MŰEGYETEM 1782

Budapest University of Technology and Economics

Doctoral School of Informatics

Department of Telecommunications and Media Informatics

Novel NLP Methods for Improved Text-To-Speech Synthesis

Sevinj Yolchuyeva, M.Sc.

Thesis booklet

Supervisors

Bálint Gyires-Tóth, Ph.D.

Géza Németh, Ph.D.

Budapest, Hungary

February 2021

1 Introduction

Text-to-Speech (TTS) technology generates synthetic voice using textual information only. Thus, it may serve as a more natural interface in human-machine interaction. TTS is an essential tool in many application areas such as digital personal assistants, dialogue systems, talking solutions for blind and visually impaired people, people who have difficulties in spelling (dyslexics), teaching aids, text reading, talking audiobooks and toys. Over the last years, significant research progress was achieved in this field. Generally, 'today's state-of-the-art TTS is either based on unit selection or statistical parametric methods. Particular attention has been paid to Deep Neural Network (DNN) based TTS lately, due to its advantages in flexibility, robustness and small footprint. Among the essential properties of a speech synthesis system are naturalness and intelligibility. Naturalness expresses to what extent the output approaches human speech, whereas intelligibility is the easiness with which the information content can be understood. Text-to-Speech systems may be divided into two subsystems: natural language processing-based text processing and speech generation. Natural Language Processing (NLP) derives from the combination of linguistic and computer sciences. It mainly contains three steps for TTS systems: text analysis, phonetic analysis and prosodic analysis. Text analysis includes segmentation, text normalization and Part-of-Speech (POS) tagging. Phonetic conversion assigns phonetic transcription to each word. There are several approaches to phonetic conversion. Two main directions are rule and dictionary-based, or data-driven statistical and machine learning approaches. Prosodic analysis performs intonation, amplitude and duration modelling of speech. The NLP subsystem has a great influence on the achievable performance of the whole TTS system. The communicative context of the system is typically determined (domain-specific TTS synthesis) a priori or ignored.

Recently, some works, such as Deep Voice [1] and Tacotron [2] have been done to replace some or all components of traditional TTS systems with deep neural networks in an end-to-end structure. However, most commercial systems are still based on unit selection TTS, where speech is generated by the concatenation of acoustic units selected from a speech corpus. This technique permits high, controllable synthetic voice quality. To create a new voice for a unit selection TTS system, a voice talent reads aloud a dedicated text. Then all recorded utterances are phonetically segmented: in general, phonemic transcriptions are derived from the text with grapheme-to-phoneme conversion (G2P) systems and aligned automatically on the speech signal [3, 4]. G2P conversion is the process of generating pronunciation for words based on their written form [5]. The encoder-decoder structure was studied for the G2P task [6, 7, 8] before, but usually recurrent neural network-based architectures, like LSTM and GRU networks were involved. For example, 'Baidu's end-to-end text-to-speech synthesizer, called Deep Voice, uses the multi-layer bidirectional encoder with 'GRU's non-linearity and an equally deep unidirectional GRU decoder [1].

TTS systems need to work with texts that contain non-standard words, including numbers, dates, currency amounts, abbreviations and acronyms. Due to this reason, text normalization is an essential task for a TTS system to convert written-form texts to spoken-form strings. Rule-based, dictionary-based methods, Finite State Automata (FSA) are applied for different languages [9]. In order to embolden more research in this direction, a challenge with an open-source dataset was published [10]. Generally, two different models are utilized in this domain: a bidirectional sequence-to-sequence and an attention-based RNN sequence-to-sequence model.
Recent advances in deep learning significantly improved the development of Text-to-Speech (TTS) systems through more effective and efficient learning of 'speakers' voice and speaking styles and more natural generated speech. To create an effective dialogue system, which one of the stages of TTS is very hard and Spoken Language Understanding (SLU) is essential. SLU aims to form a semantic frame that captures the semantics of user utterances or queries. Intent detection is one of the main tasks of an SLU system, that focuses on classifying the user's intent and extracting semantic concepts as constraints for natural language. Rule-based approaches, like conditional random field (CRF) and Support Vector Machines, were investigated for intent detection [12, 13]. Furthermore, artificial neural network-based models have also been investigated. Convolutional neural networks (CNN) are applied for classifying intents in [14]. The combination of CNN and the triangular CRF model (TriCRF) is proposed for the intent labels and the slot filling in [15].

2 Research Objectives

The general research topic of my Ph.D. work is novel NLP methods for improved text-to-speech synthesis. I have chosen the topic of novel NLP methods for improved TTS synthesis for my research because of its novelty and numerous challenges. In the current thesis booklet, I summarize the novel outcomes of my research grouped in the three research objectives.
The *first research* objective is creating novel models for grapheme-to-phoneme (G2P) conversion. G2P is essential to develop a phonemic lexicon in TTS and automatic speech recognition (ASR) systems. This part of my research introduces and evaluates novel convolutional neural network (CNN) based and Transformer architecture based G2P approaches. The suggested methods approach the accuracy of previous state-of-the-art results in terms of phoneme error rate.

The *second research* objective proposes novel models for text normalization. I developed CNN-based text normalization, and the training, inference times, accuracy, precision, recall, and F1-score were evaluated on an open-source dataset. The performance of CNNs is evaluated and compared with a variety of different

Long Short-Term Memory (LSTM) and Bi-LSTM architectures with the same dataset.

The *third research* topic introduces novel models for intent detection. I developed novel models, which utilize the combination of Bi-LSTM and Self-attention Network (SAN) for this task. Experiments on different datasets were evaluated.

3 Research Methodology

During my research, the results of the proposed methods were investigated using common, research field-specific methodologies. In this chapter, I introduce the datasets, tools and evaluation methods used in the research. In the next sections, I present the methods with materials per thesis groups:

3.1. Grapheme-to-phoneme (G2P) conversion

In this part of my research (see below Thesis group I), I used the CMU pronunciation[1] and NetTalk datasets, which are frequently used by various researches [5, 6, 8]. The training and testing splits are the same as found in [5, 6, 8], thus, the results are comparable. CMUDict contains a 106,837-word training set and a 12,000-word test set (reference data). 2,670 words are used as a development set; there are 27 graphemes (uppercase alphabet symbols plus the apostrophe) and 41 phonemes. NetTalk contains 14,851 words for training, 4,951 words for testing and does not have a predefined validation set. There are 26 graphemes (lowercase alphabet symbols) and 52 phonemes in this dataset. For evaluation, the following common metrics were utilized:

Phoneme Error Rate (PER) is the Levenshtein distance between the predicted phoneme sequences and the reference phoneme sequences, divided by the number of phonemes in the reference pronunciation [16]. Edit distance (also known as Levenshtein distance) is the minimum number of insertions (I), deletions (D) and substitutions (S), that are required to transform one sequence into the other. In case of multiple pronunciation samples for a word in the reference data, the sample that has the smallest distance to the candidate is used.

Word Error Rate (WER) is the percentage of words in which the predicted phoneme sequence does not exactly match any reference pronunciation, the number of word errors is divided by the total number of unique words in the reference.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed: February 2021

3.2. Text normalization

In this part (Thesis group II), I used an open data set of Kaggle[2], which consists of 9,918,441 (9.9 million) words of English text. The data were derived from Wikipedia regions, and it is split into sentences. The format of the annotated data is the same as in Table 1: the first column is called 'sentence_'id', each sentence has a sentence_id; the second column is called 'token_'id', each token within a sentence has a token_id; the third column is "'class' which shows the class of the given token; the column "'before' and "'after' give sample raw text (or token) and normalized text consequently.

Table 1. The format of the annotated data

sentence_id	**token_id**	**Class**	**Before**	**After**
			<SE>	
1	0	PLAIN	I	I
1	1	PLAIN	wake	wake
1	2	PLAIN	up	up
1	3	PLAIN	at	at
1	4	TIME	**9:00 AM**	**nine a m**
1	5	PUNCT	.	.
			</SE>	

I also used <SE> and </SE> tokens as beginning-of-sentences and end-of-sentence tokens. Moreover, each token is attributed to one of the classes, and generally, there are 16 different classes.

Table 2. Corpus size statistics for the open-source dataset.

Data	**Number of sentences**	**Number of unnormalized tokens**	**Number of normalized tokens**
Training	600040	8075998	9053849
Development	70560	970972	1086032
Test	77468	1063403	1188811

These classes are called as PLAIN, PUNCT, ADDRESS, ELECTRONIC, LETTERS, CARDINAL, VERBATIM, MEASURE, ORDINAL, DECIMAL, MONEY, DIGIT, ELECTRONIC, TELEPHONE, TIME, FRACTION. I split the dataset into 3 parts: training, development, and test sets. The number of sentences,

[2] https://www.kaggle.com/c/text-normalization-challenge-english-language/data, Accessed: February 2021.

number of unnormalized tokens and number of normalized tokens for training, development and test datasets are shown in Table 2.

I used accuracy, precision, recall, and F-measure as an evaluation metric and investigated the confusion matrices.

3.3. Intent Detection

In the third main topic of my research (Thesis group III), I used the Airline Travel Information System (ATIS) dataset, which is frequently chosen by various researchers [15, 17]. The dataset contains audio transcriptions from people making flight reservations. The training set contains 4,478 utterances, the test set contains 893 utterances, and 500 utterances are used for validation. I used Natural Language Understanding[3] benchmark dataset too. This dataset is collected from the Snips personal voice assistant; the number of samples for each intent is approximately the same. The training set contains 13,084 utterances, the validation (development) and test sets contain 700-700 utterances. All words are labelled with a semantic label in a BIO format, which "'B' means to begin, "'I' means inside, 'O' is outside. Words which don't have semantic labels are tagged with 'O'. There are 120 slot labels and 21 intent types in ATIS; there are 72 slot labels and 7 intent types in Snips dataset. Vocabulary size of these datasets is 722 and 11,241 in ATIS and Snips, respectively.

Furthermore, I used Smart Lights and Smart Speaker datasets, which are produced by [18]. Smart Lights has 6 intents allowing to turn light on and off or change its brightness or colour. It has a vocabulary size of approximately 400 words. Smart Speaker dataset has 9 intents and vocabulary size is approximately 1,270. In these two datasets, I split the dataset into 90-10% as training and test sets. The validation dataset consists of 10% of the training set.

I used accuracy, precision, recall as evaluation metrics and also investigated the confusion matrix.

3.4. Software and Hardware Details

The proposed models in these thesis groups were implemented in Keras[4] which is a deep learning framework with TensorFlow[5] backend for Python. I also used NVidia Titan Xp and Titan X GPU cards hosted in two i7 workstations with 32GB RAM for training and inference.

[3]https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines, Accessed: February 2021.
[4] https://keras.io, Accessed: February 2021.
[5] https://www.tensorflow.org, Accessed: February 2021.

4 New Results

I summarize my contributions to three thesis groups. The first group of these deals introduces novel models for G2P conversion. The models are discussed in Thesis I.1, Thesis I.2, Thesis I.3, Thesis I.4, and Thesis I.5. The second group concentrates on novel models for text normalization. Finally, the third thesis group is related to intent detection, as is one of the main tasks of SLU.

4.1. Thesis Group I. – Grapheme-to-phoneme (G2P) conversion

I designed and created a novel CNN-based sequence-to-sequence (seq2seq) architecture for G2P conversion. My approach includes an end-to-end CNN G2P conversion with residual connections, furthermore, a model, which utilizes a convolutional neural network (with and without residual connections) as encoder and Bi-LSTM as a decoder. I compare my approach with state-of-the-art methods, including Encoder-Decoder LSTM and Encoder-Decoder Bi-LSTM. Training and inference times, phoneme and word error rates were evaluated on the public CMUDict dataset for US English, and the best performing convolutional neural network-based architecture was also evaluated on the NetTalk dataset. Furthermore, I implemented the transformer network [19], which architecture is completely based on attention mechanisms for G2P conversion. I made a comparison between the transformer and CNN-based G2P methods.

LSTM-based Encoder-Decoder for G2P conversion

The encoder-decoder structures have shown state-of-the-art results in different NLP tasks.

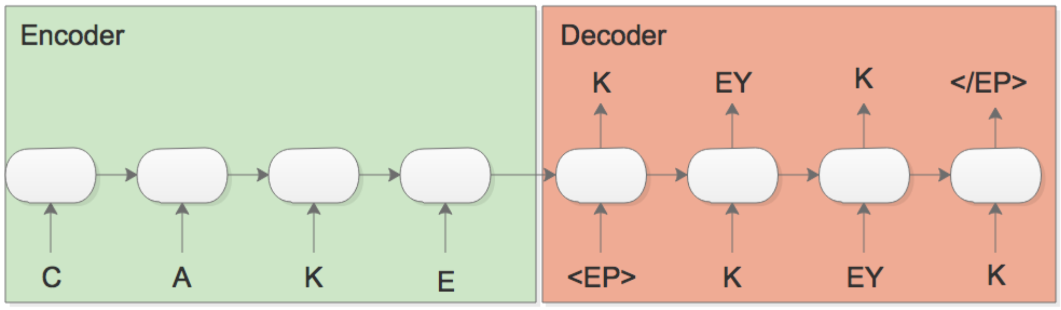


Figure 1. The input of the encoder is ‘CAKE’ grapheme sequence, and the decoder produces ‘K EY K’ as phoneme sequences. The left side is the encoder; the right side is the decoder. The model stops making predictions after generating the end-of-phonemes tag.

The main idea of these approaches has two steps: the first step is mapping the input sequence to a vector; the second step is to generate the output sequence based on the learned vector representation. Encoder-decoder models generate an output after the complete input sequence is processed by the encoder, which enables the decoder to learn from any part of the input without being limited to fixed context windows**.** Figure 1 shows an example of an encoder-decoder architecture.

Thesis I.1. [J1] *I designed and implemented a novel encoder-decoder Bi-LSTM (**BI-LSTM_BI-LSTM**) model for G2P conversion. I experimentally confirmed that encoder-decoder Bi-LSTM with* **attention layer (*BI-LSTM_BI-LSTM+ATT*)** *show significant improvement on both PER and WER compared to the previous deep learning solutions such as encoder-decoder LSTM with and without attention layer (**LSTM_LSTM** and **LSTM_LSTM+ATT**).*

I implemented several models with different hyperparameters by used encoder-decoder architectures. I selected the LSTM based models with the highest accuracy from many experiments, and I called these models **LSTM_LSTM** and **BI-LSTM_BI-LSTM**:

LSTM_LSTM: This model uses LSTMs for both the encoder and the decoder. The LSTM encoder reads the input sequence and creates a fixed-dimensional vector representation. It can be seen that both LSTMs have 1024 units; softmax activation function is used to obtain model predictions.

BI-LSTM_BI-LSTM: In this model, both the encoder and the decoder are Bi-LSTMs. The input is fed to the first Bi-LSTM (encoder), which combines two unidirectional LSTM layers that process the input from left-to-right and right-to-left. The output of the encoder is given as input for the second Bi-LSTM (decoder). Finally, the softmax function is applied to generate the output of phonemes' one-hot vector representations. During the inference, the complete input sequence is processed by the encoder, and after that, the decoder generates the output. For predicting a phoneme, both the left and the right contexts are considered. This model was also inspired by an existing solution [5].

Although the encoder-decoder architecture achieves competitive results on a wide range of problems, it suffers from the constraint that all input sequences are forced to be encoded to a fixed size latent space. To overcome this limitation, I investigated the effects of the attention mechanism proposed by [20] in **LSTM_LSTM** and **BI-LSTM_BI-LSTM**. I applied an attention layer between the encoder and decoder LSTMs in case of **LSTM_LSTM**, and Bi-LSTMs for **BI-LSTM_BI-LSTM** and called as **LSTM_LSTM+ATT**, **BI-LSTM_BI-**

LSTM+ATT respectively. The introduced attention layers are based on global attention [20].

Encoder CNNs, Decoder Bi-LSTM for G2P

Convolutional neural networks are used in various fields, including image, object and handwriting recognition, face verification, natural language processing and machine translation [21-25]. The architecture of an ordinary CNN is composed of many layer types (such as the convolutional layers, pooling layers, fully connecting layers, etc.) where each layer carries out a specific function.

I implemented a CNN as an encoder, and a Bi-LSTM as a decoder and called as **CNN_BI-LSTM**. In this model, the CNN layer takes graphemes as input and performs convolution operations. For regularization, I also introduced batch normalization in this model [23].

End-to-end CNNs with residual connections for G2P

Thesis I.2. [J1] *I created novel **end-to-end CNN (CNN_CNN)** and end-to-end CNN with residual connections (**CNN+RES**) models and showed the advantage of the residual connections for G2P conversion. I empirically confirmed that the **CNN+RES** model outperforms the same fully convolutional model without residual connections.*

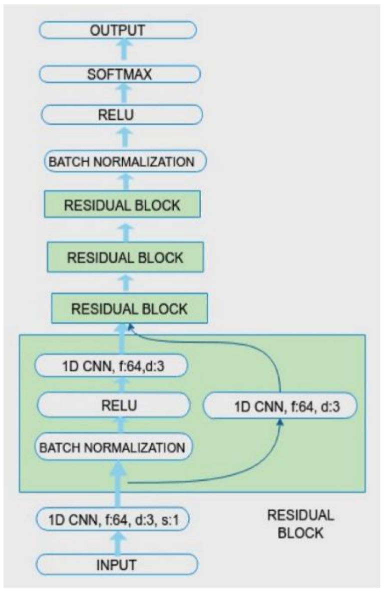
I implemented a neural network, which contains convolutional layers only with residual connections (blocks) [24] and called **CNN+RES**. These residual connections have two rules [24, 25]:

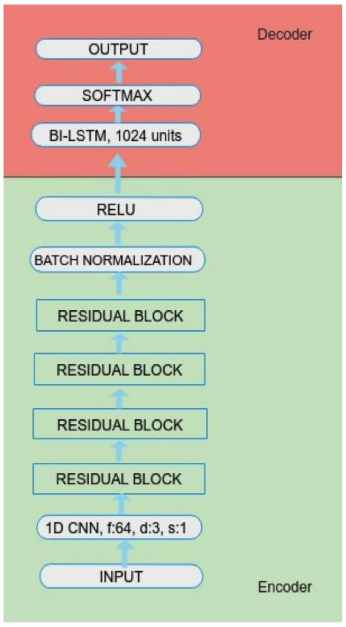
(1) if feature maps have the same size, then the blocks share the same hyperparameters.

(2) each time when the feature map is halved, the number of filters is doubled.

In this model, I applied one convolutional layer with 64 filters to the input layer, followed by a stack of residual blocks. Through hyperparameter optimization, the best result was achieved by 4 residual blocks, as shown in Figure 2(a) and the number of filters in each residual block is 64, 128, 256, 512, respectively. Each residual block contains a sequence of two convolutional layers followed by a batch normalization [23] layer and ReLU activation.

For comparison, I carried out experiments with the same fully convolutional models without residual connections (**CNN_CNN**). The phoneme and word error rates were better with residual connections, as expected.





a) b)

Figure 2. G2P conversion based on (a) convolutional neural network with residual connections (**CNN+RES**) and (b) encoder convolutional neural network with residual connections and decoder Bi-LSTM (**CNN+RES_BI-LSTM**). f, d, s is the number of the filters, length of the filters and stride, respectively.

Encoder CNNs with residual connections and Decoder Bi-LSTM for G2P

Thesis I.3. [J1] *I experimentally showed that training and inference speeds on the currently used architecture are faster in the* ***end-to-end CNNs with residual connections (CNN+RES)*** *than in encoder-decoder Bi-LSTM (BI-LSTM_BI-LSTM) although* ***encoder-decoder Bi-LSTM*** *shows slightly better result in terms of PER and WER. Moreover, I have shown that the best WER and PER values are achieved by the model of encoder CNNs with residual connections, decoder* ***Bi-LSTM (CNN+RES_BI-LSTM)*** *in the various configuration of CNN models: PER is 4.81%, and WER is 25.13% on CMUDict.*

I combined **CNN_BI-LSTM** and **CNN+RES**: the encoder has the same convolutional neural network architecture with residual connections and batch normalization, which was introduced in **CNN+RES**; the decoder is a Bi-LSTM, as in **CNN_BI-LSTM**. It was called **CNN+RES_BI-LSTM**. The structure of this model is presented in Figure 2(b).

Evaluation of Thesis I.1., Thesis I.2., and Thesis I.3.

The evaluation of the proposed models (**LSTM_LSTM, LSTM_LSTM+ATT, BI-LSTM_BI-LSTM, BI-LSTM_BI-LSTM+ATT, CNN_BI-LSTM, CNN+RES, CNN+RES_BI-LSTM)** on the CMUDict dataset is shown in Table 3. The first and second columns show the model number and the applied architecture, respectively. The third and fourth columns show the PER and WER values. The fifth column of Table 3 contains the average sum of training and validation time of one epoch. The last two columns present information about the size of the models (number of trainable parameters), and the number of epochs to reach minimum validation loss. According to the results, the encoder-decoder Bi-LSTM architecture (**BI-LSTM_BI-LSTM**) outperforms the first model, as expected. But attention-based **LSTM_LSTM**, which is called **LSTM_LSTM+ATT** outperforms **BI-LSTM_BI-LSTM** in terms of PER. The best WER and PER values are achieved by the **CNN+RES_BI-LSTM**: PER is 4.81%, and WER is 25.13%. Attention-based **BI-LSTM_BI-LSTM** (called **BI-LSTM_BI-LSTM+ATT** in Table 1) approaches the best result in terms of both PER and WER. But the number of parameters of **BI-LSTM_BI-LSTM+ATT** is twice as much as for **CNN+RES_BI-LSTM.** Although the **CNN+RES** was faster than all other models, both PER and WER of this model are the highest, however, still competitive. Moreover, this model has also the least parameters.

Table 3. Results on the CMUDict dataset.

Model	**Method**	**PER (%)**	**WER (%)**	**Time (s)**	**Numb. of epochs**	**Model size**
LSTM_LSTM	Encoder-Decoder LSTM	5.68	28.44	467.73	185	12.7M
LSTM_LSTM +ATT	Encoder-Decoder LSTM with attention layer	5.23	28.36	688.9	136	13.9M
BI-LSTM_BI-LSTM	Encoder-Decoder Bi-LSTM	5.26	27.07	858.93	177	33.8M
BI-LSTM_BI-LSTM+ATT	Encoder-Decoder Bi-LSTM with attention layer	4.86	25.67	1045.5	114	35.3M
CNN_BI-LSTM	Encoder CNN, decoder Bi-LSTM	5.17	26.82	518.3	115	13.1M
CNN+RES	End-to-end CNN (with res. connections)	5.84	29.74	176.1	142	7.62M
CNN+RES_BI-LSTM	Encoder CNN with res. connections, decoder Bi-LSTM	4.81	25.13	573.5	147	14.5M

Transformer-based G2P Conversion

Transformer networks are based on an encoder-decoder architecture and learn input - output mappings without using recurrent or convolutional neural networks (CNN) [19]. First, transformer networks were used for neural machine translation, and they achieved state-of-the-art performance on various datasets. In [19], it was shown that transformers could be trained significantly faster than recurrent or convolutional architectures for machine translation tasks. These results suggest investigating the possibility of utilizing a transformer network to G2P.

Thesis I.4. [C1] *I designed and implemented three novel transformer-based models for G2P where the first one consists of 3-3 encoder and decoder layers (**Transformer 3x3),** the second contains 4-4 encoder and decoder layers (**Transformer 4x4**), and the third one consists of 5-5 encoder and decoder layers (**Transformer 5x5**).*

Thesis I.5. [C1] *I empirically confirmed, that for* **transformer***-based G2P the Transformer 4x4 model outperforms the convolutional-based approach in terms of* ***WER*** *and my results significantly exceeded previous recurrent approaches (without attention) regarding* ***WER*** *and* ***PER*** *on CMUDict and NetTalk datasets.* ***WER*** *decreased by* ***3.03****% compared to the best alternative on CMUDict.*

For that reason, I created a novel transformer-based model for G2P conversion. The transformer is organized by stacked self-attention and fully connected layers for both the encoder and the decoder [19], as shown in the left and right halves of Figure 3, respectively. Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence to compute its internal representation.

Without using any recurrent layer, positional encoding is added to the input and output embeddings [19]. The positional information provides the transformer network with the order of input and output sequences. Both the encoder and the decoder are composed of a stack of *N* identical blocks, and each block has two layers. The first is the multi-head attention layer, which is several attention layers used in parallel. The second is a fully connected position-wise feedforward layer. These layers are followed by dropout and normalization layers [40]. At the top of the decoder, there is the final fully-connected layer with linear activation, which is followed by softmax output.

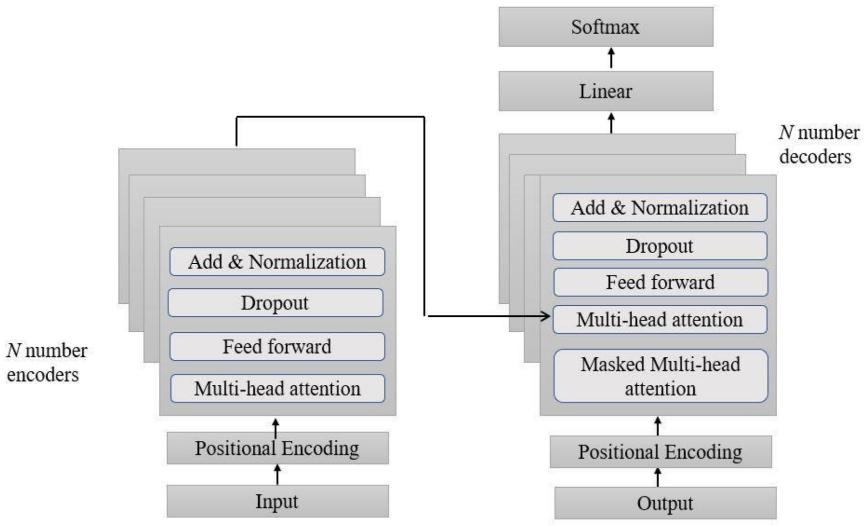


Figure 3. The framework of the proposed model.

I investigated three transformer architectures, with 3 encoder and decoder layers (it is called **Transformer 3x3** in Table 4), 4 encoder and decoder layers (it is called **Transformer 4x4** in Table 4) and 5 encoder and decoder layers (it is called **Transformer 5x5** in Table 4).

Evaluation of Thesis I.4 and Thesis 1.5.

According to the results (Table 4), **Transformer 4x4** (4 layers encoder and 4 layers decoder) outperforms **Transformer 3x3** (3 layers encoder and 3 layers decoder). Contrary to expectations **Transformer 5x5** (5 layers encoder and 5 layers decoder) didn't outperform **Transformer 4x4** (4 layers encoder and 4 layers decoder). Increasing the numbers of encoder-decoder layers leads to much more training parameters.

Table 4. Results on the CMUDict and NetTalk dataset.

Dataset	**Model**	**PER**	**WER**	**Time [s]**	**Model size**
CMUDict	Transformer 3x3	6.56	23.9	76	1.49M
	Transformer 4x4	**5.23**	**22.1**	98	1.95M
	Transformer 5x5	5.97	24.6	126	2.4M
NetTalk	Transformer 3x3	7.01	30.67	33	1.50M
	Transformer 4x4	**6.87**	**29.82**	39	1.96M
	Transformer 5x5	7.72	31.16	48	2.4M

In the G2P task, similar complexity to NMT (neural machine translation) can be rarely permitted. The high number of parameters sometimes does not even result in better performance. In terms of PER, **Transformer 5x5** is better than **Transformer 3x3** on CMUDict but didn't exceed **Transformer 4x4**, **Transformer 3x3** in the

point of WER on both CMUDict and NetTalk. In Table 5, I compared the performance of the Transformer 4x4 model with previously state-of-the-art results on both CMUDict and NetTalk databases.
The first column shows the dataset, the second column presents the method used in previous solutions with references, PER and WER columns tell the results of the referred models, and the last column presents information about the number of parameters (weights). For NetTalk, I can exceed previous results significantly. I should point out that the results of the **Transformer 4x4** model are close to encoder CNN with residual connections, decoder Bi-LSTM model (**CNN+RES_BI-LSTM**) regarding PER, but WER is better in the proposed model.

Table 5. Results on the CMUDict and NetTalk datasets.

Data	Method	PER (%)	WER (%)	Model size
NetTalk	Joint sequence model [5]	8.26	33.67	N/A
	Encoder-decoder with global atten. [8]	7.14	29.20	N/A
	Encoder CNN with res. conn, decoder Bi-LSTM (**CNN+RES_BI-LSTM**) [J1]	5.69	30.10	14.5 M
	Transformer 4x4	**6.87**	**29.82**	**1.95M**
CMUDict	Encoder-decoder LSTM [6]	7.63	28.61	N/A
	Joint sequence model [4]	5.88	24.53	N/A
	Combination of sequitur G2P and seq2seq-atten. and multitask learn. [30]	5.76	24.88	N/A
	Deep Bi-LSTM with many-to-many alignment [31]	5.37	23.23	N/A
	Joint maximum entropy (ME) n-gram model [29]	5.9	24.7	N/A
	Encoder CNN, decoder Bi-LSTM (**CNN+RES_BI-LSTM**) [J1]	4.81	25.13	14.5 M
	End-to-end CNN (**CNN+RES**) [J1]	5.84	29.74	7.62 M
	Encoder-decoder LSTM (**LSTM_LSTM**) [J1]	5.68	28.44	12.7 M
	Transformer 4x4	**5.23**	**22.1**	**2.4 M**

Conclusions

Various sequence-to-sequence (seq2seq) models for the G2P task were described, and the results are compared to previously reported state-of-the-art research. In **CNN+RES** and **CNN+RES_BI-LSTM**, I applied CNNs with residual connections. The **CNN+RES_BI-LSTM** model, which uses convolutional layers with residual connections as encoder and Bi-LSTM as decoder outperformed most the previous solutions on the CMUDict and NetTalk datasets in terms of PER. Furthermore, **CNN+RES**, which contains convolutional layers only, is

significantly faster than other models and still has competitive accuracy. My solution achieved these results without explicit alignments. The experiments are conducted on a test set, which is 9.8% and 24.9% of the whole CMUDict and NetTalk databases, respectively.
I also investigated a novel transformer architecture for the G2P task. **Transformer 3x3** (3 layers encoder and 3 layers decoder), **Transformer 4x4** (4 layers encoder and 4 layers decoder), and **Transformer 5x5** (5 layers encoder and 5 layers decoder) architectures were presented including experiments on CMUDict and NetTalk. I evaluated PER and WER, and the results of the proposed models are very competitive with previous state-art results. The number of parameters (weights) of all proposed models is less than in case of the CNN and the recurrent models. As a result, the time consumption of training process decreased.
The same test set is used in all cases, so I consider the results comparable. To draw conclusions on whether one model is better than another, the goal must be defined. If inference time is crucial, then smaller model sizes are favourable (e.g. **CNN+RES**), but if lower WER and PER are the main factors, then **CNN+RES_BI-LSTM** and **Transformer 4x4** outperform the others.

4.2 Thesis Group II. Text normalization

Alongside ordinary words, natural-language texts also contain non-standard words, such as currency names, dates, addresses, e-mail addresses, measurements, etc. Before using these text elements in a Text-To-Speech synthesis (TTS) or Automatic Speech Recognition (ASR) system, it is necessary to normalize these parts by replacing them with an appropriate ordinary word or word sequence. That reason text normalization is a critical step in the variety of tasks involving speech and language technologies. In this thesis group, I implemented LSTM-based, Bi-LSTM-based, and a novel CNNs-based for text normalization. I compared the performance of these models with the same dataset.

The proposed models for text normalization have 2 steps:

1. Identify the class for each word (token).

2. Generate an output depending on its class in their fully expanded form.

I classified tokens between 16 classes. I created token sequences, which combine three consecutive tokens for the input of the presented models. For instance, the input and output of '<SE> I wake up at 9:00 AM . </SE>' is shown in Table 6.

Table 6. Creating input and output of ‘<SE> I wake up at 9:00 AM . </SE>‘.

Input	Output
‘<SE> I wake	PLAIN
I wake up	PLAIN
wake up at	PLAIN
up at 9:00 AM	PLAIN
at 9:00 AM .	TIME
9:00 AM . </SE>	PUNCT

The class of the middle token is the output for each token sequence. All three models were used according to the sequence classification architecture presented in Figure 4.

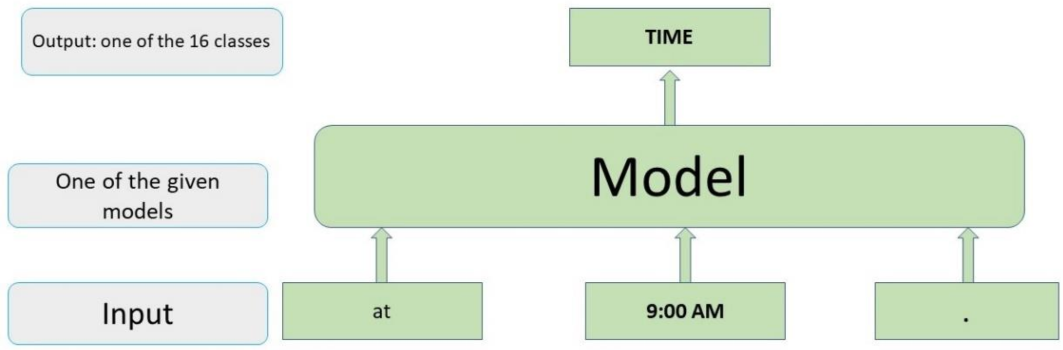


Figure 4. The sequence classification architecture for the text normalization task. The sequence of ‘at 9:00 AM . ‘ is fed to the model. The output is **‘TIME’**, which is the class of middle token - **‘9:00 AM’**.

LSTM-based Text normalization

Thesis II.1. [J2, O1] *I built and implemented deep learning-based text normalization systems: **RNN-based (LSTM_TN)**, **BI-LSTM (BILSTM_TN)** based and a novel CNN-based alternative with **continuous bag-of-words (CBOW)** and the **skip-gram (SG)** embedding method.*

I implemented 2 RNN based models, which one is LSTM and one is Bi-LSTM for text normalization, and it was called as **LSTM_TN** and **BILSTM_TN** respectively. I used the continuous-bag-of-words (CBOW) method for embedding dataset in these models.

In **LSTM_TN**, LSTM reads the input sequence one token sequence at a time and predicts one token at a time as the output sequence. During the training process, the output sequence is given to the model. The model is trained to maximize the crossentropy of the correct sequence given its context. During inference, it predicts the class of a given word, taking into account its context. LSTM has 1024 units, the time step is 3, and softmax activation function is used to obtain model predictions.

In **BILSTM_TN**, Bi-LSTM reads input on both directions with two sub-layers. These sublayers compute both forward and backward hidden sequences, which are combined to compute the output sequence. The parameters of training (optimization method, regularization, etc.) are identical to the settings used in case of the other models. This way, I try to ensure a fair comparison among the models.

CNN-based Text normalization

Thesis II.2. [J2, O1] *I compared the following models: **LSTM_TN, BILSTM_TN, CBOW_CNN and SG_CNN** and proved on Kaggle dataset that all models have similar performance, but the **CNN-based** models have the least parameters: it has **1.1M** parameters where **LSTM_TN** has **2.5M**.*

I created a novel CNN-based models for text normalization, and they were called **CBOW_CNN** and **SG_CNN** by based on respectively the continuous-bag-of-words (CBOW) and the Skip-gram (SG) method for embedding dataset. These models contain convolutional layers only with residual connections [38, 39]. I first apply one convolutional layer, with 512 filters to the input layer, followed by a stack of residual blocks. Through hyperoptimization, the best result was achieved by 3 residual blocks, and the number of filters in each residual block is 512, 256, 128. After these blocks, one more batch normalization layer and ReLU activation follow, and the architecture ends with a fully connected layer coupled with a softmax activation function.

Evaluation of Thesis II.1 and Thesis II.2.

I evaluated models on an open-source dataset by using accuracy, precision, recall, and F1-score metrics. The overall accuracy is presented for each model in Table 7; in Table 9 the accuracy is calculated for each class and the precision, recall, F1 score for SG + end-to-end CNN with residual connections. According to the results, all four alternatives show similar performance, but the CNN-based model has the least parameters. There are also slight differences in class-based accuracy in Table 8. The accuracies of ADDRESS, FRACTION, TIME, ORDINAL, DIGIT classes in the end-to-end model with **SG_CNN** are higher than in other models. The DECIMAL, FRACTION classes were learned better by **BILSTM_TN**. Generally, the end-to-end CNN model with SG outperformed the CBOW version of the same model.

Table 7. Results on the open-source data

	Method	Acc(%)	Time(s)	Model size
LSTM_TN	LSTM (1024 units)	99.35	887.6	4.6M
BILSTM_TN	Bi-LSTM (512 units)	99.34	1242.3	2.5M
CBOW_CNN	CBOW + end-to-end CNN (with res.connections)	99.38	929.01	1.1M
SG_CNN	SG + end-to-end CNN (with res. connections)	99.44	929.01	1.1M

Table 8. Results on the open-source dataset for each class.

Class	Number of tokens	LSTM_TN (%)	BILSTM _TN(%)	CBOW_ CNN (%)	SG_CNN (%)	SG_CNN (%)		
						Prec.	Rec.	F1-sc.
PLAIN	776586	99.7	99.7	99.7	99.8	0.99	0.99	0.99
PUNCT	197716	99.9	99.9	99.9	99.9	0.99	0.99	0.99
DATE	26646	98.72	98.76	98.90	98.99	0.99	0.98	0.98
LETTERS	15885	80.35	80.50	79.80	81.22	0.93	0.81	0.86
CARDINAL	13618	98.63	95.74	98.76	98.89	0.96	0.99	0.97
VERBATIM	8225	96.53	96.76	96.89	97.22	0.98	0.97	0.97
MEASURE	1386	93.14	88.60	93.01	91.34	0.97	0.91	0.94
ORDINAL	1566	92.46	91.76	92.46	93.99	0.98	0.94	0.96
DECIMAL	954	96.12	98.53	96.3	96.12	0.82	0.96	0.88
MONEY	574	86.75	79.79	97.9	87.97	0.90	0.87	0.88
DIGIT	594	66.16	61.11	66.83	68.01	0.90	0.68	0.77
ELECTRONIC	546	84.21	82.05	80.58	83.51	0.81	0.83	0.81
TELEPHONE	418	75.59	74.64	75.83	75.35	0.96	0.74	0.83
TIME	150	55.33	54.66	51.33	60.6	0.75	0.60	0.67
FRACTION	144	28.47	32.63	27.7	37.5	0.88	0.37	0.52
ADDRESS	57	40.35	35.08	29.8	43.85	0.89	0.43	0.57

While analyzing these errors, I found some misclassified tokens, where their verbalizations are correct. As Table 9 shows '128, 23 thousand, 375 million' are from DECIMAL class, but their prediction class was incorrect. For instance, '23 thousand' was misclassified as CARDINAL or '375 million' as MONEY. However, after verbalization the output was correct.

Table 9. Some misclassified examples with correct verbalizations

Class	**Before**	**Predicted**	**After**
DECIMAL	128	CARDINAL	one hundred twenty eight
DECIMAL	23 thousand	CARDINAL	twenty three thousand
DECIMAL	375 million	MONEY	three hundred seventy five million

Conclusion

I proposed three different models for the normalization task. These models are LSTM, Bi-LSTM and a novel end-to-end CNN architecture with residual connections. For the vector representations of words, I used CBOW and SG methods and compared them on the end-to-end model. Generally, the comparison shows that **SG_CNN** with residual connections is better than others.

4.3. Thesis Group III. Intent Detection

Intent detection classifies speakers' intent and extracts semantic concepts as constraints for natural language. In this thesis group, I combined and evaluated self-attention network (SAN) and a Bi-LSTM with different embedding methods. The proposed approach shows significant improvement by using a transformer model and deep averaging network-based universal sentence encoder compared to pretrained models, like Word2vec.

Self-attention networks for intent detection

Self-attention networks (**SAN**) have shown promising performance in various NLP tasks, especially in machine translation. One of the main points of **SANs** is that the strength of capturing long-range dependencies from all data.

Thesis III.1. *[C2] I designed and implemented a novel intent detection system which is based on **self-attention network (SAN)** and **Bi-LSTM** with using various embedding methods. The performance of the proposed model is compared with the combination of **SAN** and **LSTM** on the Snips, ATIS, Smart Speaker and Smart Light datasets. The experiments showed that the accuracy was better for the **Bi-LSTM**-based system than for the LSTM alternative.*

Thesis III.2. *[C2] I have empirically confirmed that the proposed model in **Thesis III.1** by using the **transformer model (TM)** universal embeddings show better performance than by using the **deep averaging network (DAN)** universal embeddings in terms of accuracy and **F1-score** on all datasets.*

In this sub-thesis group, I presented a novel intent detection system which is based on the self-attention network (see Figure 6).

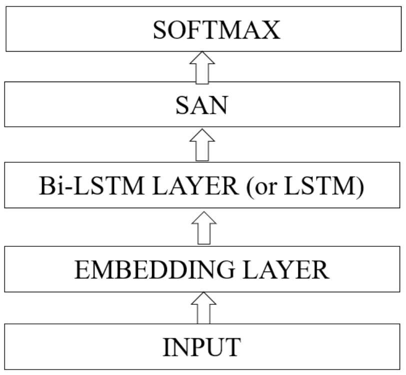


Figure 6. Proposed model architecture.

I used 300-dimension Word2vec[6] embeddings and FastText[7] word embeddings, which are trained on Wikipedia. I also investigated transformer model (TM) and deep averaging network (DAN) based universal encoder models [32], and all embeddings are combined LSTM and Bi-LSTM, as below:

1. SAN and LSTM (**SAN + LSTM**)
2. SAN and Bi-LSTM (**SAN + Bi-LSTM**)

Both proposed models encode each word to its embedding first. I did experiments with different pre-trained embeddings. As the next step, the contextual information in the input sentences (utterances) is encoded. In the first model (**SAN + LSTM**), LSTM, in the second one (**SAN + Bi-LSTM**), a Bi-LSTM was used. In SAN + Bi-LSTM, the contextual information is fed to Bi-LSTM, which combines two unidirectional LSTM layers that process the input from left-to-right and right-to-left, respectively. Both models are followed by the SAN, which is based on an attention mechanism for selecting specific parts of a sequence by relating its elements at different positions [19]. To score attention weight vectors, I used the approach from [28].

Evaluation of Thesis III.1 and Thesis III.2.

I evaluated the performance of the models by accuracy, precision, recall, F1-score on Snips, Smart Speaker, Smart Lights, and ATIS, and the results are presented in Table 10. The first column describes the proposed models, and other columns show the overall accuracy and F1-score for each dataset in Table 10.

[6] https://code.google.com/archive/p/word2vec/, Accessed: February 2021.
[7] https://fasttext.cc/, Accessed: February 2021.

Table 10. Result of proposed models for intent detection.

Model	Snips		Smart Lights		Smart Speaker		ATIS	
	Acc	F1-s.	Acc	F1-s.	Acc	F1-s.	Acc	F1-s.
Word2Vec + SAN + LSTM	94.2	0.94	91.8	0.90	94.9	0.94	93.93	0.92
FastText + SAN + LSTM	94.6	0.94	92.1	0.92	95.1	0.95	94.51	0.94
DAN + SAN + LSTM	94.1	0.94	90.2	0.90	91.7	0.90	93.56	0.93
TM + SAN + LSTM	94.2	0.94	93.6	0.93	94.2	0.93	94.81	0.94
Word2Vec+SAN+Bi-LSTM	95.6	0.96	93.8	0.94	97.7	0.98	94.49	0.93
FastText + SAN + BiLSTM	96.1	0.96	93.4	0.92	97.7	0.97	95.77	0.94
DAN + SAN + Bi-LSTM	94.2	0.94	93.2	0.93	94.7	0.95	94.91	0.93
TM + SAN + Bi-LSTM	96.5	0.97	96.6	0.97	97.7	0.98	96.81	0.95

For all datasets, SAN + Bi-LSTM consequently have shown better results than SAN + LSTM, as expected. For Snips, the accuracy of FastText + SAN +Bi-LSTM and TM +SAN +Bi-LSTM is almost the same. The result of Word2Vec + SAN + Bi-LSTM, FastText + SAN + Bi-LSTM, and TM + SAN + Bi-LSTM is almost the same for Smart Speaker. The lowest accuracy score for Smart Lights was produced by DAN + SAN + LSTM, which is 90.2%. The highest accuracy score for ATIS was produced by TM + SAN + Bi-LSTM, which is 96.81. This result for ATIS is comparable with [17, 26]. I observed that TM based universal encoder could help to improve accuracy.

Conclusion

In this thesis, I have proposed various models for intent detection. First, an end-to-end CNN model with residual connections has proposed by combined different pre-trained models. Then I have created models based on SAN and Bi-LSTM for intent detection. Utterance vectors of DAN and TM based Universal sentence encoders were also investigated. The results were evaluated with the help of confusion matrix and accuracy. I carried out experiments with SAN + LSTM, however, the accuracy was worse than with SAN + Bi-LSTM.

5. Possible applications of the scientific results

Besides the theoretical novelties of the thesis groups, their practical application is also an important factor.

In Theses I. I demonstrated a novel seq2seq based, and a Transformer based G2P conversion approach. Results of Theses I. can be applied in TTS systems, as G2P is essential for TTS. Reaching state-of-the-art performance in these systems partly depends on the accuracy of G2P conversion. In other words, inaccurate G2P conversion results in unnatural pronunciation or even incomprehensible synthetic speech.

In Theses II. I demonstrated a novel CNN-based text normalization model. The text normalization component in TTS systems is a crucial component, especially in the increased deployment of TTS systems embedded in other applications. Results of Theses II. can be applied in a range of speech and language processing applications. Text normalization for speech synthesis is heavily used in mobile and spoken assistant applications. Some errors are catastrophic, impacting not only the naturalness of the voice but the accuracy of the rendition [33, 34, 35].

In Theses III. I proposed a self-attention network-based intent detection approach. Conversational agents are exploding in popularity. Intelligent Chatbots help users accomplish specific tasks by identifying user intent from text or voice conversations using artificial intelligence. In other words, currently, most chatbot frameworks are based around the concept of intent and entity detection, which involves identifying both the intent of an utterance and the entities relevant to that intent. Besides chatbot, intent detection is applied in various fields such as e-commerce, travel consumption, medical treatment, network fraud and air target combat fields. In many applications, it is also connected with speech-to-text and text-to-speech services. Furthermore, this task is also essential for Interactive Voice Response (IVR) systems, which allows callers and computers to interact through an automated system. Cognitive IVR systems rely on artificial intelligence (AI) to understand and communicate with callers. The AI can be trained to detect the caller’s intent, speeding up issue resolution. The novel, high accuracy approaches for intent detection are important for these systems.

Furthermore, my publications are cited by researchers of large companies such as Google, Facebook, Amazon, Apple [36,37]. In [36], they presented several novels sequence-to-sequence architectures for text normalization. Besides the results of other works, they also investigated the results of the proposed models for text normalization (Section 4.4) in Table 9 [36] and made a comparison between my results and theirs.

Deep neural networks may be capable of learning universal representations for information, independent of language, and even more, that they might possibly be capable of learning some relationships between and among families of different languages. In my doctoral dissertation, proposed models for different tasks are language independent. Although models were investigated for English only, these can be applied to different languages with language-specific modifications, primarily regarding the dataset.

The results of all thesis groups were implemented in experimental systems.

Acknowledgement

Hereby I thank my supervisors, Dr. Bálint Gyires-Tóth and Prof. Géza Németh, for their continuous help and support during my research. They laid the foundations of my scientific way of thinking, and they guided me throughout my PhD studies.
I have to thank the colleagues of the Speech Technology and Smart Interactions Laboratory. Dr. Csaba Zainkó with his coding knowledge, Dr. Tamás Gábor Csapó with his experience, Bartalis István Mátyás helped me how I can solve daily problems in Hungary. They all helped me during my work and contributed to the writing of my dissertation. Besides, I thank the help of Mohammed Salah Al-Rahdi for the discussion of my research. I am grateful to Stan Chen for providing the NetTalk dataset for Thesis group I.

I would like to express my thanks to my parents; sisters and brother. Their encouraging comments have felt me very well.

I especially thank my family: my husband Agil for his encouraging comments, patience. Finally, I would like to express my thanks to my son Atilla that he accepted that I spend more time with my research, especially in the first two years. Furthermore, they continuously supported me during the years of graduate school and created a calm atmosphere that is needed for research.

The research presented in this thesis booklet, carried out by BME was supported by the Ministry of Innovation and the National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme, by the NRDI Fund based on the charter of bolster issued by the NRDI Office under the auspices of the Ministry for Innovation and Technology.

References

[1] S.Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta, and M. Shoeybi.(2017). Deep Voice: Real-Time Neural Text-to-Speech,' Proceedings of the 34th International Conference on Machine Learning, vol. 70, 195-204.

[2] Wang, Y., Skerry-Ryan, R.J., Stanton, D., Wu, Y., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q.V., Agiomyrgiannakis, Y., Clark, R., & Saurous, R.A. (2017). Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model. ArXiv, abs/1703.10135.

[3] K. Vythelingum, Y. Esteve, and O. Rosec.(2017). Error detection of grapheme-to-phoneme conversion in text-to-speech synthesis using speech signal and lexical context. In 2017 IEEE Automatic Speech Recognition and Understanding Workshop.

[4] Bisani, M., & Ney, H. (2008). Joint-Sequence Models for Grapheme-to- Phoneme Conversion. Speech Communication, 50 (5): 434–451, doi: 10.1016/j.specom.2008.01.002.

[5] Rao, K., Peng, Fuchun, Sak, H., & Beaufays F. (2015). Grapheme-to-Phoneme Conversion Using Long Short-Term Memory Recurrent Neural Networks. IEEE International Conference on Acoustics, Speech and Signal Processing, 4225–4229.

[6] Yao, K., & Zweig, G. (2015). Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion. Proceedings of the Annual Conference of the International Speech Communication Association, 3330–3334.

[7] Mousa, A., E., &, Schuller, B. (2016). Deep Bidirectional Long Short-Term Memory Recurrent Neural Networks for Grapheme-to-Phoneme Conversion Utilizing Complex Many-to-Many Alignments. Proceedings of the Annual Conference of the International Speech Communication Association, 2836–2840, doi:10.21437/Interspeech.2016-1229.

[8] Toshniwal, S., & Livescu, K. (2016). Jointly learning to align and convert graphemes to phonemes with neural attention models. IEEE Spoken Language Technology Workshop (SLT), doi:10.1109/SLT.2016.7846248

[9] Allauzen, Cyril, Michael Riley, and Brian Roark. (2016). Distributed Representation and Estimation of WFST-Based N-Gram Models. In Proceeding of ACL Workshop on Statistical NLP and Weighted Automata: 32–41.

[10] Sproat, Richard, and Navdeep Jaitly. (2016). RNN Approaches to Text Normalization: A Challenge. arXiv: preprint arXiv: 1611.00068

[11] Graves, Alex, Santiago Fernandez, Faustino Gomez, and Jurgen Schmidhuber. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. Proceedings of the 23rd international conference on Machine Learning: 369–376.

[12] Marcelo Mendoza, Juan Zamora. (2009). Identifying the Intent of a User Query Using Support Vector Machines. SPIRE 2009. Lecture Notes in Computer Science, vol 5721.

[13] Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, Xiaofei He. (2018). Dialogue Act Recognition via CRF-Attentive Structured Network. 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, 225-234.

[14] Homa B. Hashemi, Amir Asiaee, Reiner Kraft. (2016). Query Intent Detection using Convolutional Neural Networks. In International Conference on Web Search and Data Mining, Work-shop on Query Under-standing.

[15] .Xu, P., & Sarikaya, R. (2013). Convolutional neural network-based triangular CRF for joint intent detection and slot filling. 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, 78-83.

[16] V. I. Levenshtein. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals,' Soviet Physics Doklady, vol. 10, no. 8, 707–710.

[17] Goo, C., Gao, G., Hsu, Y., Huo, C., Chen, T., Hsu, K., and Chen, Y. (2018). Slot-Gated Modeling for Joint Slot Filling and Intent Prediction. Proceedings of Annual Conference North American Chapter of the Association for Computational Linguistics, 753-757.

[18] Saade, A., Coucke, A., Caulier, A., Dureau, J., Ball, A., Bluche, T., Leroy, D., Doumouro, C., Gissel-brecht, T., Caltagirone, F., Lavril, T., Primet, M. (2018). Spoken Language Understanding on the Edge. CoRR, abs/1810.12735.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017).

[20] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. CoRR, abs/1409.0473.

[21] Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. (2017). Convolutional Sequence to Sequence Learning. arXiv preprint arXiv: 1705.03122

[22] Ioffe, S., & Szegedy. C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32 International Conference on Machine Learning, PMLR 37, 448-456.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, doi:10.1109/CVPR.2016.90.

[24] Greff, K., K.Srivastava, R., & Schmidhuber, J. (2017). Highway and Residual Networks Learn Unrolled Iterative Estimation. arXiv preprinted arXiv:1612.07771.

[25] J. Ba, R., Kiros, and G. E. Hinton. (2016). Layer Normalization. CoRR, abs/1607.06450.

[26] Hakkani-Tur, D., Tur, G., Celikyilmaz, A., Chen, Y.N., Gao, J., Deng, L., and Wang, Y.Y. (2016). Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM. In Proceedings of the 17th Annual Meeting of the International Speech Communication Association, 715-719.

[27] Guo, D., Tür, G., Yih, W., and Zweig, G. (2014). Joint semantic utterance classification and slot filling with recursive neural networks. 2014 IEEE Spoken Language Technology Workshop (SLT), 554-559.

[28] Chang Xu, Cecile Paris, Surya Nepal, Ross Sparks (2018). Cross-Target Stance Classification with Self-Attention Networks. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 778-783.

[29] L. Galescu, and J.F. Allen (2002). Pronunciation of Proper Names with a Joint N-Gram Model for Bi-Directional Grapheme-to-Phoneme Conversion. 7th International Conference on Spoken Language Processing, pp. 109–112, 2002.

[30] B. Milde, C. Schmidt, and J. Köhler (2017). Multitask Sequence-to-Sequence Models for Grapheme-to-Phoneme Conversion. INTERSPEECH 2017 – 18th Annual Conference of the International Speech Communication Association, pp. 2536-2540.

[31] A. E. Mousa and B. W. Schuller (2016) Deep Bidirectional Long Short-Term Memory Recurrent Neural Networks for Grapheme-to-Phoneme Conversion Utilizing Complex Many-to-Many Alignments. INTERSPEECH 2016 – 17th Annual Conference of the International Speech Communication Association, pp. 2836-2840.

[32] Cer, D., Yang, Y., Kong, S., Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve, Y., Chris Tar, Sung, Y.H., Strope, B., Kurzweil, R. (2018). Universal Sentence Encoder. CoRR, abs/1803.11175.

[33] Mansfield, C., Sun, M., Liu, Y., Gandhe, A., & Hoffmeister, B. (2019). Neural Text Normalization with Subword Units. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). vol. 2, 190-196.

[34] Zhang, H., Sproat, R., H. Ng A., Stahlberg, F., Peng, X., Gorman, K., Roark, B. (2018). Neural Models of Text Normalization for Speech Applications. Association for Computational Linguistics, vol. 45, no. 2.

[35] Gokcen, A., Zhang, H., Sproat, R. (2019) Dual Encoder Classifier Models as Constraints in Neural Text Normalization. Proc. Interspeech 2019, 4489-4493, DOI: 10.21437/Interspeech.2019-1135.

[36] Zhang, H., Sproat, R., H. Ng A., Stahlberg, F., Peng, X., Gorman, K., Roark, B. (2018). Neural Models of Text Normalization for Speech Applications. Association for Computational Linguistics, vol. 45, no. 2.

[37] Mansfield, C., Sun, M., Liu, Y., Gandhe, A., & Hoffmeister, B. (2019). Neural Text Normalization with Subword Units. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). vol. 2, 190-196.

Publications:

Journal Papers

[J1] Yolchuyeva, S., Németh, G., and Gyires-Tóth, B. (2019). Grapheme-to-Phoneme Conversion with Convolutional Neural Networks. Applied Science. 2019, 9(6), 1143; doi:10.3390/app9061143. (WoS, IF = 2.2).

[J2] Yolchuyeva, S., Németh, G., and Gyires-Tóth, B (2018). Text normalization with convolutional neural networks. International Journal of Speech Technology. (2018) 21: 589. (WoS, IF = 1.22).

[J3] Mamedov K. Sh., Mamedov K.K., Yolchuyeva S.K. (2015) Solving the Mixed-integer Knapsack Problem by Decrease of Dimension and Use of Dynamic Programming. Automatic Control and Computer Sciences, 2015, Vol. 49, No. 4, pp. 231-238. (IF = 0.67)

[J4] Mamedov K., Mamedov K., Yolchuyeva S. (2014). Reduction of Dimension to Solve Mixed-Integer Knapsack Problem and Using Dynamic Programming Method, British Journal of Sciences, Education and Culture, London University Press, July-December 2014, p. 387-392.

[J5] Mamedov K., Nagiyeva (Yolchuyeva) S. (2013), The Method of Constructing Suboptimal Solutions of Mixed-Boolean Programming Problems with multiple restrictions, Transactions of the National Academy of Sciences of Azerbaijan, Series 3, 2013, p. 59-69.

Conference Papers

[C1] Yolchuyeva, S., Németh, G., and Gyires-Tóth, B. (2019). Transformer based Grapheme-to-Phoneme Conversion. Proc. Interspeech 2019. 2095-2099, doi: 10.21437/Interspeech.2019-1954.

[C2] Yolchuyeva, S., Németh, G., and Gyires-Tóth, B. (2019) Self-Attention Networks for Intent Detection. In: Recent Advances in Natural Language Processing, Sep 2–4 2019, Varna, Bulgaria.

[C3] Yolchuyeva, S., Németh, G., and Gyires-Tóth, B. (2019). In: XV. Magyar Számítógépes Nyelvészeti Konferencia, január 24-25,2019, Szeged, Magyarország.

[C4] Mammadov K., Sevinj N,, and Hasan V. (2012), Methods of constructing suboptimal solution of multiple Mixed-Boolean programming problems. IV International Conference 'Problems of Cybernetics and Informatics' (PCI), doi: 10.1109/ICPCI.2012.6486468.

Other Publications

[O1] Yolchuyeva, S., (2018). New NLP Methods for Improved Text-To-Speech Synthesis. Google Speech Summit 2018, 2-4 May, London.

Citations

J1:

1. Meersman, R., (2019). Grapheme-to-phoneme conversion using neural networks. Master thesis. Ghent University, Belgium.

2. Zach Ryan, Mans Hulden (2020). Data Augmentation for Transformer-based G2P. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 184–188.
3. Kaili Vesik, Muhammad Abdul-Mageed, Miikka Silfverberg (202). One Model to Pronounce Them All: Multilingual Grapheme-to-Phoneme Conversion With a Transformer Ensemble. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 146–152.
4. Nikhil Prabhu, Katharina Kann (2020). Frustratingly Easy Multilingual Grapheme-to-Phoneme Conversion. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, 123–127.
5. Beáta Lőrincz (2020). Concurrent phonetic transcription, lexical stress assignment and syllabification with deep neural networks. 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Volume 176,108-117.
6. Stan, A. (2020) RECOApy: Data Recording, Pre-Processing and Phonetic Transcription for End-to-End Speech-Based Applications. Proc. Interspeech 2020, 586-590, DOI: 10.21437/Interspeech.2020-1184.
7. Suyanto Suyanto, Andi Sunyoto, Rezza Nafi Ismail, Ema Rachmawati, Warih Maharani (2021). Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization, Journal of King Saud University - Computer and Information Sciences,doi: https://doi.org/10.1016/j.jksuci.2021.01.006

J2:

1. Mansfield, C., Sun, M., Liu, Y., Gandhe, A., & Hoffmeister, B. (2019). Neural Text Normalization with Subword Units. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). vol. 2, 190-196.
2. Zhang, H., Sproat, R., H. Ng A., Stahlberg, F., Peng, X., Gorman, K., Roark, B. (2018). Neural Models of Text Normalization for Speech Applications. Association for Computational Linguistics, vol. 45, no. 2.
3. Gokcen, A., Zhang, H., Sproat, R. (2019) Dual Encoder Classifier Models as Constraints in Neural Text Normalization. Proc. Interspeech 2019, 4489-4493, DOI: 10.21437/Interspeech.2019-1135.
4. Conkie, A., & Finch, A. (2020). Scalable Multilingual Frontend for TTS. ArXiv, abs/2004.04934. .
5. Guozhang, Z., Chenkai, M., Wenxian, F., and Zhang R.,. (2020). A Classification Approach to Text Normalization. 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE).
6. Tian Tian. (2019). Domain Adaptation and Model Combination for the Annotation of Multi-source, Multi-domain Texts. PhD thesis. New Sorbonne University Paris 3, France.
7. Jiazhao Li, Corey Lester, Xinyan Zhao, Yuting Ding, Yun Jiang and V.G.Vinod Vydiswaran. (2020). PharmMT: A Neural Machine Translation Approach to Simplify Prescription Directions. Findings of the Association for Computational Linguistics: EMNLP 2020.

C1:

1. Weiran Wang, Yingbo Zhou, Caiming Xiong, Richard Socher (2020). An investigation of phone-based subword units for end-to-end speech recognition. arXiv:2004.04290v2.

2. Zach Ryan, Mans Hulden (2020). Data Augmentation for Transformer-based G2P. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 184–188.
3. Kaili Vesik, Muhammad Abdul-Mageed, Miikka Silfverberg (2020). One Model to Pronounce Them All: Multilingual Grapheme-to-Phoneme Conversion With a Transformer Ensemble. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 146–152.
4. Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya McCarthy, Shijie Wu, Daniel You (2020). The SIGMORPHON 2020 Shared Task on Multilingual Grapheme-to-Phoneme Conversion. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 40-50.
5. Omnia ElSaadany, Benjamin Suter. (2020). Grapheme-to-Phoneme Conversion with a Multilingual Transformer Model. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. 85–89.
6. Ya. S. Pikalyov, T. V. Yermolenko. (2019). System of Automatic Transcription Generation of Russian-Language Words Exceptions on the Basis of Deep Learning. Problems of Artificial Intelligence 2019 №4 (15), 35-50.
7. Nikhil Prabhu, Katharina Kann (2020). Frustratingly Easy Multilingual Grapheme-to-Phoneme Conversion. Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, 123–127.
8. Beáta Lőrincz (2020). Concurrent phonetic transcription, lexical stress assignment and syllabification with deep neural networks. 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Volume 176,108-117.
9. Stan, A. (2020) RECOApy: Data Recording, Pre-Processing and Phonetic Transcription for End-to-End Speech-Based Applications. Proc. Interspeech 2020, 586-590, DOI: 10.21437/Interspeech.2020-1184.
10. Suyanto Suyanto, Andi Sunyoto, Rezza Nafi Ismail, Ema Rachmawati, Warih Maharani (2021). Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization, Journal of King Saud University - Computer and Information Sciences,doi: https://doi.org/10.1016/j.jksuci.2021.01.006.