defined as the difference between the confidences of two most likely phonemizations ϕ_1, ϕ_2:

$$c_{marg}(g,\phi) = c(g,\phi_1) - c(g,\phi_2) \tag{3.10}$$

This produces the following two calculations for our Sequitur- and LSTM-based systems, respectively, where K is the number of beams used in the LSTM system:

$$\begin{aligned} c^{seq}_{marg}(g,\phi) &= \underset{\phi' \in \Phi^*}{\operatorname{argmax}}\, c^{seq}(g,\phi') - \underset{\phi' \in (\Phi^* \setminus \phi_1)}{\operatorname{argmax}}\, c^{seq}(g,\phi') \\ c^{lstm}_{marg}(g,\phi) &= \underset{1 \le k \le K}{\operatorname{argmax}}\, c^{lstm}(g,\phi^k) - \underset{k \in ([1,..,K] \setminus k_{best})}{\operatorname{argmax}}\, c^{lstm}(g,\phi^k) \end{aligned} \tag{3.11}$$

This metric will hopefully support a request-decision component in differentiating situations in which the produced phonemization is *moderately likely, but significantly better than any alternatives* against those in which the phonemization is *moderately likely, but only marginally better than at least one alternative*.

As an extension of the confidence margin, the *confidence entropy* $p_{entr}(g,\phi)$ of a phonemization output ϕ for a given word g is defined. This is similar to (Kominek and Black, 2006), where an entropy function over Markov chain emission probabilities is used in the context of traditional active-learning-based training of a G2P system.

$$\begin{aligned} c^{seq}_{entr}(g,\phi) &= 1/ - \sum_{\phi' \in \Phi^*} c^{seq}(g,\phi') * \log^{seq}(c(g,\phi')) \\ c^{lstm}_{entr}(g,\phi) &= 1/ - \sum_{1 \le k \le K} c^{lstm}(g,\phi^k) * \log(c^{lstm}(g,\phi^k)), \end{aligned} \tag{3.12}$$

i.e. the inverse of the information-theoretical *entropy* over the probabilities of all potential phonemization candidates. This should give a request decision component a better impression of the ambiguousness of the G2P component's output.

[1] The beam search approach for our LSTM-based system does, of course, not calculate the likelihoods of *all possible phonemizations*, but only that of the the k best candidates.

3.3 Ignorance-based Approaches

"Ignorance represents the distance of a new query point from the training samples seen so far." - Edwin Lughofer (Lughofer, 2012)

In 2012, a series of new ideas for the field of Active Learning was proposed by Edwin Lughofer. Among them, the idea of *request by ignorance* the most promising one for use in this thesis.

The fundamental idea behind *request by ignorance* is that a system utilizing active learning techniques should request labeling for those samples that are the *most different* from those already known and labeled. While the definition of the distance of a single sample to the set of known ones is far from trivial in many cases, it is especially difficult in the context of grapheme-to-phoneme conversion, as there are no inherent numeric attributes attached to any single symbol or words; and as such, the definition of a distance metric between *two individual words* as well as between *an individual word and a group of words* poses a considerable challenge.

In this thesis, multiple different approaches to this problem are implemented:

The **total grapheme sequence likelihood** of a given input word for the Sequitur-based G2P model. This value is independent of any chosen output phonemization and represents the system's training-data-based estimation of the *total* likelihood of the input sequence occurring.

For the LSTM-based approach, the cosine distance of an input word's average embedding vector within the network (i.e. of the average of the activation vectors of the embedding layer over the timesteps) to the overall average of all training dataset embeddings will be used as a metric for the system's ignorance regarding that word; additionally, the same metric will be computed for the average of the network's hidden layer activations.

3.4 Word-Origin-based Approach

In their recent paper, Milde, Schmidt, and Köhler (2017) suggest that, for German G2P systems, "loan words and names with predominately English pronunciation, [... and] the same for French" pose a particular challenge due to their different morphology.

As described in Chapter 5, early tests for these word classes showed a diminishing impact on the test data used, and so no further effort into a sophisticated application of this fact in the request decision component was made. It is noteworthy, however, that the detection of a word's language has seen successful research in the last years, and so approaches for this purpose exist, such as detecting Anglicisms in German (Leidig, Schlippe, and Schultz, 2014b) and Afrikaans (Leidig, Schlippe, and Schultz, 2014a) text or the detection of English words in Hindu social media (Das and Gambäck, 2014).

3.5 Error Predictor based Approaches

In the previous subchapters, multiple potential metrics to enable the decision component to *directly* identify problematic phonemizations that require an oracle's help were introduced. As an additional metric, the combination of those metrics in the form of an *error prediction* system will be implemented.

This error predictor is trained on a part of the test data and, for each sample in this test subset, receives *all other confidence metrics* as input an the number of errors made in the phonemization of that sample by the G2P component as an output. This should make it possible to integrate the confidence metrics into one unified measurement of expected inaccuracy, further improving the decision component's performance.

The approach chosen for this error predictor is *multiple linear regression*, a common approach used in machine learning applications where an output variable's dependency on one or multiple input variables is learned by a model. While more complex models - such as the feed-forward neural networks introduced in previous chapters - would be a valid choice for the error predictor's approach, linear regression has the distinctive advantage of learning *linear dependencies* between the input variables and the output, making it possible to use their learned *weight vectors* to identify the input variables that the predictor assumes to have the highest influence on the G2P system's output error.

4 Test Setup

In this section, the test setup, optimizations and data sets used to evaluate the different request approaches' performance are presented.

4.1 Test System Architecture

The setup to measure the performance of the different approaches can be seen in Figure 4.1. Samples from the test set (i.e. words in their graphemic representation) are fed into the respective *G2P component*. The G2P component's phonemization hypothesis for the given word is then supplied to a *decision component*, together with the relevant phonemization metadata (such as the component's confidence in the hypothesis or the word's grapheme form or phoneme form embedding vector). Based on this metadata, the decision component then either accepts the G2P component's phonemization and adds it to the system's phonemization list for the test set; or it initiates a 'manual' request.

In a real-life application, this manual request would then be forwarded to a human user, such as a linguist expert, for manual labeling in the form of a manual phonemization. In the test setup, this *oracle* is virtualized: when receiving a phonemization request, it performs an immediate lookup for the word's correct phonemization in the test set labels. This access to - in the context of the test set - guaranteedly correct phonemization means that in all our results, we make the assumption that a *perfect oracle* is available. This is a strong and often unrealistic assumption for human oracles (Donmez and Carbonell, 2008). However, investigating approaches to compensate for oracle-induced labeling errors is a field of research on its own and will not be discussed in this thesis.

After the oracle returns the requested labeling for a given word, this phonemization is added to the system's phonemization list for the test set. Subsequently, performance metrics, such as the PER, are computed for the respective approach and the selected hyperparameters (such as hypothesis confidence threshold[1]).

All code used in this thesis was written in the Python[2] programming language. Different virtualenv[3] instances were maintained to cleanly separate the program environments of the different G2P systems. All code, including the LaTeXfiles and graphics used for this document,

[1]The *threshold* is the value of the relevant decision metric (such as confidence) that determines for which input words labeling requests are being made. E.g., any phonemizations for which the system's confidence lies under 0.42 are being sent to the oracle for clarification.

[2]https://www.python.org/

[3]https://virtualenv.pypa.io/

were version controlled in a git[4] repository hosted on GitHub[5].

The models used were trained on the author's home computer, using a standard consumer-grade setup in regards to working memory, CPUs and a dedicated graphics card.

4.2 Decision Parameter Virtualization

During early tests, it quickly became apparent that individual test runs for each set of decision parameters (such as the confidence value) is infeasible for any relevant granularity - as well as unnecessary: both the Sequitur G2P and the LSTM-based G2P systems are fully deterministic in their phonemization output at application time. Thus, the selection of decision parameter thresholds was further virtualized. The fully-trained models for Sequitur G2P and LSTM-based G2P were both run only *once* against the test dataset, without any interference by a request decision component. During these executions, any potentially decision-relevant metadata was kept and stored in a .JSON file, containing the grapheme form, the hypothesized phonemization and a rich amount of metadata for each sample in the test dataset.

This facilitates the evaluation of different decision-making approaches for different decision thresholds without the highly expensive execution of the actual G2P components; for example, the graphs shown in 5.3 were generated by sorting the test samples by the G2P component's confidence in the respective phonemizations, requesting correct labeling for the n samples with the lowest confidence, and calculating the resulting PER.

4.3 Dataset Sources & Preprocessing

One main data source was utilized in the training and evaluation of the present Sequitur- and LSTM-based systems:

CMUdict version 0.7b[6], the most recent version of the Carnegie Mellon University Pronouncing Dictionary. CMUdict 0.7b contains slightly over 134.000 English words and their respective phonemizations in a slightly modified version of the ARPABET. All entries are manually labeled and thus, the CMUdict is a common data set to be used as a gold standard in G2P tasks.

The CMUdict dataset split being used will be that of the sequence-to-sequence tutorial of the Windows CNTK framework[7]. This is in accordance with the data set split utilized by Milde, Schmidt, and Köhler (2017), whose results will be used as a sanity check for the basic performance of the two underlying G2P systems used in this thesis. This split yields approximately 115.000 training samples and almost 13.000 test samples. Because of differences in the choice of development sets and the dealing with duplicate input words, the Sequitur-based system's effective test set is slightly different from the one used in (Milde, Schmidt, and Köhler, 2017), lying closer to 12.000 samples.

[4]https://git-scm.com/
[5]https://github.com/
[6]http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[7]https://github.com/Microsoft/CNTK/tree/master/Examples/SequenceToSequence/CMUDict/Data

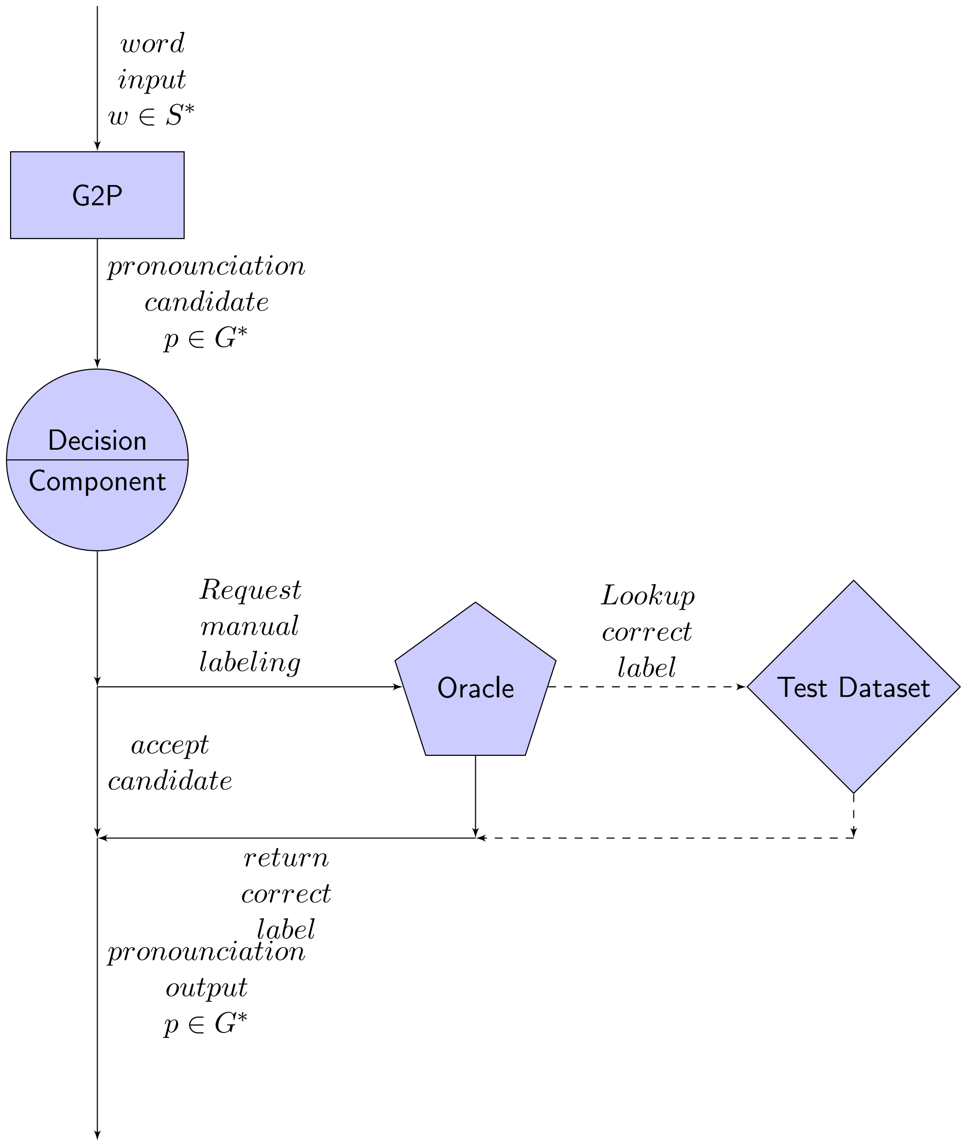


Figure 4.1: The test setup.

For the error predictor based approaches, the test set was further split into two equal parts, resulting in approximately 6.000 samples each in the predictor-train and -test sets.

To evaluate the influence of words of foreign origin on the phonemization accuracy as proposed in Chapter 3.4, the English Wiktionary[8] was crawled for English words tagged as "true French", i.e. words imported from French without any adaptation. This set was intersected with the test dataset from the CNTK-CMUdict-split, producing a list of 1334 words with their known correct pronunciation from the CMUdict dataset.

[8]https://en.wiktionary.org

5 Evaluation and Results

5.1 Approach Performance Metrics

5.1.1 WER and PER

In G2P performance evaluation, two performance metrics are most commonly used: the *Word Error Rate* (WER) and the *Phoneme Error Rate* (PER). (Bisani and Ney, 2008)

The WER is defined simply as the ratio of words with at least one phonemization error over the total number of words in the test set.

The PER is the sum of the *Levenshtein Distances* of all word phonemizations to their correct counterparts over the total number of phonemes in the test set:

$$PER = \frac{\sum_{w_i \in N} (S_i + D_i + I_i)}{|N|}, \tag{5.1}$$

where

- N is the test dataset,
- w_i is the ith word in the dataset,
- S_i, D_i and I_i are the number of substitutions, deletions and insertions respectively that are needed to transform the system's predicted phonemization into the correct one

5.1.2 Request Performance Index

As, to the author's knowledge, the systematic evaluation of request decision approaches as described in this thesis has never been attempted before, no previous unified metric for the comparison of such approaches exists. A new metric of this kind must therefore be defined; if possible, independent of parameters specific to both the selected G2P approaches as well as the G2P domain as a whole.

To achieve this, a novel metric - called the *Request Performance Index* (RPI) - is proposed in this subchapter and used in the subsequent comparison of the different request decision approaches' performance.

In Chapter 3.1, the usage of a *random-based requesting approach* as a performance baseline was proposed. It was demonstrated that such an approach can be expected to result in a continuous *linear* reduction of errors with rising number of manual labeling requests made.

In this thesis, the problem of finding an optimal request decision approach will be considered as an *optimization problem* of the *area under the function* γ that plots the *remaining* *errors* against the *number of requests* made for the current request threshold ρ:

$$\gamma(|R^{\rho}|, \alpha) = |N_{errors}| - |R^{\rho}_{errors}|, \tag{5.2}$$

where

- ρ is the current requesting threshold,
- $R^{\rho} \in N$ is the subset of samples from the test set N that receives manual labeling requests under the current ρ for the approach α and
- N_{errors} and R_{errors} are the number of phonemization errors in the data set and the requested set, respectively.

The area under this curve for an approach α is thus:

$$\int \gamma(\alpha) = \sum_{\rho=\rho_0}^{\rho_n} \gamma(|R^{\rho}|), \tag{5.3}$$

where

- ρ_0 is the request threshold at which no manual requests are performed for any samples in the test set for the approach α, and
- ρ_n is the request threshold at which a manual request is performed for every sample in the test set for the approach α.

This area is normalized by the area of the baseline approach, *random-based requesting*. Since this approaches curve is linear, as discussed in Chapter 3.1, this area is a triangular area:

$$\int \gamma(random) = \frac{1}{2} \cdot |N_{errors}| \cdot |N| \tag{5.4}$$

As a result, the Request Performance Index of an approach α is defined as:

$$\begin{aligned} RPI(\alpha) &= \frac{\int \gamma(\alpha)}{\int \gamma(random)} \\ &= \frac{2 \cdot \int \gamma(\alpha)}{|N_{errors}| \cdot |N|} \end{aligned} \tag{5.5}$$

This normalization means that randomly requesting manual labeling should result in a RPI *close to* 1; request decision approaches that perform *worse* than random requesting should result in a RPI *larger* than 1; and request decision approaches that perform *better* than random requesting should result in a RPI of *less* than 1.

N-Gram	PER	WER
1	42.55%	97.37%
2	17.77%	64.84%
3	10.26%	41.50%
4	7.22%	30.02%
5	6.48%	27.10%
6	6.23%	26.28%
7	6.20%	26.15%
8	6.20%	26.14%

Table 5.1: The baseline performance for different n-gram Sequitur models.

5.2 Basic Model Performance

In this chapter, the basic performance statistics of the two chosen G2P approaches are presented. For the request decision approaches discussed in this thesis to be applicable to real-world G2P applications, the performance of the underlying G2P systems must be sufficiently close to that of current state-of-the-art G2P systems. It will be discussed whether this is the case for both the Sequitur- and the LSTM-based systems utilized as core systems in this thesis.

As reference performance statistics, the results of the previously mentioned, very recent paper on Sequitur- and LSTM-based G2P systems by Milde, Schmidt, and Köhler (2017) will be used.

5.2.1 Sequitur

For the joint sequence models, the Python Sequitur implementation[1] was used. Significant additions were made to the code to allow the output of decision-relevant metadata.

The error rates of the Sequitur-based G2P system trained in the context of this paper are listed in Table 5.1.

These results largely match those produced by Milde et al. (with the reference values being 6.12% PER and 25.71%WER for an 8-gram model). Thus, it can be concluded that the Sequitur-based model used as a basis for the request decision approaches presented in this paper is representative of current state-of-the-art Sequitur-based G2P models.

5.2.2 LSTM

For the LSTM-based model, a Python program based on the Tensorflow machine translation tutorial[2] was implemented.

[1]https://github.com/sequitur-g2p
[2]https://github.com/tensorflow/nmt

Adopting many of the hyperparameters proposed by Milde, Schmidt, and Köhler (2017), an LSTM with the following hyperparameters was trained:

- 3 layers of256 neurons each
- an embedding size of 10
- a dropout probability of 0.5 for the neurons of each layer
- attention model: *scaled Luong-style* attention[3]
- the *Adam* optimization algorithm

The resulting network achieved an PER of $\mathbf{15.73\%}$ on the test data from the CNTK-CMUdict-split. This is a significant decrease of accuracy over the system PER of 6.81% reported in (Milde, Schmidt, and Köhler, 2017); however, this is explainable by the fact that several improvements made by Milde et al. were not implemented in our system, for reasons of implementation- as well as training-effort. These improvements include the reversal of input sequences, the usage of a bi-directional encoder network and the inclusion of residual learning.

In any case, the PER achieved by our system is still that of a moderately well-working G2P system, and this decrease in accuracy should not diminish the applicability of the insights generated in this thesis to other, more fine-tuned LSTM-based G2P systems.

5.3 Random-based Approaches

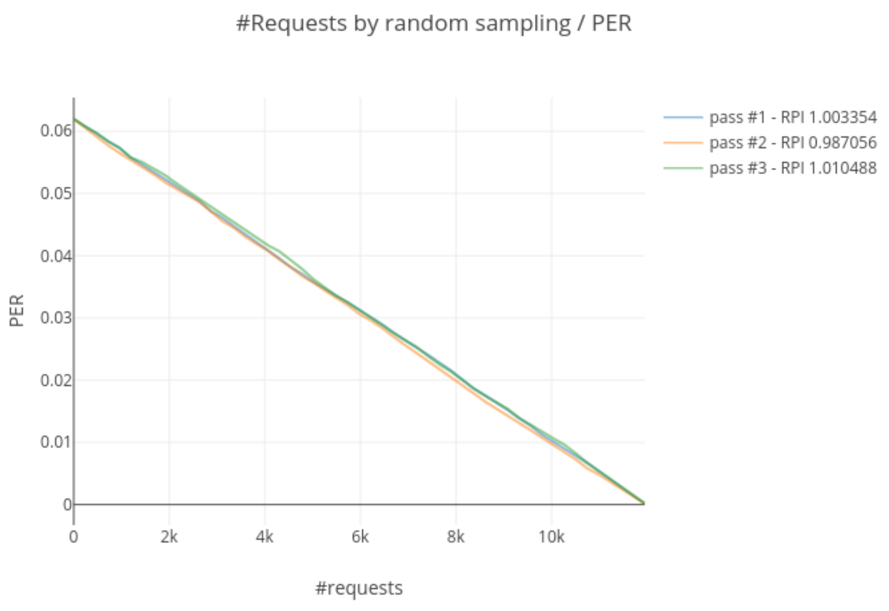
In the previous subchapter, the request performance index was introduced and the hypothesis made that the RPI of a decision component that randomly selects samples for labeling should have an RPI close to 1. In this subchapter, this hypothesis is evaluated. For both Sequitur- and LSTM-based systems, the RPI of random requesting was determined by having multiple simulation runs, in each of which the system ordered the test samples randomly and requested manual labeling for the top n, with an increasing n simulating an increasing probability for any one sample to be sent to the oracle.

The resulting graphs can be seen in Figure 5.1. As is clearly visible, the overall test set error is reduced linearly with increasing number of requested samples, as assumed in Chapter 3.1. Additionally, the resulting RPIs are close to 1, confirming the choice of basing the calculation of the RPI on random requesting as a baseline.

5.4 Theoretical Optimum

After establishing random requesting as the approach with the worst (realistic) RPI, the best-case performance, i.e. that of a theoretical, perfect error predictor is evaluated. This perfect error predictor is simulated by ordering the test data samples by the actual errors made on

[3]As implemented in https://github.com/tensorflow/nmt/blob/master/nmt/attention_model.py



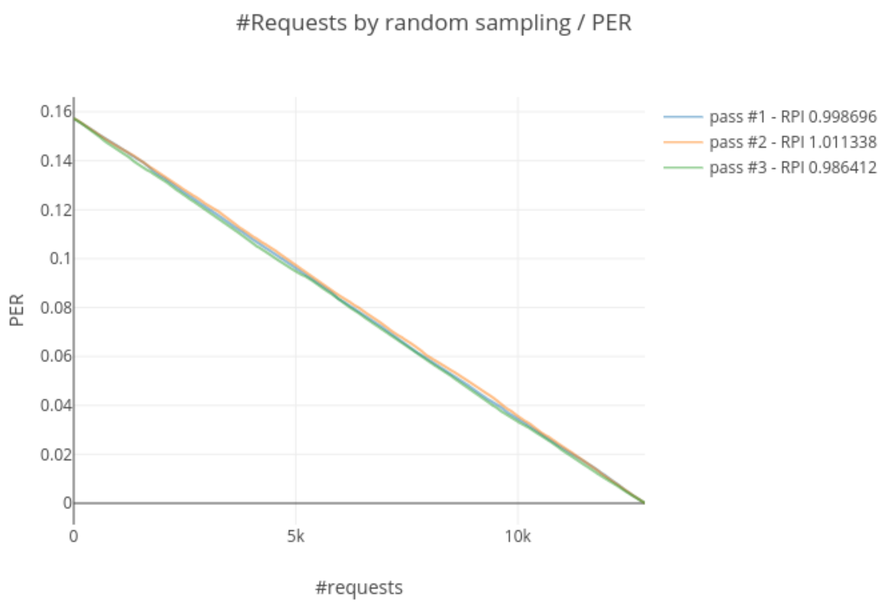


Figure 5.1: The performance of random-requesting for the Sequitur-based system (top) and the LSTM-based system (bottom).

them by the respective G2P system and requesting the n samples with the highest number of errors. The results can be seen in Figure 5.2, resulting in a theoretically achievable RPI of 0.199 for the Sequitur-based and one of 0.402 for the LSTM-based G2P system.

5.5 Confidence-Based Approaches

Figure 5.3 shows the results of confidence-based requesting for the two G2P systems. The RPIs of 0.48 for the joint sequence model and 0.66 for the LSTM based model show that confidence-based requesting of samples results in a massively increased accuracy gain per sample request over random requesting.

This effect is more pronounced for the Sequitur-based system. A possible explanation for this is that this system's output probability is, in fact, a *true* probability measure, based directly on the graphone sequence evidence from the training set, while for the LSTM-based system, no such direct link exists.

Surprisingly, there is only very little improvement gained from the more complex confidence metrics, *confidence margin* and *confidence entropy*: while the usage of the confidence entropy only marginally improves the approach's RPI, switching to the confidence margin actually *decreases* the system's performance. This observation, in concert with the fact that the confidence entropy's calculation is significantly more expensive than that of the default confidence, makes the "natural" confidence the superior vanilla choice for most applications.

An interesting observation regarding the performance of confidence-based requesting for *differently well-trained models* can be found in Figure 5.4. It shows that more well-trained (i.e. higher n-gram) Sequitur models offer an increasingly useful "natural" confidence: while the confidence of the undertrained 1-gram model yields only moderate improvements over random requesting, this gain improves with increasing n-gram-count, until the RPI converges, similar to the PER of the respective systems.

This indicates that while the "natural" confidence metric is less useful for undertrained models, its usefulness improves together with a system's overall accuracy for a given task.

5.6 Ignorance-Based Approaches

While it was possible to jointly discuss the performance on both Sequitur- and LSTM-based systems for the other request decision approaches, the concrete implementations of the *ignorance-based* requesting differ too much between the two models to discuss them in parallel. Therefore, this subchapter is divided into two sections, detailing the implementation and the results of ignorance-based requesting for the two systems, respectively.

5.6.1 Sequitur

The results of ignorance-based requesting are shown in Figure 5.5.

While the resulting RPI of $\mathbf{0.65}$ is inferior to that of confidence-based requesting with 0.48, the results are still somewhat interesting, as they clearly demonstrate that the total likelihood

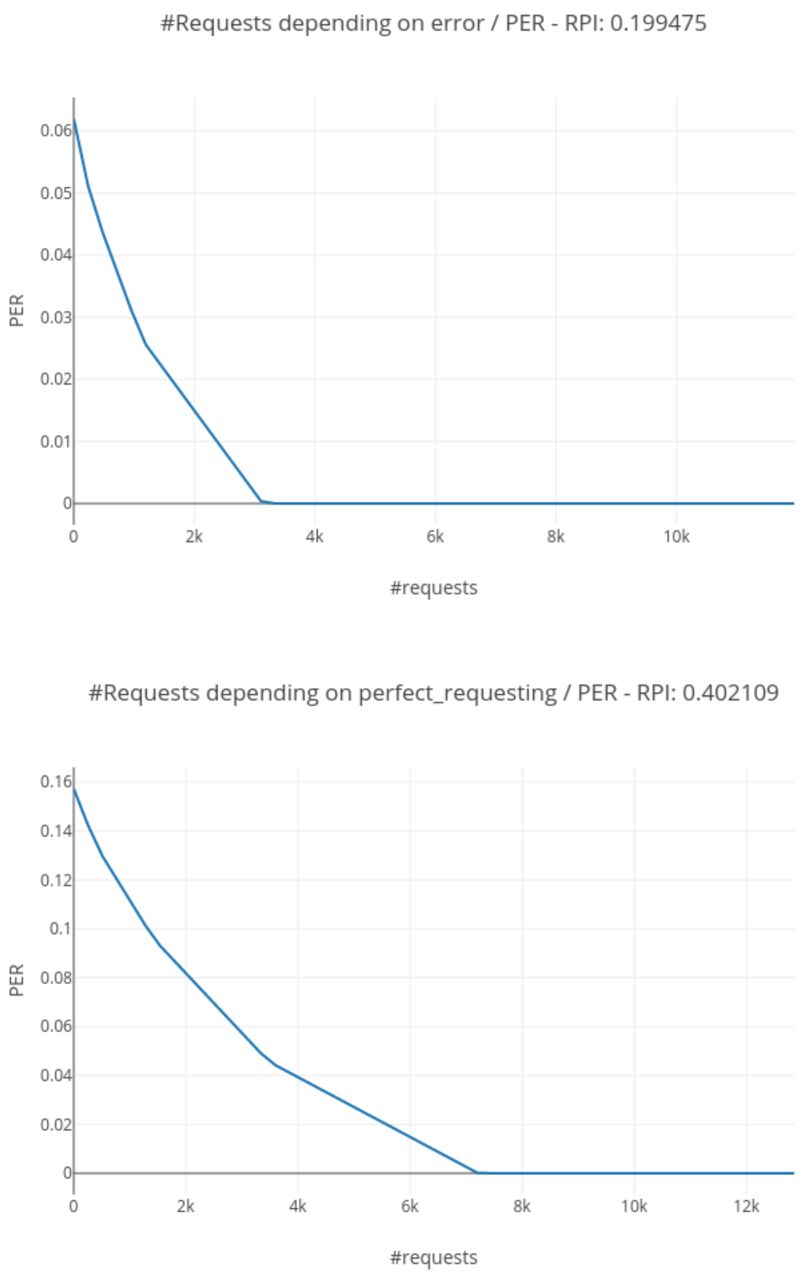
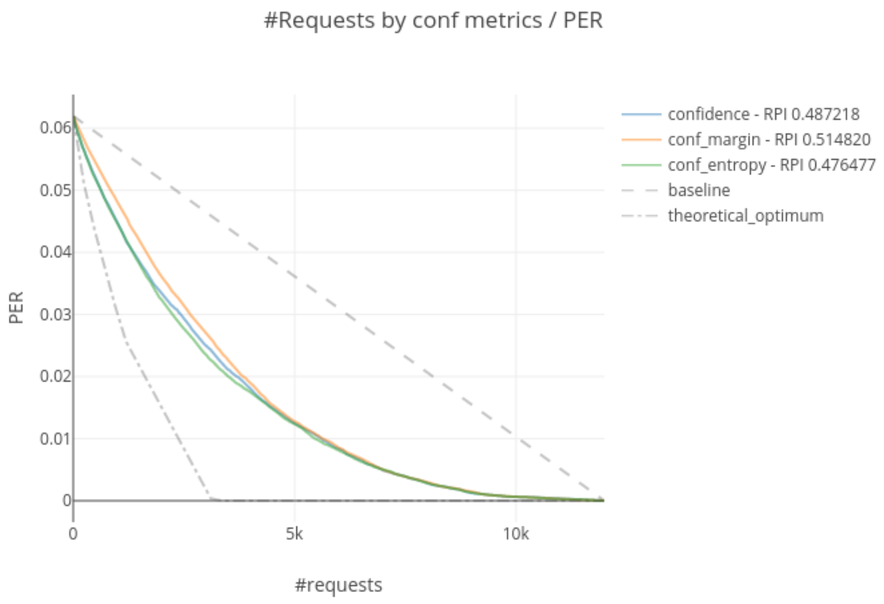


Figure 5.2: The performance of a theoretical, perfect error predictor for the Sequitur-based system (top) and the LSTM-based system (bottom).



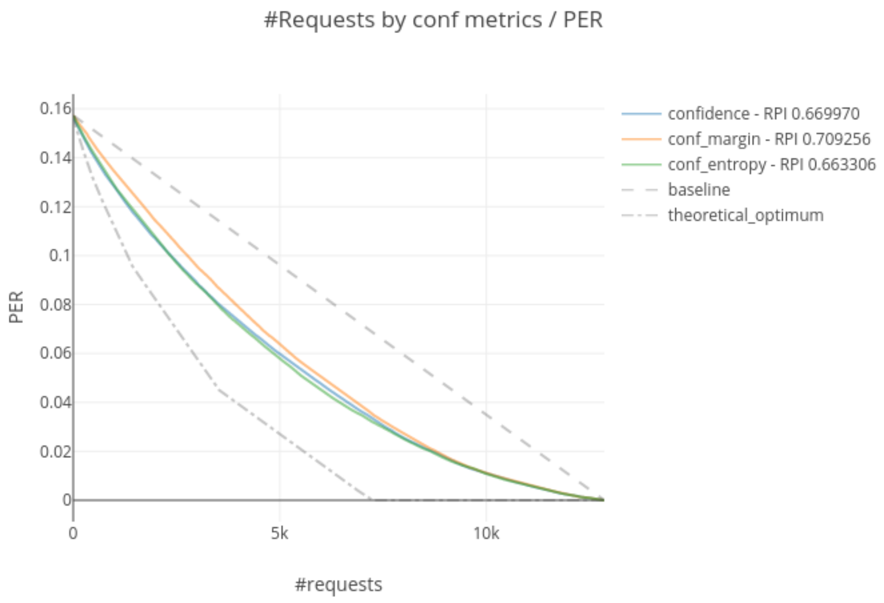


Figure 5.3: The performance of confidence-based requesting for the Sequitur-based system (top) and the LSTM-based system (bottom).

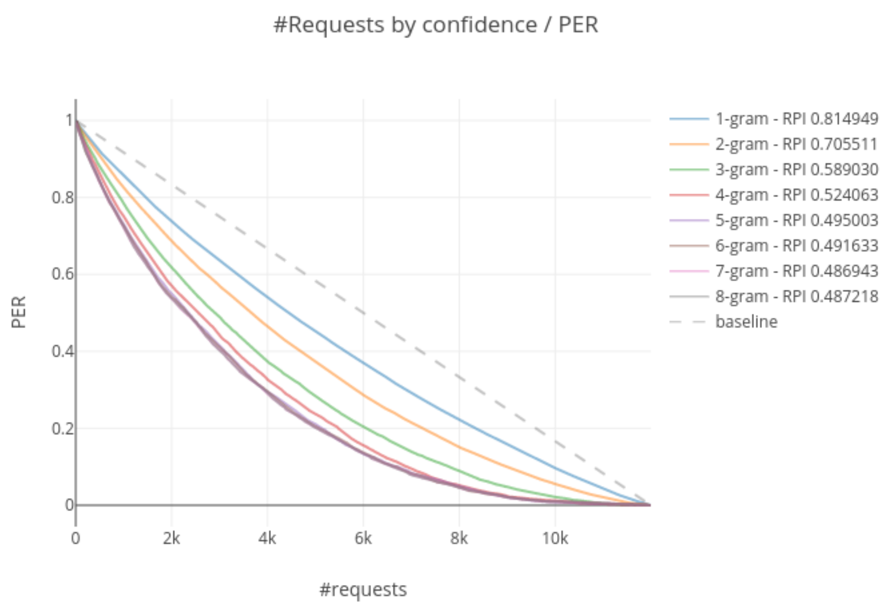


Figure 5.4: The performance of confidence-based requesting for different n-gram Sequitur models, normalized by their PER at 0 requests.

of an input word g based on the training set is a valid metric of the system's ignorance towards that input word; and furthermore, that this ignorance is a valid parameter to predict the necessity for the sytem to request help in its phonemization.

5.6.2 LSTM

The results of various ignorance-based request approaches for the LSTM G2P system can be seen in Table 5.2.

These results somewhat clearly demonstrate that metrics based on the average of the activations of the embedding layer neurons or the hidden layer neurons are not a valid measurement

Approach	RPI
Average embedding distance	1.05
First layer encoder cosine distance	1.02
Sum of encoder first layer cosine distances	1.10
Sum of encoder second layer cosine distances	1.09
Sum of encoder third layer cosine distances	1.11

Table 5.2: The RPIs of several unsuccessful ignorance-metrics for the LSTM system.

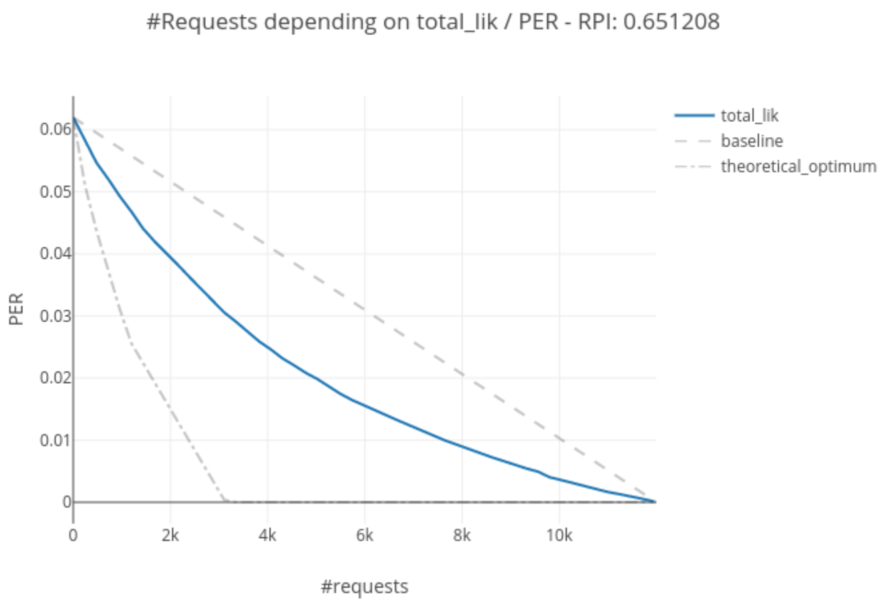


Figure 5.5: The performance of ignorance-based requesting for the Sequitur-based system.

of the system's ignorance towards an input sample.

The explanation for these results is quite straight-forward: metrics such as the average (or the *harmonic mean*, which was implemented but did not yield improved results) are lossy operations in regard to the *sequence* of input symbols. Following this approach, the two input words *"blue"* and its French counterpart *"bleu"* would have the same average vectors at least for the embedding layers, despite posing a very different challenge for a G2P component.

The effects of this are easily demonstrated by the words calculated by this approach as the most "regular" (i.e. words with the smallest average embedding distance) and their least "regular" counterparts. Figure 5.6 shows a selection from the 30 most and least "regular" words, respectively. It appears that the calculated "regularity" of words is dependent mainly on two factors: their individual *letters*, without any regards to their sequence, with letters more commonly used in the English language being a benefit; and, significantly, their *length*. The likely explanation for the preference of longer words is that the averaged embedding vector of a *long word with mostly typical letters* is closest to that of a massive corpus such as the training set.[4]

Following these observations, a further approach was implemented. Using the same test data split as for the error predictor models, a small multi-layer LSTM was trained using the embedding vector sequences of approximately 6.000 word inputs and the respective errors

[4]In fact, in an earlier iteration of the LSTM system, with a different dataset split, one of the words considered to be most "regular" by the approach turned out to be the infamous *Supercalifragilisticexpialidocious* from the musical Mary Poppins.

Most regular words	Least regular words
carolingian	spey
travaglini	sep
materializes	pesch
facilitator's	jest
radicalized	setups
quadrupling	essa
laryngitis	esso
artiodactyls	smother
artiodactyls	expects
petralia	suspects

Figure 5.6: Most and least "regular" words from the test set, according to their average embedding cosine distance to the training set's average embedding vector.

made in their phonemization by the LSTM-based G2P system; subsequently, its accuracy in predicting the G2P system's error *based on the make-up of the input word alone* was evaluated on the remaining 6.000 test samples. The results can be seen in Table 5.3.

The best of these systems achieves an RPI of $\mathbf{0.68}$, only marginally worse than that of confidence-based requesting for the LSTM-based system at an RPI of 0.66. While this is not a true ignorance-based approach, as the smaller LSTMs were trained without any access to the training data of the G2P system, they are able to (reasonably well) identify problematic phonemizations on the morphology of the input word alone, without any further information from the system. This makes them at least related to ignorance-based requesting approaches.

The fairly good performance of both the ignorance-based approach for the Sequitur-based system (RPI: 0.65) and the semi-ignorance-based requesting approach for the LSTM-based system (RPI: 0.68) show that even for non-numeric input data, a reasonably effective metric of ignorance towards an input sample can be found.

5.7 Word-Origin-Based Approaches

As a preliminary test of the influence that words of foreign origin have on a G2P system's phonemization accuracy, the Sequitur-based system was evaluated on the 1344 words of French origin crawled from the English Wiktionary. The phoneme error rate achieved on this test subset was $\mathbf{16.07\%}$, almost three times as high as the base error on the full test dataset. This confirms that the observation made by Milde, Schmidt, and Köhler (2017) that words of foreign origin are especially problematic for a G2P component hold true for English as well as German.

Due to the fact that these words make up less than ten percent of the test samples, however, no further effort was put into this approach. Even if a component capable of perfectly distinguishing these foreign word from non-foreign ones, the impact on the overall

Layers	Neurons	RPI
1	8	0.734
1	16	0.694
1	32	0.688
1	64	0.697
1	128	0.723
1	256	0.749
2	8	0.692
2	**16**	**0.674**
2	32	0.683
2	64	0.705
2	128	0.724
2	256	0.741
3	8	0.709
3	16	0.684
3	32	0.679
3	64	0.694
3	128	0.727
3	256	0.718

Table 5.3: The performance of a symbol-embedding-LSTM based ignorance-request approach for different LSTM layouts.

performance of a request decision component would likely be neglectable.

It is noteworthy, however, that such a component *is* likely to have an impact in G2P domains where a significant number of foreign words can be expected, such as French loan words in the context of fashion, or Latin terms in the context of medicine.

5.8 Error-Predictor-Based Approaches

The performance of the error predictor models can be found in Figure 5.7. In addition to the linear predictor described in section 5.8, a simple predictor-FFNN with one hidden layer of 128 units was trained to learn any potential non-linear relationships between the input variables and the produced error. The error predictors' performance for both systems was effectively identical to that of the confidence-based approaches. On the joint sequence based model, the error predictor scored an RPI of 0.478 (compared to the confidence-based RPI of 0.487), and on the LSTM based model it scored an RPI of 0.66 (compared to the confidence-based RPI of 0.651).

This indicates that a system's confidence in a phonemization is the main (and solely required) metric for manual request decision, at least out of the ones discussed in this thesis.

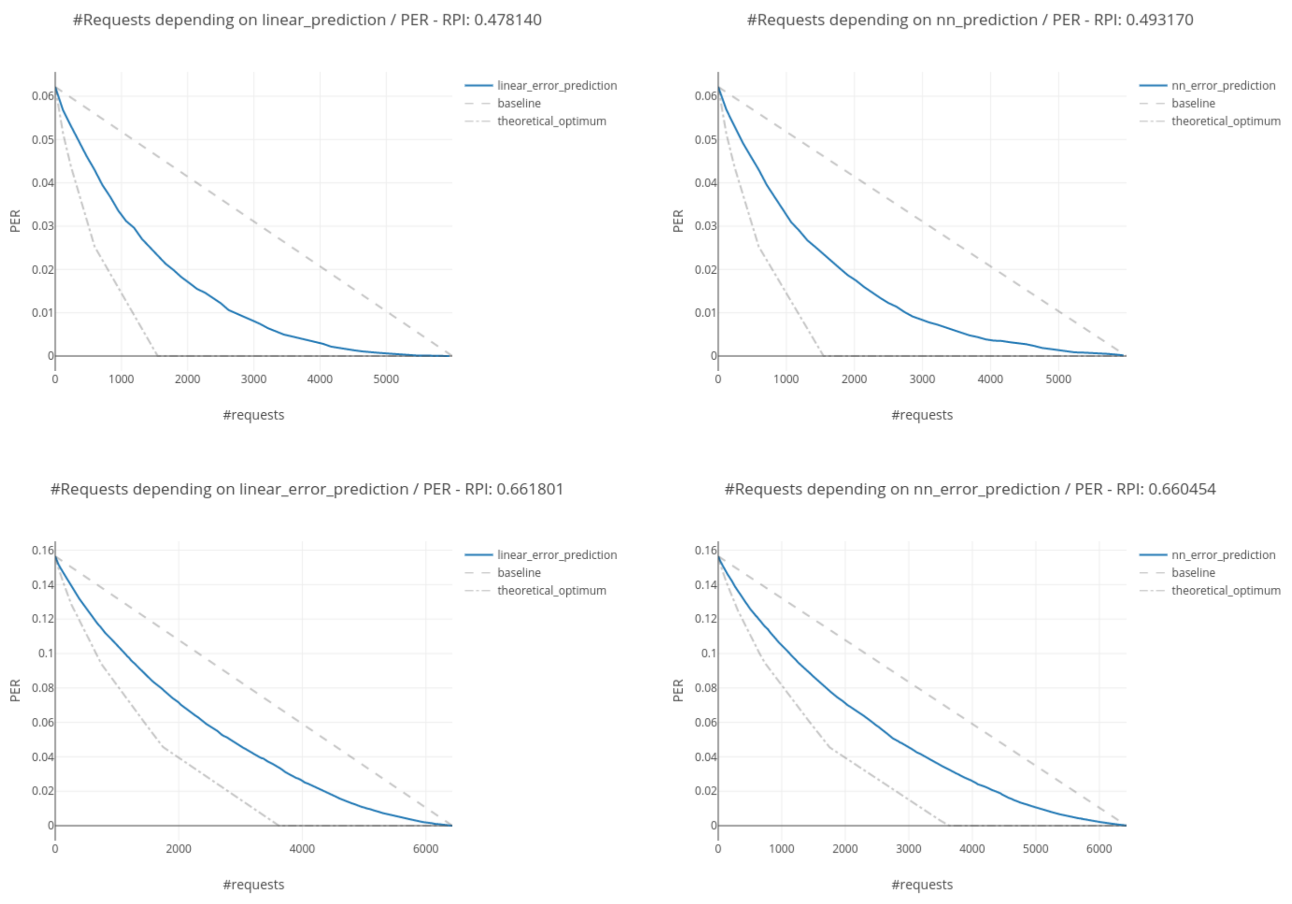


Figure 5.7: The performance of different error predictor models for the Sequitur-based system (top) and the LSTM-based system (bottom).

6 Conclusion and Future Work

In this thesis, a novel application for active learning in G2P systems was introduced: enabling a system to ask for help on particularly problematic phonemizations. This was achieved by defining metrics that enable the system to identify potentially problematic in- or outputs.

It could be shown that the *natural confidence* of both the Sequitur- and the LSTM-based systems is a considerably robust metric to support this decision that, additionally, is easily implemented at very little cost.

Further metrics were introduced with mixed results; some of the metrics predictably failed to prove useful, while others, such as the *ignorance* metrics, likely require further refinement. With the implementation of the ignorance metric, especially that for the Sequitur system, it could be shown that finding such a metric for non-numeric inputs is not only possible, but potentially useful to predict that system's error in the phonemization of an input.

The Request Performance Index, a normalized performance metric to compare application-time requesting approaches, was introduced and demonstrated to accurately express an approach's performance.

The list of evaluated decision approaches is unlikely to be anything approaching exhaustive. The *request by word origin* approach, in particular, demands further research, and with more work, more problematic word categories could certainly be identified.

Another potential future research topic of interest is the consideration of the *individual labeling cost* of an input sample. While in this thesis, each word was considered to be of equal labeling difficulty for a human oracle, the reality is certainly different. This effect is likely even stronger for machine learning domains where more complicated operations are required for the correct manual labeling of a data sample.

Lastly, the exploration of *domain-specific* decision metrics analogue to the *word origin* and *word morphology* for the G2P domain is likely to yield not only exiting results for application-time help requesting, but also for general insights into the respective domain.

The overall combination that achieved the best results was the 8-gram joint sequence model using the system's natural confidence to request labeling. Given the ease of employing the Sequitur implementation for joint-sequence models, accessing the CMUdict pronunciation dataset or pre-trained Sequitur models, and extracting the system's confidence in an output, this system architecture offers an excellent, ready-to-use G2P system capable of dealing with most practical G2P applications, while additionally being capable of requesting expert help on especially problematic inputs.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *International Conference on Learning Representations 2015*. (San Diego, California).

Baranes, Adrien and Pierre-Yves Oudeyer (2013). "Active learning of inverse models with intrinsically motivated goal exploration in robots". In: *Robotics and Autonomous Systems* 61.1, pp. 49–73.

Bisani, Maximilian and Hermann Ney (2008). "Joint-sequence models for grapheme-to-phoneme conversion". In: *Speech communication* 50.5, pp. 434–451.

Cho, Kyunghyun et al. (2014a). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *Conference on Empirical Methods in Natural Language Processing 2014*. (Doha, Qatar). Association for Computational Linguistics.

Cho, Kyunghyun et al. (2014b). "On the properties of neural machine translation: Encoder-decoder approaches". In: *Conference on Empirical Methods in Natural Language Processing 2014*. (Doha, Qatar). Association for Computational Linguistics.

Collis, Jaron (2017). *Glossary of Deep Learning: Word Embedding*. December 9, 2017. URL: `https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca`.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Das, Amitava and Björn Gambäck (2014). "Identifying languages at the word level in code-mixed indian social media text". In: *Proceedings of the 11th International Conference on Natural Language Processing*. (Goa, India). Association for Computational Linguistics, pp. 169–178.

Deligne, Sabine and Frederic Bimbot (1995). "Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams". In: *International Conference on Acoustics, Speech, and Signal Processing, 1995. (ICASSP-95)*. (Detroit, Michigan). Vol. 1. IEEE, pp. 169–172.

Donmez, Pinar and Jaime G Carbonell (2008). "Proactive learning: cost-sensitive active learning with multiple imperfect oracles". In: *17th ACM conference on Information and knowledge management*. (Napa Valley, California). Association for Computing Machinery, pp. 619–628.

Durrani, Nadir, Helmut Schmid, and Alexander Fraser (2011). "A joint sequence translation model with integrated reordering". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. (Portland, Oregon). Association for Computational Linguistics, pp. 1045–1054.

Faruqui, Manaal et al. (2014). "Retrofitting word vectors to semantic lexicons". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association* *for Computational Linguistics*. (Denver, Colorado, USA). Association for Computational Linguistics, pp. 1606–1615.

Gershman, Samuel and Joshua B Tenenbaum (2015). "Phrase similarity in humans and machines". In: *Proceedings of the 37th Annual Conference of the Cognitive Science Society*. (Pasadena, California). Cognitive Science Society.

Hai Son, Le, Alexandre Allauzen, and François Yvon (2012). "Measuring the influence of long range dependencies with neural network language models". In: *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. (Montréal, Canada). Association for Computational Linguistics, pp. 1–10.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Imamura, Kenji, Genichiro Kikui, and Norihito Yasuda (2007). "Japanese dependency parsing using sequential labeling for semi-spoken language". In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, pp. 225–228.

Jacobs, Robert A (1988). "Increased rates of convergence through learning rate adaptation". In: *Neural networks* 1.4, pp. 295–307.

Kapoor, Ashish et al. (2007). "Active learning with gaussian processes for object categorization". In: *11th International Conference on Computer Vision*. (Rio de Janeiro, Brazil). IEEE, pp. 1–8.

Kenter, Tom, Alexey Borisov, and Maarten de Rijke (2016). "Siamese cbow: Optimizing word embeddings for sentence representations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*. (Berlin, Germany). Vol. 1. Association for Computational Linguistics.

Kenter, Tom and Maarten De Rijke (2015). "Short text similarity with word embeddings". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. (Melbourne, Australia). ACM, pp. 1411–1420.

Kim, Young-Bum and Benjamin Snyder (2013). "Optimal Data Set Selection: An Application to Grapheme-to-Phoneme Conversion". In: *2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. (Atlanta, Georgia). Association for Computational Linguistics, pp. 1196–1205.

Koehn, Philipp (2004). "Pharaoh: a beam search decoder for phrase-based statistical machine translation models". In: *Machine translation: From real users to research*, pp. 115–124.

Kominek, John (2009). "TTTS from zero: Building synthetic voices for new languages". PhD thesis. Carnegie Mellon University, Language Technologies Institute, School of Computer Science.

Kominek, John and Alan W Black (2006). "Learning pronunciation dictionaries: language complexity and word selection strategies". In: *Conference of the North American Chapter of the Association of Computational Linguistics*. (New York City, New York). Association for Computational Linguistics, pp. 232–239.

Leidig, Sebastian, Dipl-Inform Tim Schlippe, and Ing Tanja Schultz (2014a). "Single and Combined Features for the Detection of Anglicisms in German and Afrikaans". In: *Bachelor's Thesis*.

Leidig, Sebastian, Tim Schlippe, and Tanja Schultz (2014b). "Automatic detection of anglicisms for the pronunciation dictionary generation: a case study on our German IT corpus". In: *SLTU*, pp. 207–214.

Lughofer, Edwin (2012). "Single-pass active learning with conflict and ignorance". In: *Evolving Systems* 3.4, pp. 251–271.

Lund, Kevin and Curt Burgess (1996). "Producing high-dimensional semantic spaces from lexical co-occurrence". In: *Behavior Research Methods, Instruments, & Computers* 28.2, pp. 203–208.

Luong, Minh-Thang, Hieu Pham, and Christopher D Manning (2015). "Effective approaches to attention-based neural machine translation". In: *2015 Conference on Empirical Methods in Natural Language Processing*. (Lisbon, Portugal). Association for Computational Linguistics, pp. 1412–1421.

Ma, Xiaolei et al. (2015). "Large-scale transportation network congestion evolution prediction using deep learning theory". In: *PLoS One* 10.3, e0119044.

Mandel, Michael I, Graham E Poliner, and Daniel PW Ellis (2006). "Support vector machine active learning for music retrieval". In: *Multimedia systems* 12.1, pp. 3–13.

Marchi, Erik et al. (2017). "Deep Recurrent Neural Network-Based Autoencoders for Acoustic Novelty Detection". In: *Computational intelligence and neuroscience*.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Mikolov, Tomas et al. (2010). "Recurrent neural network based language model". In: *Interspeech 2010*. (Makuhari, Japan). Vol. 2. International Speech Communication Association, pp. 1045–1048.

Mikolov, Tomas et al. (2013). "Efficient estimation of word representations in vector space". In: *International Conference on Learning Representations - Workshop Papers*. (Scottsdale, Arizona).

Milde, Benjamin, Christoph Schmidt, and Joachim Köhler (2017). "Multitask Sequence-to-Sequence Models for Grapheme-to-Phoneme Conversion". In: *Interspeech 2017*. (Stockholm, Sweden). International Speech Communication Association, pp. 2536–2540.

Minsky, Marvin L and Seymour Papert (1969). *Perceptrons: an introduction to computational geometry*. MIT Press.

Mnih, Volodymyr, Nicolas Heess, Alex Graves, et al. (2014). "Recurrent models of visual attention". In: *Conference on Neural Information Processing Systems*. (Montréal, Canada), pp. 2204–2212.

Novak, Josef R et al. (2012). "Improving WFST-based G2P conversion with alignment constraints and RNNLM N-best rescoring". In: *Thirteenth Annual Conference of the International Speech Communication Association*. (Portland, Oregon). International Speech Communication Association.

Rosenblatt, Frank (1958). "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological review* 65.6, p. 386.

Salakhutdinov, Ruslan, Joshua B Tenenbaum, and Antonio Torralba (2013). "Learning with hierarchical-deep models". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1958–1971.

Schlippe, Tim et al. (2012). "Automatic Error Recovery for Pronunciation Dictionaries". In: *Interspeech 2012*. (Portland, Oregon). International Speech Communication Association, pp. 2298–2301.

Settles, Burr (2010). "Active learning literature survey". In: *Computer Sciences Technical Report 1648*. University of Wisconsin, Madison.

Shen, Binbin et al. (2011). "Combining Active and Semi-supervised Learning for Homograph Disambiguation in Mandarin Text-to-Speech Synthesis". In: *Interspeech 2011*. (Florence, Italy). International Speech Communication Association.

Sutton, Charles and Andrew McCallum (2006). *An introduction to conditional random fields for relational learning*. Vol. 2. Introduction to statistical relational learning. MIT Press.

Toshniwal, Shubham and Karen Livescu (2016). "Jointly learning to align and convert graphemes to phonemes with neural attention models". In: *Spoken Language Technology Workshop (SLT) 2016*. (San Diego, California). IEEE, pp. 76–82.

Yadav, Neha, Anupam Yadav, and Manoj Kumar (2015). *An introduction to neural network methods for differential equations*. Springer.

Yang, Jie et al. (2003). "Automatically labeling video data using multi-class active learning". In: *9th International Conference on Computer Vision*. (Catania, Italy). IEEE, pp. 516–523.

Yu, Lei et al. (2014). "Deep learning for answer sentence selection". In: *Conference on Neural Information Processing Systems - Deep Learning Workshop 2014*. (Montréal, Canada).

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 12.12.2017

Mario Mohr

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 12.12.2017

Mario Mohr