TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
55/2018

Neural Networks for Language Modeling and Related Tasks in Low-Resourced Domains and Languages

OTTOKAR TILK

TTÜ PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

This dissertation was accepted for the defence of the degree of Doctor of Philosophy in Informatics on June 15th, 2018.

Supervisors: Tanel Alumäe, Ph.D.
Tallinn University of Technology
Tallinn, Estonia

Prof. Emer. Leo Võhandu
Tallinn University of Technology
Tallinn, Estonia

Opponents: Ebru Arısoy Saraçlar, Ph.D., Assistant Professor
MEF University
Istanbul, Turkey

Anton Ragni, Ph.D., Senior Research Associate
University of Cambridge
Cambridge, United Kingdom

Defence of the thesis: August 30th, 2018, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for doctoral or equivalent academic degree.

Ottokar Tilk _____
signature

ISSN 2585-6898 (publication)
ISBN 978-9949-83-321-4 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-322-1 (PDF)

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
55/2018

Tehisnärvivõrgud keele modelleerimise ja sellega seotud ülesannete jaoks väheste ressurssidega valdkondades ja keeltes

OTTOKAR TILK

Contents

List of publications

This dissertation is based on the following publications:

I Tilk, O. and Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human Language Technologies – The Baltic Perspective*, volume 268, pages 149–152. IOS Press, DOI: 10.3233/978-1-61499-442-8-149

II Tilk, O. and Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687. ISSN: 1990-9770, `https://www.isca-speech.org/archive/interspeech_2015/i15_0683.html`

III Tilk, O. and Alumäe, T. (2016). Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051. DOI: 10.21437/Interspeech.2016-1517

IV Kurimo, M., Enarvi, S., Tilk, O., Varjokallio, M., Mansikkaniemi, A., and Alumäe, T. (2017). Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987, ISSN: 1574-0218, DOI: 10.1007/s10579-016-9336-9

V Tilk, O. and Alumäe, T. (2017). Low-resource neural headline generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 20–26. Association for Computational Linguistics, DOI: 10.18653/v1/w17-4503

Author's contribution to the publications

I The author's contributions were participating in defining the methodology, implementing the model, running the experiments and writing most of the paper in collaboration with the co-author.

II The author's contributions were participating in defining the methodology, implementing the models, running the experiments, analysing the results and co-writing the paper.

III The author's contributions were defining the methodology and the research problem, implementing the models, running the experiments, analyzing the results and writing most of the paper in collaboration with the co-author.

IV The author's contributions were in sections about multi-domain language models and adaptation: participating in defining the methodology, implementing the models, running the perplexity experiments, analyzing the results and co-writing the sections.

V The author's contributions were defining the methodology and part of the research problem, implementing the models, running the experiments, analyzing the results and writing most of the paper in collaboration with the co-author.

Introduction

Motivation

Survival of small languages largely depends on their utility in modern use cases like voice interfaces for computer systems, automatic transcription, chatbots, automatic translation and summarization, predictive keyboards, optical character recognition and handwritten text recognition. One crucial component in these and other applications where the machine has to generate text or estimate the probability that a string is valid in a language is a language model. The quality of the language model largely depends on the amount and quality of the available training data. Small languages generally have less training data available and some applications like automatic speech recognition (ASR) systems require manual labor to produce suitable training data limiting the amount further. Thus obtaining good performance in low-resourced languages and domains requires extra effort. Neural network based language models have shown great generalizability even when training data is scarce (Gandhe et al., 2014) and flexibility in utilizing context which makes them a good candidate for low-resource scenarios. In this work we focus on three low-resource problems related to language modeling and propose solutions based on neural networks.

First, we focus on training better language models for small domains which can improve the quality of speech recognition, machine translation and all other applications where language models are used in these domains. Secondly, we present models for restoring punctuation in ASR output to improve the readability and usefulness of automatic transcripts. To utilize both text and audio based cues the model has to be trained on large texts and limited amount of prosody annotated transcripts. Finally, we propose methods, other than getting more data, to improve automatic generation of headlines for documents. Automatically generated headlines greatly simplify browsing automatic transcripts and other texts where human generated headlines are not present.

Claims

This dissertation consists of the following claims:

1. Recurrent neural network language models can benefit from a multi-domain component (Alumäe, 2013) when the dataset consists of texts from multiple small domains.

2. The multi-domain architecture can be effectively used for low-resource adaptation.

3. Recurrent neural networks are better for hidden event language modeling and enable learning a joint model on large text data and smaller prosody annotated data.

4. Pre-training all parameters of a neural headline generation model helps to utilize the available data fully and improves the quality of generated headlines on small datasets.

To support these claims we develop and empirically evaluate several neural networks based models and methods, described in the next section.

Contributions

The contributions of this thesis can be grouped in to three language modeling related areas:

- Language modeling (Claims 1 and 2):
  - Development of a multi-domain recurrent neural network language model that enables using the same recurrent language model for multiple potentially small target domains.
  - A method that enables using the multi-domain architecture for language model adaptation with very small amount of in-domain data.
- Punctuation restoration (Claim 3):
  - Development of a long short-term memory recurrent neural network punctuation restoration model for unsegmented text and an open source toolkit.
  - A bidirectional extension to the recurrent neural network punctuation restoration model for unsegmented text with an attention mechanism. This work also resulted in a public open source toolkit.

  - A two-stage training method that allows a text based punctuation restoration model to be integrated with a low-resource prosody based model while simultaneously adapting the model to the target domain.

- Headline generation (Claim 4):
  - Development of neural headline generation model pre-training methods that in conjunction enable initialization of all the parameters of the model and utilize all the available text to improve the performance on small datasets.
  - Analysis of the effect of pre-training different components of neural headline generation models.

Outline of the thesis

The rest of the thesis is organized as follows.

Chapter 1 gives a general overview of language modeling. More detailed background of subproblems is given in Chapters 2-4.

Chapter 2 is based on the publications I (Tilk and Alumäe, 2014) and IV (Kurimo et al., 2017). The chapter describes a multi-domain extension that is suitable for recurrent neural network language models and experiments on an Estonian medical domain dataset. The chapter also presents a method for using the multi-domain component for low-resource adaptation with experiments on Finnish broadcast news dataset.

Chapter 3 is based on the publications II (Tilk and Alumäe, 2015) and III (Tilk and Alumäe, 2016). The chapter presents two recurrent neural network model architectures for punctuation restoration along with a two-stage training method for utilizing smaller texts annotated with prosodic features. The proposed models are evaluated on Estonian and English.

Chapter 4 is based on the publication V (Tilk and Alumäe, 2017). The chapter describes three pre-training methods for neural networks based headline generation models to improve the performance when little data is available. The methods are evaluated and analyzed on Estonian and English datasets.

Chapter 5 concludes the thesis with a discussion about the suitability and effectiveness of the proposed methods along with some possible future work directions.

Abbreviations

ASR automatic speech recognition.

BRNN bidirectional recurrent neural network.

FFNN feedforward neural network.

GRU gated recurrent unit.

LM language model.

LSTM long short-term memory.

NLP natural language processing.

PPL perplexity.

ReLU rectified linear unit.

RNN recurrent neural network.

ROUGE recall-oriented understudy for gisting evaluation.

SER slot error rate.

SGD stochastic gradient descent.

WER word error rate.

Chapter 1

Language modeling

In this chapter we give a general overview of different language modeling approaches and how their performance is evaluated. More thorough overview of task specific backgrounds is given in corresponding chapters.

Language can be modeled on different linguistic unit levels: characters, morphemes or morphs, compound-split words or words. The larger the unit the more likely is the occurrence of out-of-vocabulary units. On the other hand, smaller units may produce nonsense words during generation. In this section we focus on word-based language models, although most of the methods work with other linguistic units as well.

Language models are used for estimating the probability of a word sequence $P(w_1, \ldots, w_k)$ which is factored into the product of the probabilities of the individual words using the chain rule:

$$P(w_1, \ldots, w_k) \approx \prod_{t=1}^{k} P(w_t|c_t) \tag{1.1}$$

where $P(w_t|c_t)$ is the probability of the word w_t given its context c_t (preceding words $w_1, \ldots, w_{t-1}$ for example). By sampling from the probability distribution at each position in the sequence the language model can also be used for generating text.

1.1 Types of language models

This section gives an overview of the most widely used models to estimate the word probability $P(w_t|c_t)$. We will not cover less common approaches like decision tree or random forest based and structured language models.

1.1.1 N-gram models

The n-gram language models are fast and relatively easy to train and are one of the most widely used type of language models. The n-gram models make

an assumption that each word depends only on the $n-1$ preceding words (Markov assumption) and uses context $c_t = w_{t-(n-1)}, \ldots, w_{t-1}$ to estimate the word probability. So the sequence probability is estimated with:

$$P(w_1, \ldots, w_k) \approx \prod_{i=1}^{k} P(w_t | w_{t-(n-1)}, \ldots, w_{t-1}) \tag{1.2}$$

The context is limited to $n-1$ words due to the shortcomings of n-gram models — n-gram models estimate the word probabilities based on the n-gram count statistics in the training corpus:

$$P(w_t | w_{t-(n-1)}, \ldots, w_{t-1}) = \frac{C(w_{t-(n-1)}, \ldots, w_{t-1}, w_t)}{C(w_{t-(n-1)}, \ldots, w_{t-1})} \tag{1.3}$$

Given a vocabulary V with size $|V|$ the maximum number of different possible n-grams N grows exponentially with increasing n-gram length:

$$N = |V|^n \tag{1.4}$$

With large vocabularies and long contexts the storage of all the counts becomes problematic, but more importantly, the fraction of valid n-grams that are never seen in the training data rises quickly (often referred to as *the sparsity issue*). As words are treated as discrete units, the n-grams with 0 counts in the training corpus would get a 0 probability according to the Equation 1.3 and the model would consider the entire sequence impossible if even one word does not match a sequence in the training corpus. Limiting n to a small number (typically 3-5) alleviates the issue and there exist additional methods to further reduce the sparsity problem. For example, the n-gram model can fall back to a shorter $n-1$-gram in case the initial n-gram has unreliable estimate or multiple estimates with different n can be interpolated. Another options is to discount the counts of observed n-grams and distribute the leftover probability mass between all unseen n-grams. These techniques are used and combined in several smoothing methods (Good, 1953; Katz, 1987; Kneser and Ney, 1995) for n-gram models. The model size can also be reduced by pruning — for example, deleting the counts that affect the training set perplexity (PPL) less than some threshold (Stolcke, 1998). Additional efforts towards helping the n-gram models generalize better to unseen n-grams have been made with class-based models (Brown et al., 1992) that automatically assign each word to a semantic or syntactic equivalence class and predictions are typically made by interpolating word-based and class-based probability estimates. To give the n-gram models some awareness about the context beyond the last $n-1$ words, cache-based models have

been proposed (Kuhn and Mori, 1990). Cache-based models use the relative frequency of recent use to estimate the probability of a word. Despite the described enhancements and extensions, the inability to utilize the context to a more full extent and poor generalizability are remaining problems of the n-gram models.

1.1.2 Maximum entropy models

Maximum entropy language model (Rosenfeld, 1994) is a log-linear model that computes the conditional word probability $P(w_t|c_t)$ based on a weighed sum of many context based features:

$$P(w_t|c_t) = \frac{e^{\sum_j \lambda_j f_j(w_t,c_t)}}{\sum_{w_k \in V} e^{\sum_j \lambda_j f_j(w_k,c_t)}} \tag{1.5}$$

where V is the vocabulary of the model, the functions f_j are feature functions and the feature weights $\lambda_j \in \Lambda$ are the learnable parameters. Several algorithms, like generalized iterative scaling (Darroch and Ratcliff, 1972) or gradient descent and their more advanced versions can be used to optimize the parameters Λ of the model. The feature functions f_j are typically binary valued functions that indicate the occurrence of a relation between a predicted word w_t and its context c_t. For example:

$$f_j(w_t, c_t) = \begin{cases} 1 & w_t = \text{modeling } \& \ \mathit{lastword}(c_t) = \text{language} \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

The feature functions offer a great flexibility and can include local context based word or word class n-gram features, but also long range trigger features (Rosenfeld, 1994). The performance of the model largely depends on the quality of features and construction of good features can be difficult.

Maximum entropy models may alternatively be viewed as a single-layer neural network with no hidden layer and as such can be integrated into a neural network language model as a submodel (Mikolov et al., 2011a).

1.1.3 Neural network models

Earliest attempts at modeling language with neural networks took place already in the 80s of the last century. For example Elman (1989) used a recurrent neural network in a small scale experiment (29-word vocabulary and a corpus of 10000 artificially generated 2- to 3-word sentences) to predict the next word based on previous ones. It took over 10 years of advancements in computing power for neural networks to gain attention in language modeling and to be used in practical experiments (Bengio et al., 2001; Schwenk and Gauvain, 2002), but instead of recurrent models, feedforward models were

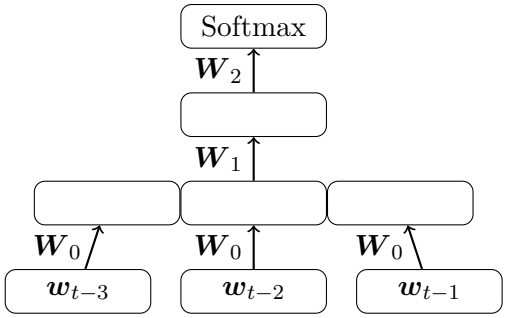


Figure 1.1: The description of a 3-gram FFNN LM architecture.

used and remained the main neural approach for almost another 10 years until Mikolov et al. (2010) developed a recurrent neural network (RNN) language model (LM) that worked well.

The feedforward neural network (FFNN) LMs are essentially an n-gram approach that predict the next word w_t based on the last $n-1$ words, but they solve the generalization problem that exists in models that treat each word as a discrete unit, by projecting each word into a lower dimensional space where words are represented by continuous vectors that are jointly and automatically learned with the rest of the model. The trained continuous vector representations of words (often referred to as distributed representations or word embeddings) are similar for words that are similar. Thus the model can generalize and assign similar probabilities to unseen but similar sequences, even if none of the word identities match a sequence in the training data as long as the words in the sequence are semantically or syntactically similar (Bengio et al., 2001). The idea of distributed representations itself is much older (Hinton, 1986), but it was Bengio et al. (2001) who finally applied it to language modeling.

A typical architecture of FFNN LMs is described in Figure 1.1. The $n-1$ context words are represented by one-hot vectors (an input vocabulary V_i sized vector with $|V_i|-1$ zeros and a 1 at the position that corresponds to the word index in the input vocabulary). The projection weights $\boldsymbol{W}_0$ are shared by all context positions and the matrix-vector multiplication in the distributed representation computation is typically replaced with a low cost word index based row lookup operation. The distributed representations of context words are then concatenated and passed through a nonlinear hidden layer. The hidden layer is followed by the computationally most expensive output layer that computes the conditional probability distribution for the

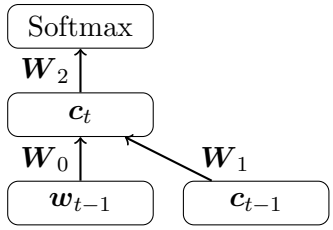


Figure 1.2: Description of a RNN LM architecture.

next word over the entire output vocabulary V_o using the Softmax function:

$$\text{Softmax}_j = \frac{e^{\boldsymbol{h}_t \boldsymbol{W}_2[j]}}{\sum_{k=1}^{|V_o|} e^{\boldsymbol{h}_t \boldsymbol{W}_2[k]}} \tag{1.7}$$

for $j = 1, \ldots, |V_o|$, where $\boldsymbol{h}_t$ is the output of the hidden layer and $[\cdot]$ selects the column vector of the matrix.

The FFNN LMs solve the generalization problem, but the limited context problem remains (although, FFNN LMs have been successfully experimented with using context of up to 29 words (Schwenk et al., 2014), it still imposes a fixed limit on the model). To solve the fixed context limitation, RNN based LMs can be used (Mikolov et al., 2010). Like FFNN LMs the RNN LMs also use distributed representations of words for better generalization, but instead of using $n-1$ context words as an input, only the last word w_{t-1} is given as input to the model at each time step t and the rest of the information about the context is stored in the recurrent hidden layer. The recurrent hidden layer state $\boldsymbol{c}_t$ represents the context for predicting the word w_t and is computed based on the distributed representation of the last word w_{t-1} plus an affine transformation (although Mikolov et al. (2010) omitted the bias $\boldsymbol{b}_1$) of the previous context $\boldsymbol{c}_{t-1}$ passed through a nonlinear function f:

$$\boldsymbol{c}_t = f(\boldsymbol{w}_{t-1}\boldsymbol{W}_0 + \boldsymbol{c}_{t-1}\boldsymbol{W}_1 + \boldsymbol{b}_1) \tag{1.8}$$

The context vector $\boldsymbol{c}_t$ is then passed through the Softmax output layer to obtain the conditional probability distribution for the next word w_t. A general description of the RNN LM can be seen in Figure 1.2.

The simple RNN LM by Mikolov et al. (2010) used a logistic sigmoid nonlinearity in the recurrent layer which can be difficult to train to remember long range dependencies (Bengio et al., 1994) in the sequence. To fully utilize the potential of recurrent models, long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) can be used in the recurrent layer (Sundermeyer et al., 2012). Each LSTM unit has an internal recurrence and multiple gating mechanisms to control its behavior:

- An input gate to control the flow of new information into the unit.
- An output gate to control when the internal information should be exposed to the rest of the model.
- A forget gate that erases the stored information when necessary.

The internal recurrence has a one-to-one connection with weight 1 and identity activation that enables it to store information for many time steps and the gradients to flow back in time without vanishing or exploding.

The main disadvantage of neural networks based language models is their computational complexity mainly due to the large output layer that needs to compute probabilities for all words in the output vocabulary for proper normalization. There are several methods to overcome the complexity of the output layer. For example, a much smaller shortlist vocabulary of the most frequent words can be used in the output and a faster approach like an n-gram model can be used to compute the probabilities for less frequent words (Schwenk and Gauvain, 2002; Bengio et al., 2001). Another option is to assign words into classes (Mikolov et al., 2011b) and factorize the output layer so that the output word probability can be computed based on two much smaller Softmax layers. Even greater speed-ups can be obtained by decomposing the output layer into a binary tree based on binary hierarchical clustering of a WordNet semantic hierarchy (Morin and Bengio, 2005).

The flexibility of neural network based LMs enables using them separately for a particular problem in a system (e.g., choosing the most plausible word sequence from several acoustically similar hypotheses in a speech recognition system) or to integrate them into a more complex neural model as a submodel (e.g., as a conditional LM decoder in a encoder-decoder network (Bahdanau et al., 2015) for neural machine translation or summarization) enabling end-to-end training for a given task.

1.1.4 Hidden event language models

Instead of predicting the next word given the context $P(w_i|c_i)$ like standard LMs, hidden-event LMs (Stolcke et al., 1998) estimate the probability of a hidden event (e.g., disfluency, segment boundary or punctuation) between words $P(e_i|c_i)$.

Hidden event LM uses an n-gram model that is trained on a joint sequence of words and events. Each word or event probability is estimated based on a context of $n-1$ preceding words and/or events. When testing, the n-gram model is used to estimate the transition probabilities in a hidden Markov model where word-event pairs correspond to states and words to observations. A forward-backward dynamic programming algorithm (Rabiner and Juang,

1986) is then used to compute the probability of an event $P(e_i|w_1, \ldots, w_k)$ based on the word sequence (Stolcke et al., 1998).

Another option is to use the n-gram model to compute the weights of the edges in a finite state automaton, where words correspond to nodes and edges emit events, and then find the path with the lowest cost (Gravano et al., 2009).

As the hidden event LM uses an n-gram model, the same shortcomings that were described in Section 1.1.1 apply — the context is limited and the model does not generalize well.

Despite the limitations, the n-gram based hidden event LMs have been used for punctuation restoration (Gravano et al., 2009), sentence segmentation (Stolcke and Shriberg, 1996), reconstruction of compound words (Alumäe, 2007), disfluency annotation (Stolcke et al., 1998) and other tasks.

1.2 Evaluation

To evaluate and compare language models independently, the most commonly used measure is PPL of a sequence that was not observed during training:

$$PPL = b^{-\frac{1}{k}\sum_{t=1}^{k}\log_b P(w_t|c_t)} \tag{1.9}$$

The PPL value shows how confused the model is on average per word in the test data — a $PPL = x$ can be interpreted as if the model had to choose uniformly and independently among x possibilities for each word. During training, only the exponent $-\frac{1}{k}\sum_{t=1}^{k}\log_b P(w_t|c_t)$ (i.e., the cross-entropy estimate) is generally used as the training objective.

While PPL evaluation is simpler, the usefulness for the target task can be better estimated by integrating the LM into the system and evaluating the system as a whole on the target task with appropriate metrics. For example:

- word error rate (WER) in speech recognition;
- recall-oriented understudy for gisting evaluation (ROUGE) (Lin, 2004) in summarization and headline generation;
- slot error rate (SER) (Makhoul et al., 1999) in punctuation restoration;
- bilingual evaluation understudy (BLEU) (Papineni et al., 2002) in machine translation.

A model with lower PPL might not always produce better results on the target task. One of the reasons is that the model is trained on correct sequences and therefore relies on correct context to make predictions, while as part of the whole system its input may contain errors made by other components of the system.

Chapter 2

Multi-domain recurrent neural network language modeling and adaptation

This chapter is based on the work in publications I and IV.

Research questions:

- Can the multi-domain extension for FFNN LM (Alumäe, 2013) be applied to RNN LM with similar gains in performance?
- Can the multi-domain architecture also be used for low-resource adaptation?

2.1 Background

Language models are known to perform better with more training data and when the training data matches the target domain in style (Moore and Lewis, 2010). Unfortunately, there are use cases where obtaining large amount of data from specific domain is not possible. For example, speech generally differs from written text in style, but using it for language model training requires laborious manual transcription. Sometimes the model needs to be adapted to a specific topic or domain that has a limited amount of text available. These problems are amplified for small languages.

Language model adaptation and multi-domain language models are both approaches designed to deal with the scarcity problem of target domain training data — they increase the total amount of training data by supplementing target domain with texts from other domains to improve learning general patterns in the language while biasing the final model towards the specific style of the target domain.

2.1.1 Language model adaptation

Adaptation (Park et al., 2010) usually starts with first training the language model on general text (out-of-domain data) that might not produce an ideal model for target domain, but enables the model to learn general word semantics and language structures. The general model is then adapted to the target domain (although sometimes the general and target-domain models are trained jointly (Alumäe and Kurimo, 2010)). There are multiple ways to adapt the model, but the overall goal is to minimize the loss on target domain while avoiding overfitting to the often small target domain dataset and forgetting the general patterns.

Language model adaptation can be categorized into two groups — supervised and unsupervised adaptation. In this work we focus on the supervised case.

Supervised adaptation is more accurate than unsupervised adaptation (Bacchiani and Roark (2003) report that unsupervised adaptation gives about twice as large improvement in WER compared to unsupervised adaptation) but requires target domain text or manual transcripts if the target domain is speech.

Unsupervised adaptation is useful in situations where resources for transcribing audio are not available as the transcription process can be quite laborious. Instead of manual transcripts the unsupervised adaptation approach uses the automatic transcripts produced by the ASR system with an unadapted language model to adapt the model.

One approach for both supervised and unsupervised adaptation is to add domain-specific parameters to the model after training the general model and only train the added parameters while the rest of the parameters are kept fixed during adaptation. For example, (Park et al., 2010) added a linear adaptation layer (more precisely $N - 1$ copies of the same shared layer, one for each context word) between the projection/context layer and the hidden layer in their FFNN LM during unsupervised adaptation. This approach enables the model to learn about the target domain with the added parameters, avoids overfitting by keeping the amount of domain-specific parameters relatively small, and prevents forgetting general information by keeping the rest of the parameters fixed.

Another approach is to use special regularization to keep the model from deviating too much from the general model and overfitting to the target domain. For example, Alumäe and Kurimo (2010) used a special parameter regularization in a maximum entropy language model during adaptation. Their approach is similar to L2 regularization, but instead of 0-mean Gaussian prior for parameters, it uses the parameters of the general model as the mean for the Gaussian prior and forces the parameters

to remain close to the general model unless there is good evidence in the target domain data to deviate. The regularization approach can easily be utilized in a neural network as well — in fact a method where output probabilities are constrained instead of the parameters has been used in a context dependent deep neural network hidden Markov acoustic model by Yu et al. (2013). They add Kullback–Leibler divergence as a regularization term to the objective function and show that it is equivalent to converting the target distribution from a one-hot to a soft target. The soft target consists of a linear interpolation of the ground truth from adaptation data and the output distribution from the general model (the interpolation coefficient is a tunable regularization hyperparameter).

One set of methods tries to solve the adaptation problem by changing the way the data is sampled during training. Schwenk and Gauvain (2005) use weighted sampling to train a FFNN LM where target domain samples are drawn with much higher probability than general samples (e.g., 1.0 for target domain and 0.1 for general samples). Additional benefit and their main motivation for this method is greatly reduced training time. Shi et al. (2014) propose curriculum learning (Elman, 1993) for RNN LM adaptation which uses all the available data, but changes the order of samples so general samples are seen at the beginning of training and target domain samples are presented to the model towards the end of training so they have a greater influence over the final model. Two effective method they propose either keep the target domain samples towards the end of each epoch (*Data-Sort*) or show the target domain samples only during last epochs (*All-Specific*). Similar, but slightly more sophisticated method was also used by Mikolov et al. (2011a). A simple approach (similar to *All-Specific* in Shi et al. (2014)) for unsupervised adaptation that seems to be effective in RNN LM models is retraining the model on unsupervised data for just one epoch (Kombrink et al., 2011) with different learning rate. Data selection also falls under this category. In data selection a target domain language model is used to score (e.g., cross-entropy or cross-entropy difference between target domain and general model) text segments from general text and a cutoff is optimized on validation data (Moore and Lewis, 2010). Segments below the cutoff point are left our during training (i.e., sampled with 0 probability).

Daume III (2007) proposes a general adaptation method for wide range of natural language processing (NLP) tasks that is based on creating copies of input features — a general version, a target domain version and source domain version. Source domain samples consist of general and source domain versions of features and target domain samples consist of general and target domain features. Although this method could be applied to neural network language models as well, the amount of target domain specific parameters would be too large — this is essentially equivalent to creating 3 copies of

the embedding matrix and combining the domain-specific and general word vector for each word in the sample.

N-gram models most commonly use linear interpolation (Wegmann et al., 1999) or count merging (both can be seen as different parametrizations for maximum *a posteriori* adaptation (Bacchiani and Roark, 2003)) instead of adaptation. In case of linear interpolation, a separate model is trained for each domain and a convex combination of probabilities given by component models is taken so the combined output is still a probability estimate. The interpolation weights are tuned on the validation data of the target domain. Another option is to use log-linear interpolation (Klakow, 1998) that amplifies component model agreements, but on the other hand requires renormalizing the outputs if probability estimate is desired (Broman and Kurimo, 2005). Interpolation has been also used for RNN LM models (e.g., Shi et al. (2014)), but it increases the computational cost during test time as multiple already complex models have to do a forward pass. Often a neural network language model is interpolated with a less computationally intensive n-gram model (e.g., Gandhe et al. (2014)) with improvements in performance.

Our approach in this work for the multi-domain architecture based adaptation falls under the supervised adaptation category. Similarly to Park et al. (2010) we add new domain-specific parameters to the general model, but the amount of the added parameters is much smaller enabling us to train or tune these parameters on the validation set like it is done with interpolation coefficients in language model interpolation. Training on validation data enables use-cases where the amount of target domain data is exceptionally limited.

2.1.2 Multi-domain language models

Multi-domain models are useful when there is more than one target domain. Multi-domain models enable using a single joint model for N domains instead of using N separately adapted models, reducing memory requirements and improving processing speed. These advantages may be critical on devices with limited resources or when the number of domains is large.

Alumäe (2013) proposes a multi-domain FFNN LM where the ability to use a single model for multiple domains is achieved by adding a domain switch input to the model and reserving a small portion of model parameters for domain-specific use. The majority of parameters are still allowed to learn general patterns that are common across all domains.

The adaptation method by Daume III (2007) can also be used in multi-domain models by creating $K + 1$ versions of input features where K is the number of domains. The disadvantage of this approach in case of neural network language models is a large amount of domain-specific parameters and a rapidly growing size of the model that can become prohibitive when

the number of domains is large.

Shi et al. (2014) also explore the multi-domain problem (more precisely, they deal with sub-domains and call it within-domain adaptation), but they use a separate RNN LM for each domain instead of a joint multi-domain model.

Our contribution is extending the FFNN LM multi-domain model (Alumäe, 2013) to RNN LM models. Compared to other multi-domain approaches it requires less domain-specific parameters than Daume III (2007) and does not require a separate model for each domain like Shi et al. (2014).

2.2 Multi-domain recurrent neural network language model

Our multi-domain RNN LM is based on the FFNN LM extension proposed by Alumäe (2013). The model augments the classic FFNN LM architecture by replacing the hidden layer weights of the model with a factored order-3 rank-d tensor, where d is the size of the factor layer. This allows each domain to select a domain specific slice from the factored tensor and use it as a domain specific hidden layer weight matrix. The domain specific weight matrices allow the model to share general information across domains as the tensor is factored into three matrices out of which two are shared across domains, while also factoring in domain specific differences with the third factorization matrix. Partially domain-specific parameters enable the FFNN LM to complement the relatively small local n-gram context with additional information about the general style and topic of the current target domain. One of our goals is to find out whether the information about the current target domain is also useful in RNN LMs that can potentially use a much larger context than FFNN LMs and derive the general topic and style information from there.

Preliminary experiments revealed that a direct application of the FFNN multi-domain extension to RNNs does not work and actually degrades the performance of the recurrent model in some cases. The unexpected degradation motivated further experiments and analysis which revealed the cause as the combined effect of two factors:

- Mathematical properties of the factored tensor.
- The sequential processing of the word stream in RNN LMs.

The next section describes the problems in detail.

2.2.1 Problem

Mathematical properties of the factored tensor in backpropagation

Similarly to the multi-domain FFNN LM we placed the factored weight tensor right after the context representation layer so that the domain-specific weight matrix processes the context of sample t (recurrent layer state at time step t in case of RNN LMs; $n-1$ concatenated word embeddings of tth n-gram sample in case of FFNN LMs). We omitted the factor bias included in the original model by Alumäe (2013) as similar effect (avoiding scaling of forward and backward signals too close to zero) can be achieved by initializing domain specific weights to ones.

Given context representation $\boldsymbol{c}$, the forward pass for the described LM with factored tensors is:

$$\hat{\boldsymbol{y}} = \text{Softmax}\left((\boldsymbol{d}\boldsymbol{W}_d \circ \boldsymbol{c}\boldsymbol{W}_c)\boldsymbol{W}_y + \boldsymbol{b}_y\right) \tag{2.1}$$

and loss L is:

$$L = -\sum_i \boldsymbol{y}_i \log \hat{\boldsymbol{y}}_i \tag{2.2}$$

where $\boldsymbol{d}$ and $\boldsymbol{y}$ are one-hot encoded domain input and target word vectors respectively, and $\circ$ denotes an element-wise (Hadamard) product. Here we have omitted the class output layer for simplicity, but it still illustrates the problem without loss of generality.

In case of factored tensors the gradient of the loss function L with respect to context vector $\boldsymbol{c}$ is:

$$\nabla_{\boldsymbol{c}} L = \left(\boldsymbol{d}\boldsymbol{W}_d \circ \left((\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{W}_y^T\right)\right)\boldsymbol{W}_c^T \tag{2.3}$$

The problem is in the term $\boldsymbol{d}\boldsymbol{W}_d$ (the domain vector) that multiplies all the gradients that are backpropagated through the factored tensor. The domain vector has a similar contribution to the backpropagated gradients as the error signal — both are a linear combination of the rows or columns of one weight matrix.

Sequential processing in recurrent neural network language models

Unlike FFNN LM models, which randomly sample n-grams, the RNN LM models process the text sequentially. This opens opportunities like classed output layer (Mikolov et al., 2011b), that enable using full vocabulary with little computational overhead, but can render some methods that are effective with FFNN LM models ineffective in RNN LM models.

With sequential processing the model sees all the samples from each domain one-by-one. This entails that for long durations the domain vector remains constant and, in case of factored tensors, appears in backpropagated gradients as a significant factor. This can cause the model to fit well to the domains appearing at the end of the sequence while rendering the domain vectors of the domains at the beginning of the sequence less compatible with the rest of the parameters of the model.

To verify the hypotheses we performed experiments with two FFNN LM models:

- conventional FFNN LM with no domain-specific parameters (Bengio et al., 2001)
- multi-domain FFNN LM with domain-specific weight matrices (multiplicative interaction with domain vector) (Alumäe, 2013)

and test them with random sampling and without random sampling in original and reverse order.

Both models are trained with stochastic gradient descent (SGD) using mini-batches of 200 samples and learning rate of 0.1 until validation loss has not improved for last 5 epochs. We use shortlist vocabulary (Schwenk and Gauvain, 2002) of 1024 most frequent tokens in the training set. The embeddings are 100-dimensional, hidden layer size is 500 and uses rectified linear unit (ReLU) activation (Glorot et al., 2011). The multi-domain model uses factor layer of size 300.

As seen in Figure 2.1, it is evident that the domains towards the end of the sequence tend to be more difficult in general, but it is also clear that both the conventional single-domain model and multiplicative multi-domain model tend to perform better on domains that were seen most recently during training.

2.2.2 Solution

The solution to the described problems is simple — replacing multiplication with addition.

Although a factored tensor gives the network an entire domain-specific weight matrix, we found that this kind of power is apparently not crucial to switch between domains and a simple domain specific bias is sufficient plus it solves the problem of domain vectors contributing too much to the gradients.

When using domain-specific bias (simply replacing multiplication with addition) the forward pass becomes:

$$\hat{\boldsymbol{y}} = \text{Softmax}\left((\boldsymbol{d}\boldsymbol{W}_d + \boldsymbol{c}\boldsymbol{W}_c)\boldsymbol{W}_y + \boldsymbol{b}_y\right) \tag{2.4}$$

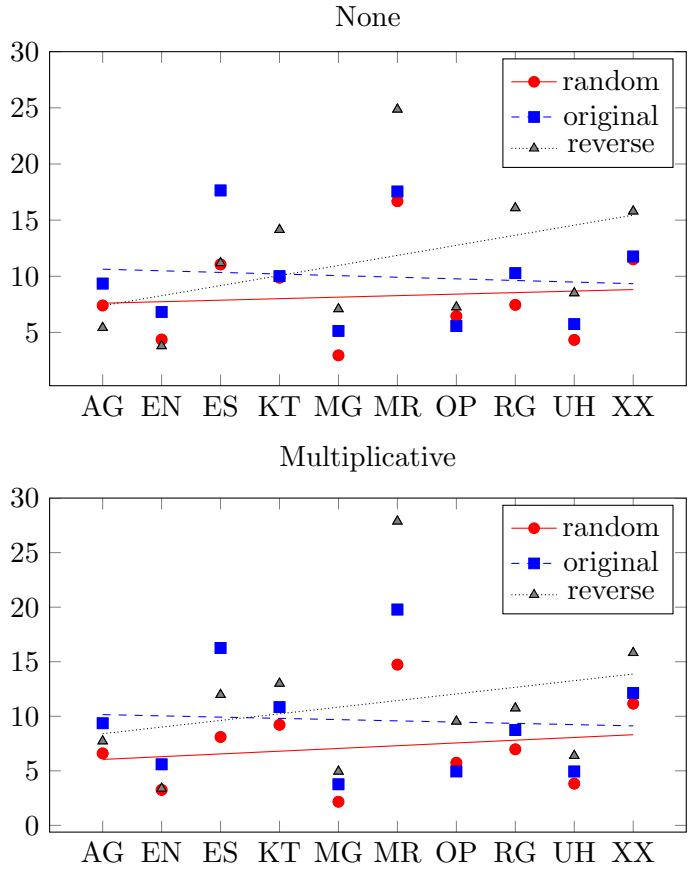


Figure 2.1: (Color online) Test set shortlist perplexities per domain for FFNN LM with different domain vector integration (none and multiplicative) and sampling methods (random sampling, original order and reverse order). Lines show linear regression trends.

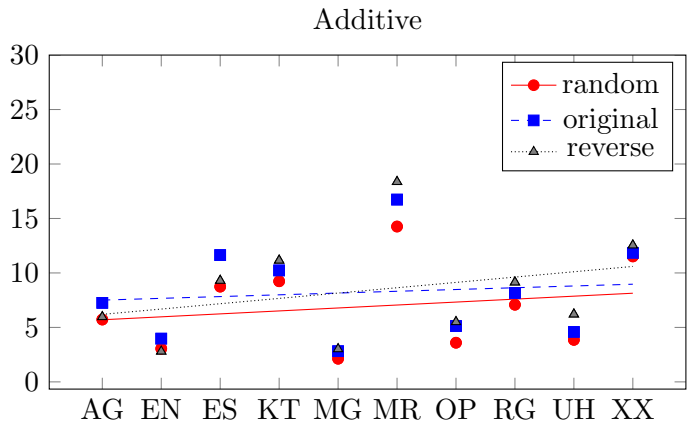


Figure 2.2: (Color online) Test set shortlist perplexities per domain for FFNN LM with additive domain vector integration and different sampling methods (random sampling, original order and reverse order). Lines show linear regression trends.

and the gradient becomes:

$$\nabla_{\boldsymbol{c}} L = \Big((\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{W}_y^T \Big) \boldsymbol{W}_c^T \tag{2.5}$$

which removes the domain vector term from the gradient.

We show empirical evidence that this simple modification solves the problem by performing the same experiments as in subsection 2.2.1 and train a multi-domain FFNN LM (Alumäe, 2013) where multiplication in the factor layer is replaced with addition. The model is tested again with and without random sampling and in the latter case the samples are processed both in the original and reverse order.

When it comes to the additive model (Figure 2.2), the slope of the trend line is much less affected by the order of samples compared to single-domain and multiplicative multi-domain model in Figure 2.1 indicating that additive interaction with domain vectors is indeed less sensitive to the sequential training. Since the gap between perplexities of different models tends to reduce towards the end of the sequence, we can interpret the additive domain vector as a domain-specific memory vector that helps the model to avoid forgetting information about domains at the beginning of the sequence while the single-domain and multiplicative multi-domain model do not have this advantage (although the multiplicative multi-domain model could potentially behave similarly, it is mostly neutralized by the mathematical characteristics described in subsection 2.2.1).

With random sampling both the additive and multiplicative multi-domain

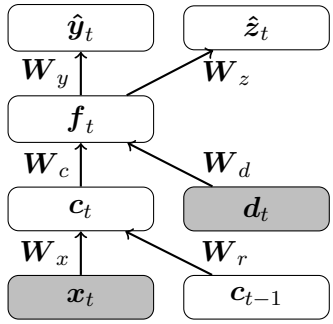


Figure 2.3: Description of MD RNN LM. Input layers are highlighted with gray background.

model perform similarly and improve the overall perplexity compared to the single-domain model. In sequential training without random sampling the additive model improves the perplexity much more than the multiplicative model.

Having shown the suitability of the additive multi-domain layer for sequential training in out preliminary FFNN LM experiments, we plugged it into the RNN LM model. To keep the multi-domain layer nonlinear we decided to add a sigmoid activation function:

$$\hat{\boldsymbol{y}} = \text{Softmax}\left(\sigma(\boldsymbol{d}\boldsymbol{W}_d + \boldsymbol{c}\boldsymbol{W}_c)\boldsymbol{W}_y + \boldsymbol{b}_y\right) \tag{2.6}$$

The word class output $\hat{\boldsymbol{z}}$ is computed similarly with $\boldsymbol{W}_y$ and $\boldsymbol{b}_y$ replaced by $\boldsymbol{W}_z$ and $\boldsymbol{b}_z$.

The full model is described in Figure 2.3. Compared to the original RNN LM model by Mikolov et al. (2010) the multi-domain RNN LM has two additional layers: the domain input layer ($\boldsymbol{d}_t$) and the factor layer ($\boldsymbol{f}_t$) with added parameters $\boldsymbol{W}_c$ and $\boldsymbol{W}_d$.

The location of the factor layer allows it to function as a compression layer (Mikolov et al., 2011b) reducing the overall computational cost of training and testing while also minimizing the number of domain-specific parameters to avoid overfitting to small domains.

2.2.3 Dataset

In our experiments we use a corpus that consists of medical notes from 10 different radiology domains (X-ray, computed tomography, ultrasound, etc.) that are all regarded as target domains. Domains vary in size greatly ranging from 27K tokens to 3.8M tokens adding up to 10.8M tokens in total. The details of the corpus are given in Table 2.1.

Our initial experiments with the corpus were performed with n-gram

Table 2.1: Sizes of domains in the radiology dataset in number of tokens.

DOMAIN	train	dev	test
AG	38075	1082	1004
EN	43255	68	222
ES	413735	1047	603
KT	3825363	2601	3408
MG	305908	143	180
MR	838995	606	994
OP	27391	405	26
RG	2372355	1462	1509
UH	2737499	1580	1197
XX	247407	991	484
total	10849983	9985	9627

models and revealed some characteristics of the corpus.

Firstly, the style and vocabulary of the medical corpus differs significantly from typical written texts or speech transcripts. This was confirmed by the fact that including written texts from newspapers and web or transcripts from broadcast news and interviews did not improve the performance of the n-gram model on the medical corpus. Therefore we do not use non-medical corpora in RNN LM experiments.

Secondly, the domains are rather different, which was apparent from the very large interpolation coefficients (mostly > 0.9) for the target domain n-gram model and very small coefficients for other domains. On the other hand, interpolated models still performed better than a single model over all domains or separate models trained only on the target domain data. Given these properties, the multi-domain approach seems like a great fit for the corpus — it can factor in domain-specific differences while also learning general patterns shared across domains.

2.2.4 Training and testing details

In our experiments we compare a single-domain and multi-domain RNN LM with a back-off n-gram language model baseline.

Both RNN LMs are a modified version of the RNN LM toolkit by Mikolov et al. (2011c) and use one model for all domains. Hyperparameters are chosen with following values:

- vocabulary size: 52293;
- number of word classes: 230;

Table 2.2: Test set full vocabulary perplexities of evaluated models.

Model	PPL
n-gram	22.2
RNN LM	20.9
MD RNN LM	17.5
MD RNN LM + n-gram	15.6

- recurrent layer size: 600;
- factor/compression layer size: 300;
- learning rate: 0.1;
- L2 regularization coefficient: 1e-7;

Both models are trained with backpropagation through time (Rumelhart et al., 1986) truncated at 3 steps. For a more fair comparison, the single-domain model uses a compression layer in place of factor layer so the multi-domain model has only 3000 (0.0063%) additional parameters compared to the single-domain model (300 per domain).

Unlike the RNN LM models, the back-off n-gram ($N = 4$) baseline uses a separate model for each domain. The domain-specific n-gram models for each domain are obtained by linearly interpolating all domain-specific models with interpolation coefficients optimized on the target domain validation set. The models were trained with the SRILM toolkit (Stolcke, 2002) and use the modified Kneser-Ney discounting.

2.2.5 Results

The results of the experiments are shown in Table 2.2. The single-domain RNN LM model already improves the perplexity by 6% relative compared to the interpolated n-gram models. The multi-domain RNN LM brings additional improvements of 16% with only 0.0063% larger amount of parameters, bringing the total relative improvement compared to the baseline to 21%. When the multi-domain RNN LM and the interpolated n-gram model probabilities are combined with equal weights, then the total relative improvement rises to 30%.

2.3 Multi-domain architecture based adaptation

In this section we explore the possibility of using the multi-domain architecture for supervised language model adaptation. Having very little

domain-specific parameters, the architecture should be useful in situations where the amount of target-domain data is exceptionally limited.

2.3.1 Method

For adaptation approach we use a simplified version of the multi-domain FFNN LM by Alumäe (2013). The architecture of our model is shown in Fig. 2.4. It differs from the architecture described by Alumäe (2013) in omitting the added adaptation layer and applying the multiplicative adaptation factors directly to the pre-activation signal of the hidden layer ReLUs. The hidden layer activations are computed as shown in Equation 2.7 where $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$ are projection and hidden layer activations respectively, $\boldsymbol{W}_{1a}$ and $\boldsymbol{b}_{1a}$ are hidden layer and $\boldsymbol{W}_{1b}$ and $\boldsymbol{b}_{1b}$ are domain adaptation weights and biases respectively. $\boldsymbol{W}_{1b}$ consists of domain-specific row-vectors (domain vectors) while $\boldsymbol{b}_{1b}$ is shared across domains. To prevent the adaptation factors from shrinking the inputs to ReLU from the start of training, the weights $\boldsymbol{W}_{1b}$ or bias $\boldsymbol{b}_{1b}$ can be initialized to ones (we used the latter in our experiments).

$$\boldsymbol{y}_1 = ReLU\big(\boldsymbol{y}_0\boldsymbol{W}_{1a} \circ (\boldsymbol{d}_t\boldsymbol{W}_{1b} + \boldsymbol{b}_{1b}) + \boldsymbol{b}_{1a}\big) \tag{2.7}$$

This kind of hidden layer enables each domain to influence the structure of sparsity in the output layer inputs (i.e. which hidden layer units are more or less likely to be exactly zero for each domain) in addition to modulating the nonzero outputs. One can consider the FFNN LM as a log-linear model on top of an automatically learned feature vector obtained by transforming the input through nonlinear transformations in lower layers (Seide et al., 2011). In this perspective the multi-domain model can influence the relevance of the log-linear model input features in the context of different domains. Our experience shows that the simplified model performs just as well or even marginally better than the original one with an additional layer.

Training multi-domain models generally requires the availability of in-domain data in the training set. With limited-resource domains it is possible that there is not enough target domain data for separate training, validation and test set. This means that there might be no in-domain data left for the training phase. We propose an adaptation approach which uses exactly the same model architecture as the multi-domain model to overcome this problem. The advantage of using the multi-domain architecture for adaptation is its resistance to overfitting due to the very small amount of domain specific parameters that need to be trained on the target domain data. The amount of domain-specific parameters is limited to a single vector with a number of elements equal to the hidden layer size (usually several hundred or thousand), which is tiny compared to the total amount of parameters in the network

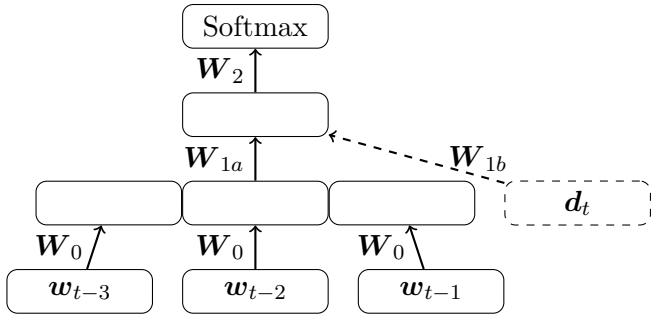


Figure 2.4: Description of the FFNN LM architecture. Dashed lines stress the parts of the network that are characteristic only to the multi-domain architecture based adapted models. The inputs (context word indices $\boldsymbol{w}_{t-1}$, $\boldsymbol{w}_{t-2}$, $\boldsymbol{w}_{t-3}$ and the domain index $\boldsymbol{d}_t$) are one-hot-encoded vectors

(usually in millions). Thus, the training error on validation data can give a good estimate of the performance on unseen data and all the available in-domain data (except the test data) can be used for adaptation.

The adaptation procedure is as follows:

1. Train a general model on out-of-domain training data using the in-domain validation data for early stopping and hyperparameter selection;

2. After the general model is ready, add the domain-specific parameters $\boldsymbol{W}_{1b}$, $\boldsymbol{b}_{1b}$ and modify the hidden layer activation according to eq. 2.7;

3. Train only the domain-specific parameters added in the previous step on the in-domain validation data until convergence, while keeping the rest of the parameters fixed.

Initially, we believed that to effectively utilize the domain vectors, the network should have a multi-domain architecture from the start and be trained as such on non-target domains. However, the preliminary experiments revealed that this is not true. The adapted model works just as well if all the multi-domain architecture specific elements are added right before training the target domain parameters.

This procedure raised a question whether the multi-domain model can also be improved by combining all the in-domain data from both training and validation set and using it to fine-tune the target domain vector as a final step of training. Unfortunately, our preliminary experiments showed that this does not significantly improve the perplexity of the test set.

Table 2.3: Finnish speech data sets

Data set	Words	Hours
fi-std-train	131005	31.4
fi-conv-train	200415	15.2
fi-conv-eval	6268	0.73
fi-news-dev	35439	5.38
fi-news-eval	37196	5.58

Table 2.4: Sizes of Finnish text data sets after preprocessing

Data set	Words
fi-general	153535459
fi-webnews	12675262
fi-newswire	31809529

2.3.2 Dataset

The evaluation of the multi-domain architecture based adaptation was carried out on the development and evaluation sets *fi-news-dev* and *fi-news-eval* (Table 2.3), which consist of Finnish broadcast news recordings collected in 2011 and 2012. For training the LMs, three data sources were used: a random subset of 23 million words from *fi-general*, a corpus of texts from Finnish web news portals (*fi-webnews*), and a corpus of newswire texts from a Finnish news agency STT (*fi-newswire*) (Table 2.4).

2.3.3 Training and testing details

In our experiments we evaluate the models in terms of PPL and WER. The models are evaluated on the Finnish broadcast news data set (*fi-news-eval* in Table 2.3). The words in the dataset are split into morphs. The PPL scores are calculated on morphs and WER scores are computed on words.

Our baseline LM is a back-off 4-gram model with modified Kneser-Ney discounting constructed over all available training data. Surprisingly, interpolating domain-specific models results in an inferior model.

It has been recently verified that FFNN LMs perform better than back-off n-gram models on under-resourced languages (Gandhe et al., 2014). One of our goals is to check whether adapted FFNN LMs bring additional improvements and what is the relationship between the relative improvement and training set size.

Four experiments are performed with each model. We start by training all the models on all available text data and continue by halving the training data for each consecutive experiment by taking every second line of the previous data set. FFNN LM hidden and projection layer size is divided by $\sqrt{2}$ every time the training data is halved. The initial hidden layer size is 500 and the projection layer size is 3×100. The FFNN LM models use a shortlist (Schwenk and Gauvain, 2002) of 1024 most frequent morphs plus an additional end of sentence token. The input vocabulary consists of 50,410 most frequent morphs plus an additional token for the beginning of sentence

and unknown morphs. When interpolating the n-gram and FFNN LM model outputs we use an equal weight of 0.5 for both models. Out-of-shortlist units are evaluated only by the n-gram model. All FFNN LMs are trained with backpropagation and mini-batch stochastic gradient descent using batch size of 200 samples and learning rate of 0.1 until the best model according to validation perplexity is not within the last 5 epochs.

In speech recognition experiments the recognition lattices were generated using systems based on the Kaldi toolkit (Povey et al., 2011), and the lattices were rescored using the FFNN LMs. Acoustic models are triphones, built using fMLLR-based speaker-adaptive training (SAT) and optimized using the boosted MMI criterion (Povey et al., 2008). Lattices are obtained after two decoding passes: first pass uses speaker-independent models, and the second pass fMLLR-transformed features with SAT-based models. The output hypotheses of the speech recognition systems consist of morphs. These were converted to word hypotheses using a hidden event LM that treats a word break as a hidden word that needs to be recovered.

2.3.4 Results

The results of PPL and WER evaluations on the test set can be seen in Tables 2.5 and 2.6 respectively. All FFNN LMs consistently outperform back-off n-gram models in PPL and WER. Utilizing FFNN LMs in addition to n-gram models gives a similar effect as using about twice as much training data: the PPL improves 7.1–10.2 % relative, statistically significant WER improvement is about 2.7–4.3 % relative. The adapted a-FFNN LMs consistently beat the unadapted FFNN LM in PPL evaluation (2.2–3.3 % relative). Unfortunately this makes no significant difference in WER. When comparing PPL on the validation and test set, it is clear that the validation PPL is a good predictor of test set performance even though it was also used as training data during adaptation phase.

2.4 Conclusions

We showed that the multi-domain extension for FFNN LMs (Alumäe, 2013) can not be directly applied to RNN LMs. This is due to the difference in the way words are sampled in each model — FFNN LMs randomly sample n-grams from text while RNN LMs process the text sequentially. The sequential processing in combination with the mathematical properties of the multiplicative multi-domain component made the RNN model overfit to more recently seen domain-specific parameters degrading the performance on domains that were present towards the beginning of the corpus. By replacing the multiplication in the multi-domain component with addition we were able to develop a method that is less affected by the sequential processing and is effective in RNNs.

Table 2.5: LM PPL with different sized training sets. Relative improvement compared to the n-gram baseline in parentheses. *a-FFNN LM* is the adapted FFNN LM

Test

Dataset size	1	1/2	1/4	1/8
n-gram	197	222	256	298
FFNN LM	183 (7.1%)	205 (7.7%)	236 (7.8%)	274 (8.1%)
a-FFNN LM	177 (10.2%)	200 (9.9%)	230 (10.2%)	268 (10.1%)

Validation

Dataset size	1	1/2	1/4	1/8
n-gram	196	223	256	300
a-FFNN LM	174 (11.2%)	199 (10.8%)	229 (10.5%)	269 (10.3%)

Table 2.6: LM test set WER with different sized training sets. Relative improvement compared to the n-gram baseline in parentheses. *a-FFNN LM* is the adapted FFNN LM

Dataset size	1	1/2	1/4	1/8
n-gram	33.3	34.0	34.9	35.7
FFNN LM	32.3 (3.0%)	32.9 (3.2%)	33.4 (4.3%)	34.3 (3.9%)
a-FFNN LM	32.4 (2.7%)	32.8 (3.5%)	33.4 (4.3%)	34.3 (3.9%)

Compared to the multi-domain FFNN LM results from Alumäe (2013) where the FFNN LM model gained only 7% (3% after combining with the n-gram model) relative on average with the inclusion of multi-domain modifications, our results on Estonian radiology dataset show much larger improvements. This was surprising as we expected the RNN to be able to derive at least some of the domain-specific characteristics from the much larger context it can utilize, which would make the explicit domain information less useful and the gap between single-domain and multi-domain models smaller. One explanation is that the simple RNN is not that good at long-range dependencies (Bengio et al., 1994), so it is possible that we would see smaller gains from the multi-domain architecture when using more advanced long-range units like LSTM (Hochreiter and Schmidhuber, 1997). However, there are also important differences between our experiments. Firstly, the dataset is different and with fewer domains so the style across domains may not vary as much as in our radiology domain and the domain vectors give less fine-grained information. Another difference is in the vocabulary handling — the FFNN LM can only affect the probabilities of the words included in the 1024 token shortlist while the RNN LM uses the full vocabulary. One hypothesis for explaining the larger performance gains in the full vocabulary multi-domain RNN LM is that the majority of the improvements from the multi-domain models come from the better modeling of less frequent words. This question needs further investigation in future work.

Although our simple solution made the multi-domain component less sensitive to the sequential processing of text, there are other solutions that would enable using the original multi-domain component by Alumäe (2013). For example, the text can be split into smaller subsequences consisting of one or more sentences and if the recurrent state is reset after each subsequence, then these subsequences can be shuffled before each training epoch like n-grams are shuffled during FFNN LM training (Chen et al., 2014). The negative side of partitioning the text into independent subsequences is that the context is now limited by the length of subsequence.

Our proposed multi-domain RNN LM requires explicit information about the domain identity during testing. An interesting future direction would be developing a component that is able to detect the domain automatically (Shi et al., 2014). Future research also includes testing the model on more datasets and performing WER experiments to verify whether the improvements in PPL also translate into improvements in the quality of ASR system output.

In our FFNN LM adaptation experiments on Finnish broadcast news dataset, we showed that the multi-domain architecture is a good fit for low-resource adaptation. By adapting only the domain-specific parameters we keep the number of adapted parameters small enough that they can be tuned directly on the validation set and the training error on the target-domain

validation set still gives an accurate performance estimate for unseen data.

The improvements in PPL we achieved with adaptation are similar to the ones of French broadcast conversations domain by Alumäe (2013) with a multi-domain model, although these experiments use different units in the vocabulary (words in Alumäe (2013), morphs in our experiments). In contrast to Alumäe (2013) our experiments showed no significant improvements in WER. Compared to the RNN LM experiments on the radiology dataset, the improvements in PPL by single-domain neural models over n-gram baseline were similar or slightly larger in our adaptation experiments. On the other hand, the RNN multi-domain model improved the PPL by 16% relative compared to the single-domain neural model on the radiology dataset while the FFNN LM adaptation experiments on Finnish broadcast news showed only 2.2–3.3 % improvement. This seems to be another indication that the better vocabulary coverage in RNN LMs enables the multi-domain or adapted models to bring larger improvements, otherwise the gap between single-domain models would have also been larger.

Our current adaptation experiments were performed on FFNN LMs, so additional experiments on RNN LMs are required in future work. The current work only verified that the multi-domain architecture can be used for adaptation, but the comparison with other adaptation methods is still required. An interesting future direction is applying the multi-domain architecture to unsupervised or online adaptation (Souvignier and Kellner, 1998).

Chapter 3

Recurrent neural networks for punctuation restoration

This chapter is based on the work in publications II and III.

Research questions:

- RNN LM models have shown advantages over traditional language models. Can RNN LM models also be used effectively as hidden event language models or more precisely for punctuation restoration?
- How can we use both large written text corpora and relatively scarce transcripts with parallel audio to utilize both lexical and prosodic features?
- Do bidirectional RNN LMs perform better than unidirectional RNN LMs? Does attention mechanism bring additional improvements?

3.1 Background

Most ASR systems output an unpunctuated sequence of words. Restoring the punctuation greatly improves the readability of transcripts and increases the effectiveness of subsequent processing, like machine translation, summarization, question answering, sentiment analysis, syntactic parsing and information extraction.

Punctuation restoration and a related task of segmentation or sentence boundary detection have been extensively studied.

Some previous approaches have used textual features only, enabling applications where audio is not available. Various methods have been used, like n-gram models (Gravano et al., 2009), conditional random fields (CRFs) (Lu and Ng, 2010; Ueffing et al., 2013), transition-based dependency parsing (Zhang et al., 2013), deep and convolutional neural networks (Che et al., 2016).

Some have treated the punctuation restoration as a machine translation task, translating from unpunctuated text to punctuated text (Peitz et al., 2011; Cho et al., 2015b).

On the other hand, there are methods that rely entirely on prosodic or audio based features, such as pause durations between words, pitch and intensity (Christensen et al., 2001; Levy et al., 2012). For example, a combination of two neural networks has been used, where the first network classifies input as speech or punctuation and the second one predicts the punctuation type (Levy et al., 2012).

Both approaches have benefits — text based approach does not require audio and has generally shown better results on reference transcripts, while prosody based models are more robust to ASR system errors — but the combination of the two brings further improvements (Kolář et al., 2006; Kolár and Lamel, 2012). Pause durations between words have been shown to be particularly helpful when combined with textual features (Christensen et al., 2001; Kolář et al., 2004). Approaches for combining textual and prosodic features can be roughly divided into two categories — a single model that utilizes both types of features, and separate models that are combined in various ways.

Single model approach has been used, for example, with maximum entropy model (Huang and Zweig, 2002; Batista et al., 2007, 2008, 2012), statistical finite state model (Christensen et al., 2001) and boosting-based classifier (Kolář et al., 2006).

A common way to combine models is to pass the outputs of the textual model along with prosodic features to the main model that makes the final punctuation decision. For example, language model posteriors can be treated as features by a decision tree (Stolcke et al., 1998), CRFs (Wang et al., 2012) or adaptive boosting algorithm (Kolár and Lamel, 2012). Another option is to use prosodic posteriors as features for a model that combines them with textual features, like in (Xu et al., 2014) where deep neural network based prosodic model posteriors were used as additional features in a text based CRFs classifier. Prosodic and textual model posteriors can also be interpolated (Stolcke et al., 1998; Kolář et al., 2004; Matusov et al., 2006; Kolář et al., 2006) or passed to a third model as features (Khomitsevich et al., 2015).

Combination of a separate textual and prosodic component makes it straightforward to achieve a greater quality textual model, as it is not limited to the availability of corresponding audio and can be separately trained on a much larger amount of text (Khomitsevich et al., 2015). Single model methods can achieve the same goal through adaptation or 2-stage training, where the model is initially trained on a large text corpus using textual features alone, and then adapted on a smaller corpus where both textual

and prosodic features are available.

In (Hasan et al., 2014), a multi-pass approach additionally refined a prosody and text based CRFs result, by taking into account the distance from the closest sentence boundary in both directions.

In this work we use two approaches. On the English dataset we use text only, as prosodic features were unavailable to us and the previous best result that we compare with. On the Estonian dataset we use textual features in combination with a prosodic feature in a single model. The prosodic feature we use is the pause duration between words, but other features can also be easily incorporated into this model. For both approaches we use two models:

- Recurrent neural network with long short-term memory.
- Bidirectional recurrent neural network (Schuster and Paliwal, 1997) in combination with an attention mechanism (Bahdanau et al., 2015).

The novelty of our work is that, to the best of our knowledge, it is the first use of both model types for punctuation restoration in unsegmented text. The source code of both models is publicly available [1]. Additionally, we propose a two-stage training method that enables a single model to effectively utilize both text only and prosody annotated datasets. In the first stage a large written text corpus is used for training textual features, and then these features are combined with the prosodic features in the second stage when the training is continued on a smaller pause annotated corpus.

The next section describes our approach in detail. Section 3.3 describes training strategies, models, data, metrics and results. Section 3.4 concludes the paper.

3.2 Methods

One of the most widely used approaches for punctuation restoration is based on the so-called hidden event LM which uses a traditional n-gram LM trained on texts that include punctuation tokens (Stolcke et al., 1998). During decoding, the LM is used to recover the most probable sequence of words and hidden punctuation symbols. Recurrent neural networks have shown several advantages over the widespread n-gram models in language modeling which we expect to transfer to hidden event language modeling as well.

First, one of the problems with n-gram models is the data sparsity issue. A better model should to be able to generalize to contexts that were not seen during training. As it is well known from language modeling, neural networks are much better at generalizing to unseen sequences, by learning

[1] `https://github.com/ottokart/punctuator` and `https://github.com/ottokart/punctuator2`

distributed representations of words (Bengio et al., 2001) or entire contexts as it is the case with RNNs (Mikolov et al., 2011b). This suggests that RNNs should be able to learn similar representations for contexts around similar punctuation and make accurate predictions even in unseen contexts.

Another weakness of n-gram models is that their context size is limited to a fixed number of tokens. Although it has been shown that increasing the context size does not help as much as getting more data (Gravano et al., 2009), it seems unjustified to expect that the relevant context size is the same for both types of punctuation and remains constant across the entire text. Therefore, one of our requirements is that the model should be able to dynamically decide on how long context is relevant. RNNs fit this requirement and are able to utilize arbitrary length sequences. Although in practice simple RNNs have difficulties in remembering long range dependencies due to vanishing gradients (Bengio et al., 1994), this problem can be alleviated by using more advanced units like gated recurrent unit (GRU) (Cho et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997).

Lastly, neural networks are very simple to augment with additional features, such as those derived from speech prosody, which gives them another advantage over n-gram models.

The main disadvantage of neural networks is their training speed, although it is not as huge problem as in language modeling as the output layer is small.

Our hidden event RNN LM approach to punctuation restoration can rely on both textual information and pause durations between words. Contrary to many previous works, we don't use any other prosodic features, such as F_0 contours, phoneme durations and energy values. This has two reasons: first, previous research (Christensen et al., 2001; Kolář et al., 2004) has shown that pause duration is by far the most useful and robust prosodic feature for predicting punctuation symbols; second, it is easy to extract pause durations from word-aligned recognition output that can be generated using any decoder, without the need to re-analyze the audio signal. We also opted against using other linguistic features, such as part-of-speech tags, because we wanted our approach to be as portable to other languages as possible.

In the next sections we first introduce two text based models followed by a general two-step method for augmenting the textual models to additionally use prosody features. The text based models can be used independently in situations where no audio with punctuation annotated transcripts is available.

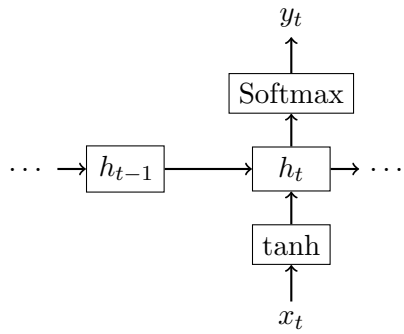


Figure 3.1: Description of the LSTM RNN model, predicting punctuation $\boldsymbol{y}_t$ at time step t for the slot before the current input word $\boldsymbol{x}_t$.

3.2.1 Long short-term memory recurrent neural network

The first text based model we propose is a unidirectional recurrent model with LSTM units that decides the suitable punctuation for a slot based on the word after the slot and the history of all previously seen words. The word after the slot is given as one-hot encoded input vector and the information about preceding words is stored in the LSTM memory units. The next word is important for predicting both commas and periods correctly. For example, in Estonian language there are many words that are almost always preceded by a comma and thus it is essential to know the following word. Also, in item listings it is indicative whether the following word is from the same category. For periods it helps to detect thematic changes and to recognize words which typically start a new sentence. Although looking only one word ahead might not be enough in all cases, the advantage of short delay is that it can be used for real-time punctuation prediction.

The forward pass of the model is described in the following equations:

$$\boldsymbol{e}_t = \tanh(\boldsymbol{x}_t \boldsymbol{W}_e) \tag{3.1}$$

$$\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{e}_t, \boldsymbol{h}_{t-1}) \tag{3.2}$$

$$\boldsymbol{y}_t = \mathrm{Softmax}(\boldsymbol{h}_t \boldsymbol{W}_y) \tag{3.3}$$

where $\boldsymbol{x}_t$ is the one-hot encoded vector representing the input word following the punctuation slot, $\boldsymbol{W}_e$ and $\boldsymbol{W}_y$ the weight matrices of embedding and output layer respectively. The LSTM unit uses forget gates (Gers et al., 2000) and peephole connections (Gers et al., 2003) as defined as by Sak et al. (2014), except biases are omitted because we found they brought no noticeable improvement. The model is described in Figure 3.1.

3.2.2 Bidirectional recurrent neural network with attention mechanism

Our second textual model is a bidirectional recurrent neural network (BRNN) (Schuster and Paliwal, 1997) which enables it to make use of unfixed length contexts before and after the current position in text.

In the recurrent layers we use GRUs (Cho et al., 2014) that are well suited for capturing long range dependencies on multiple time scales. These units have similar benefits as LSTM (Hochreiter and Schmidhuber, 1997) units while being simpler and using less parameters which is useful in a larger and more complex model.

We incorporated an attention mechanism (Bahdanau et al., 2015) into our model to further increase its capacity of finding relevant parts of the context for punctuation decisions. For example the model might focus on words that indicate a question, but may be relatively far from the current word, to nudge the model towards ending the sentence with a question mark instead of a period.

To fuse together the model state at current input word and the output from the attention mechanism we use a late fusion approach (Wang and Cho, 2016) adapted from LSTM to GRU. This allows the attention model output to directly interact with the recurrent layer state while not interfering with its memory.

Next we describe in detail how our model processes the inputs to produce the outputs. At time step t the model outputs probabilities for punctuations $\boldsymbol{y}_t$ to be placed between the previous word $\boldsymbol{x}_{t-1}$ and current input word $\boldsymbol{x}_t$. As there is no punctuation before the first word $\boldsymbol{x_1}$, the model predicts punctuations only for words $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$, where $\boldsymbol{x}_T$ is a special *end-of-sequence* token.

The sequence of one-hot encoded input words $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is first processed by a bidirectional layer consisting of two recurrent layers with GRU units, where one recurrent layer processes the sequence in forward direction and the other in reverse direction. Both recurrent layers are preceded by a shared embedding layer with weights $\boldsymbol{W}_e$. The state $\overrightarrow{\boldsymbol{h}}_t$ at time step t of the forward recurrent layer is

$$\overrightarrow{\boldsymbol{h}}_t = \mathrm{GRU}(\boldsymbol{x}_t \boldsymbol{W}_e, \overrightarrow{\boldsymbol{h}}_{t-1}) \tag{3.4}$$

where GRU is the gated recurrent unit activation function as described by Cho et al. (2014) with the exception of added biases. We use *tanh* as the new hidden state nonlinearity ϕ. The state $\overleftarrow{\boldsymbol{h}}_t$ of the reverse recurrent layer is computed similarly except the input word sequence $\boldsymbol{X}$ is processed in reverse order. The bidirectional state $\boldsymbol{h}_t$ is then constructed by concatenating the

states of the forward and backward layers at time t:

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t] \tag{3.5}$$

So this layer learns representations for each input word x_t that depend on both the preceding and following context, hopefully helping the model to better identify question indicating words as this often depends on the context (e. g. "This is *what* I do." vs. "*What* do you do?"). Also, this gives the model more information to determine whether the current word starts a new sentence or not.

The bidirectional layer is followed by a unidirectional GRU layer with an attention mechanism. This layer processes the bidirectional states sequentially and keeps track of the current position in text, while the attention mechanism can focus on relevant bidirectional context aware word representations before and after the current position. The state $\boldsymbol{s}_t$ of the layer

$$\boldsymbol{s}_t = \mathrm{GRU}(\boldsymbol{h}_t, \boldsymbol{s}_{t-1}) \tag{3.6}$$

is late fused with the attention model output $\boldsymbol{a}_t$ which is computed based on the previous state $\boldsymbol{s}_{t-1}$ and bidirectional layer states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$ as described by Bahdanau et al. (2015). The late fused state $\boldsymbol{f}_t$

$$\boldsymbol{f}_t = \boldsymbol{a}_t \boldsymbol{W}_{fa} \circ \sigma(\boldsymbol{a}_t \boldsymbol{W}_{fa} \boldsymbol{W}_{ff} + \boldsymbol{h}_t \boldsymbol{W}_{fh} + \boldsymbol{b}_f) + \boldsymbol{h}_t \tag{3.7}$$

is fed to the output layer producing the punctuation probabilities $\boldsymbol{y}_t$ at time step t

$$\boldsymbol{y}_t = Softmax(\boldsymbol{f}_t \boldsymbol{W}_y + \boldsymbol{b}_y) \tag{3.8}$$

Graphical description of the model can be seen in Figure 3.2.

3.2.3 Two-stage training for combining text and prosody

Since the amount of pause annotated data is relatively small compared to all available text, we propose a two-stage training procedure in which a purely textual model is trained in the first stage on a large text corpus. The two-stage training can be used with either of the introduced textual models and should be general enough to work with any other text based neural models.

After obtaining the fully trained textual model in the first stage, the final model utilizing text and pause duration is trained in the second stage. As proposed by Seide et al. (2011), we treat the last hidden layer outputs of the textual model as high level features learned by the network, representing everything the model knows about the textual input. The second stage model then utilizes both pause durations and these high level features of

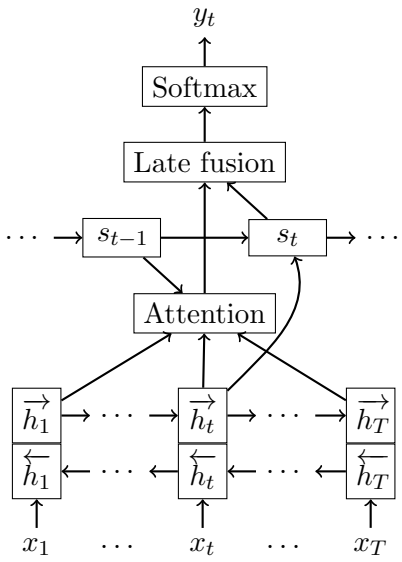


Figure 3.2: Description of the BRNN model with attention, predicting punctuation $\boldsymbol{y}_t$ at time step t for the slot before the current input word $\boldsymbol{x}_t$.

text to make decisions about punctuations. To construct a second stage model the Softmax output layer of the textual model is discarded and its last hidden layer features $\boldsymbol{x}_t^*$, concatenated with the pause duration p_t before word $\boldsymbol{x}_t$, are directed to a newly added unidirectional recurrent layer:

$$\boldsymbol{z}_t = f([\boldsymbol{x}_t^*, p_t], \boldsymbol{z}_{t-1}) \tag{3.9}$$

where f can be any recurrent unit, such as GRU or LSTM. The recurrent layer output $\boldsymbol{z}_t$ is passed to a newly initialized Softmax output layer, similar to the one described in Equation 3.8.

During the second phase of training the model learns to use the now available pause duration information in conjunction with textual features. It also enables the model to adapt to the style of speech by learning a more suitable classifier for the target domain. The remaining layers of the textual model are still used in the forward pass of the second stage model, but are fixed during training. Stacking a new classifier on top of the textual model has several advantages when compared to e.g. adapting the parameters of the existing first stage model. First, the amount of parameters that are trained on the smaller pause annotated corpus can be easily adjusted to be optimal for the smaller corpus size to prevent overfitting. Second, if the optimal size of the second stage model turns out to be smaller than the first stage model, then the training is faster and enables quicker experimentation with

different prosodic features. Finally, according to our experiments, stacking a new classifier performs better than adapting the existing parameters. The reason for that might be that as the second stage training corpus is smaller, it does not contain all the words that are in the input vocabulary of the model. Therefore some word embeddings are not updated while the rest of the model changes, causing these embeddings to become less compatible with the model.

3.3 Experiments

3.3.1 Training and testing details

The Estonian models have an input word vocabulary of 100K most frequent words in the training corpus, plus the *end-of-sequence* and *out-of-vocabulary* token. The vocabulary of our English models is constructed by taking all words that occur at least twice in the training corpus, resulting in a vocabulary of 27 244 words and the 2 special tokens.

The output vocabulary consists of the predicted punctuations (comma, period and question mark; question marks were mapped to periods in LSTM models) and a *no punctuation* token. Other punctuation symbols are either mapped to one of the punctuations in our output vocabulary or removed from corpora. For Estonian dataset, exclamation marks, semicolons and colons are mapped to periods and all other punctuation symbols are removed. In the English dataset exclamation marks and semicolons are mapped to periods, while colons and dashes are mapped to commas.

Long short-term memory recurrent neural network

The LSTM model is trained similarly in both stages. Gradients are computed with back-propagation through time (Rumelhart et al., 1986) over 5 time steps and the weights are updated using AdaGrad (Duchi et al., 2011). Learning rate starts from 0.1 and when there is no sufficient improvement on the validation set we start to divide the learning rate by 2 at each epoch (adopted from Mikolov et al. (2011c)). Training is stopped when the shrinking learning rate no longer yields enough improvements or the maximum number of 20 epochs has been reached. Weights are initialized to random uniform values in a range of ± 0.005 and all hidden states to zeros. To speed up the training, the dataset is split into 100 sequences and these sequences are fed to the network in parallel as mini-batches.

We minimize negative log-likelihood of punctuation during training. During testing we select the punctuation with highest probability according to model output at each step

The first and second hidden layer of the textual LSTM model consist of 100 tanh units and 100 single-cell LSTM blocks respectively.

The input size of the second stage LSTM model is 101 (textual features plus current slot pause duration). Hidden layer has 100 LSTM units. The second stage recurrent layer (Equation 3.9) uses the LSTM layer output $\boldsymbol{h}_t$ (Equation 3.2) as textual features $\boldsymbol{x}_t^*$ and LSTM as recurrent unit f.

Bidirectional recurrent neural network with attention mechanism

During training both stages of bidirectional models, the weights are updated using AdaGrad (Duchi et al., 2011) with a learning rate of 0.02. The L_2-norm of the gradient is kept within a threshold of 2 by normalizing it every time this threshold is exceeded (Pascanu et al., 2013).

Negative log-likelihood of the punctuation sequence is minimized during training. During testing the punctuation with highest probability according to model output is chosen at each time step. We also experimented with giving the previously predicted punctuation as an input to the model and using beam search to find the best sequence of predictions, but this caused the model to accumulate mistakes and performed worse.

The first stage of two-stage training is finished when the validation perplexity gets worse for the first time. The second stage of two-stage training and single stage training is completed when the validation perplexity has not improved in the last 5 epochs. Weights are initialized according to the normalized initialization from (Glorot and Bengio, 2010) and biases are initialized to zeros. All hidden layers consist of 256 units.

The second stage recurrent layer (Equation 3.9) uses the late fusion output $\boldsymbol{f}_t$ (Equation 3.7) as textual features $\boldsymbol{x}_t^*$ and GRU as recurrent unit f.

The models are implemented using Theano (Bergstra et al., 2010; Bastien et al., 2012) and trained on GPUs. The input sequence is partitioned into 200 word long slices. Each slice always begins with the first word of a sentence. If a slice ends with an unfinished sentence, then the unfinished sentence is copied to begin the next slice. The output sequence is one element shorter as no punctuation is placed before the first word. Slices are also used during testing, but unlike during training, the sentence boundaries predicted by the model are used. To reduce training time, the slices are shuffled before each epoch and arranged into mini-batches of 128 slices.

3.3.2 Datasets

Estonian

The Estonian dataset we use consists of two parts — a 334M word out-of-domain written text (e. g. newspapers and WWW) corpus and about 1M word in-domain pause annotated speech transcripts (broadcast news and conversations, lectures) corpus. Textual and audio data used for training the

Table 3.1: Number of tokens of textual data and the amount of hours of audio data used for training the punctuation model.

Source	#Tokens	#Hours
Newspapers	203M	
Web	74M	
Fiction	35M	
Magazines	29M	
Parliament	15M	
Social media	28M	
Lecture speech	0.3M	39.3
Broadcast news	0.1M	32.1
Broadcast conversations	0.5M	74.0
Total	386M	145.4

Estonian punctuation model is summarized in Table 3.1. As development data, we use two hours of broadcast news and 4.4 hours of broadcast conversations (radio talk shows and telephone interviews) with a corresponding 27K word transcript. The test set contains two hours of broadcast news and 5.6 hours of broadcast conversations (30K words).

The audio data is force-aligned using the speech recognition system described below. Inter-word pause durations, used for training the T-LSTM-p, TA-LSTM-p and TA-BRNN-p model, are then captured from the alignments. In order to insert reference punctuation marks into automatic transcripts, the manual transcripts were aligned with the ASR output, using minimum cost edit distance, and the punctuation marks in the reference texts were propagated to the hypothesized texts.

The ASR system that is used for aligning punctuation model training data and producing the ASR hypotheses for the evaluation data is described in detail by Alumäe (2014), although some details have been improved since then. The WER of the system is around 17%.

English

Experiments on English are performed on the IWSLT dataset which consists of TED Talks transcripts. The current best result on this dataset was achieved by Che et al. (2016). We use the same training, development and test set to train and test our models. The training and development set consist of 2.1M and 296K words respectively and come from the IWSLT2012 machine translation track training data. IWSLT2011 reference and ASR test set are used for testing and contain about 13K words each. More detailed

description of the dataset can be found in (Che et al., 2016).

3.3.3 Models

Estonian

The Estonian dataset has both out-of-domain and pause annotated target domain data available. Therefore we train our model using the two-stage approach — first training on the large out-of-domain corpus and then adapting on the pause annotated corpus. We train the two-stage Estonian model both with (TA-LSTM-p and TA-BRNN-p) and without (TA-LSTM and TA-BRNN) utilizing pause durations. The models trained without pause durations in the second stage help us assess the effect of adaptation alone and evaluate the effectiveness of the second stage training in utilizing pause duration information. To demonstrate the importance of textual features trained on a large corpus, we train an augmented textual LSTM model with additional pause duration input (T-LSTM-p) on pause annotated text only.

As this work is the first in automatic punctuation restoration for Estonian, we created our own two baselines: a hidden event LM and a decision tree that combines textual and inter-word pause information. The 4-gram hidden event LM uses a vocabulary of the same 100K words as the LSTM and BRNN models plus punctuation symbols. The model is built by interpolating the models compiled from the individual text sources using coefficients optimized on the development set. The model is smoothed using Kneser-Ney discounting and n-gram probabilities accounting for less than 10^{-7} training set perplexity improvement are pruned. The second baseline is inspired from the models proposed in (Kolář et al., 2004; Kolár and Lamel, 2012): a decision tree (4-gram+DT-p) is trained on the pause annotated corpus, using pause durations and the posterior probabilities of the punctuation symbols assigned by the hidden n-gram LM as features. This allows the decision tree to directly benefit from both inter-word pause information as well as a large text corpus which has no corresponding speech data available, similarly to the TA-LSTM-p and TA-BRNN-p model.

English

Since there is no prosody available for the English dataset we use purely textual models T-LSTM and T-BRNN in our experiments. We compare our models to the state of the art deep and convolutional neural network models by Che et al. (2016). Two-stage training, while also possible on the English dataset, would require out-of-domain data which was not used by the baselines and would give our models an unfair advantage. The baselines did use pre-trained word vectors, so we also train one T-BRNN model with

embeddings initialized to the same pre-trained word vectors [2] (T-BRNN-pre) for comparison.

3.3.4 Metrics

All models are evaluated in terms of per punctuation and overall precision, recall and F_1-score. We also report the overall SER, as F_1-score has been shown to have some undesirable properties (Makhoul et al., 1999). The metrics are based on the following statistics of the predicted punctuation: correct (C), substitutions (S), deletions (D), and insertions (I). Precision is defined as:

$$P = \frac{C}{C + S + I} \tag{3.10}$$

Recall is defined as:

$$R = \frac{C}{C + S + D} \tag{3.11}$$

F_1-score is the harmonic mean of precision and recall:

$$F_1 = 2\frac{PR}{P + R} \tag{3.12}$$

SER is defined as:

$$SER = \frac{S + D + I}{C + S + D} \tag{3.13}$$

or slot errors divided by total number of slots (Makhoul et al., 1999).

3.3.5 Results and analysis

All comparisons in this section are in terms of absolute differences. Models are evaluated on both manual transcripts and automatic transcripts generated by an ASR system.

Estonian

On the Estonian test sets (Table 3.2), it is clear that our newly proposed neural textual models (TA-LSTM and TA-BRNN) outperform the 4-gram model. The 4-gram model has decent performance in restoring commas but fails miserably when it comes to periods where especially the poor recall of 25.5 stands out. This might indicate that periods depend on longer context than the 4-gram model is able to utilize and require better generalization.

[2] `http://nlp.stanford.edu/projects/glove/`

Table 3.2: Results on Estonian reference transcripts and ASR output test set. The * indicates that the question marks have been mapped to periods.

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*SER*
Ref.	4-gram	78.3	60.2	68.1	46.5	25.5	33.0	-	-	-	71.3	50.4	59.0	61.9
	4-gram+DT-p	76.0	70.2	73.0	64.4	60.4	62.3	-	-	-	72.7	67.4	70.0	52.6
	T-LSTM-p	78.5	63.3	70.1	68.9	59.8	64.0	-	-	-	75.6	62.3	68.3	52.8
	TA-LSTM	74.5	72.2	73.3	62.8	42.9	51.0	-	-	-	71.9	63.9	67.7	52.5
	TA-LSTM-p	**82.3**	69.9	75.6	67.7	76.8	72.0	-	-	-	77.3	71.9	74.5	43.7
	TA-BRNN	75.1	**75.5**	75.3	65.6	64.1	64.8	**63.6**	43.8	51.9	72.5	71.9	72.2	46.1
	TA-BRNN-p	81.6	75.4	**78.4**	72.5	77.0	74.7	59.1	**48.7**	**53.4**	78.6	75.4	77.0	39.3
	TA-BRNN*	75.1	**75.5**	75.3	67.4	64.7	66.0	-	-	-	73.0	72.4	72.7	45.6
	TA-BRNN-p*	81.6	75.4	**78.4**	**73.8**	**77.3**	**75.5**	-	-	-	**79.2**	**76.0**	**77.6**	**38.7**
ASR	4-gram	69.9	54.4	61.2	37.7	18.6	24.9	-	-	-	63.5	44.3	52.2	74.1
	4-gram+DT-p	66.1	62.1	64.0	51.7	48.4	50.0	-	-	-	62.1	58.2	60.1	71.8
	T-LSTM-p	69.9	57.3	63.0	57.3	49.0	52.8	-	-	-	66.2	55.0	60.1	68.5
	TA-LSTM	64.5	64.6	64.6	48.8	32.2	38.8	-	-	-	61.3	55.5	58.2	72.1
	TA-LSTM-p	**71.1**	62.5	66.5	54.9	61.2	57.8	-	-	-	65.7	62.1	63.8	65.5
	TA-BRNN	63.9	**67.9**	65.8	54.0	50.3	52.1	48.8	29.9	37.0	61.3	62.6	62.0	68.7
	TA-BRNN-p	69.1	66.8	**68.0**	59.7	**61.5**	60.6	**51.2**	**31.3**	**38.9**	66.3	64.9	65.6	62.9
	TA-BRNN*	63.9	**67.9**	65.8	54.9	50.2	52.4	-	-	-	61.6	62.9	62.2	68.4
	TA-BRNN-p*	69.1	66.8	**68.0**	**60.6**	61.1	**60.8**	-	-	-	**66.6**	**65.2**	**65.9**	**62.6**

The LSTM model trained on purely textual features (TA-LSTM) achieves a $6 - 8.7\%$ improvement in F_1-score and $2 - 9.4\%$ improvement in SER over 4-gram model. While there are improvements in comma restoration, most of the gains come from better period restoration performance. This confirms that longer context and better generalization of LSTMs is indeed helpful when it comes to punctuation restoration, particularly for periods. TA-LSTM performance may be also partially attributed to the superiority of the adaptation scheme used (the 4-gram model uses linear interpolation). Just as with the 4-gram model, the TA-LSTM model is still worse at period restoration than comma restoration, although the difference is noticeably smaller. The more advanced TA-BRNN model brings further improvements and increases the gap with 4-gram model to $9.8 - 13.2\%$ in F_1-score and $5.4 - 15.8\%$ in SER despite having to deal with an additional type of punctuation. When question marks are mapped to periods, then the differences rise to $10 - 13.7\%$ in F_1-score and $5.7 - 16.3\%$ in SER.

Adding pause duration features yields noticeable reductions in error rate — improvements over corresponding purely textual model are between $3.6 - 11\%$ in F_1-score and $2.3 - 9.3\%$ in SER. Both absolute and relative improvements tend to be larger for weaker textual models and smaller for stronger textual models. As expected, the biggest benefits of using pause durations reveal themselves in large jump in period restoration recall, but precision improves as well. Commas gain noticeably in precision when utilizing pauses.

The best neural model (TA-BRNN-p) beats the 4-gram+DT-p baseline by $5.5 - 7.6\%$ in F_1-score and $8.9 - 13.9\%$ in SER and has higher performance for all punctuation marks.

When comparing T-LSTM-p, a model trained on the small pause annotated corpus only, to the TA-LSTM-p model, it becomes clear that using textual features trained on a large corpus helps a lot. Biggest improvement is in period recall, again hinting that good representations of long contexts are important for period restoration.

It is clear that all models suffer from the errors made by the ASR system. It is also evident that periods are affected more than commas. This might be another indicator that restoring periods requires larger context and as context size grows the more likely it is to encounter errors made by the ASR system. On the positive side, the gap between performance on the reference text and ASR output shows a the potential for improvements when the WER of the ASR systems improves. Another thing to note is that alignment method used to propagate reference punctuations to ASR output is not error free either. Manual inspection of punctuation restored by models revealed that many decisions made by the models that were counted as mistakes were actually better than the ground truth punctuation. While flawed, our

Table 3.3: Confusion matrix of the TA-BRNN model on the Estonian reference test set.

		Predicted			
		SPACE	COMMA	PERIOD	QUESTION
Actual	SPACE	24136	602	226	6
	COMMA	633	2502	174	4
	PERIOD	220	212	789	10
	QUESTION	15	16	14	35

Table 3.4: Confusion matrix of the TA-BRNN-p model on the Estonian reference test set.

		Predicted			
		SPACE	COMMA	PERIOD	QUESTION
Actual	SPACE	24290	446	227	7
	COMMA	688	2499	121	5
	PERIOD	169	99	948	15
	QUESTION	12	17	12	39

current ASR output test set should still give a rough performance estimate.

We also analyze the confusion matrices for the TA-BRNN and TA-BRNN-p models.

From Table 3.3 we notice that the prediction frequency of the TA-BRNN model for punctuation marks is close to the actual frequency for each punctuation mark. Commas are by far the most common punctuation mark. Most common error for commas is deletion (predicting space instead of comma), closely followed by insertion (predicting comma instead of space), and much less frequent substitution with periods. The majority of period prediction errors are roughly equally distributed between insertion, deletion, and substitution with comma. Question mark is noticeably less frequent than other punctuation marks in the test set which makes the conclusions unreliable.

Using pause duration between words as an additional feature in the TA-BRNN-p model reduces the overall amount of errors (Table 3.4). As expected, the most significant improvement is in period prediction. The number of errors where a period is substituted with comma is cut in half. Smaller but

still impressive reductions are in comma insertion, comma substitution with period, and period and question mark deletion.

English

The results on English test sets (Table 3.5) are for text based models only. We compare our models to the state-of-the art baselines by Che et al. (2016). The overall F_1-score improves by 8.9% on reference text and by 10.5% on ASR output when comparing our T-BRNN model to the best baseline (DNN-A). The best baseline in terms of SER is the DNN model and the T-BRNN model reduces it by 11.6% on reference text and by 15.5% on ASR output. The T-BRNN model shows improvements in all metrics for all punctuation types. The biggest difference is in the question mark restoration performance, as the models from (Che et al., 2016) were unable to restore any question marks due to a limited fixed size context (3 words before and 2 words after the slot) that rarely included the question indicating words that often are in the beginning of the sentence. Both the T-BRNN model and the T-LSTM model were both able to restore question marks, as their preceding context length is not limited to a fixed size. The T-LSTM model showed the lowest period restoration scores, indicating that looking further than one word into the following context is important for sentence boundary detection. The T-BRNN model showed improvements despite the fact that DNN, DNN-A and CNN-2A used an external source of information in the form of pre-trained word vectors (trained on 6B tokens). When we use the same word vectors as an initialization for our word embeddings, then we get further improvements and achieve our best result (T-BRNN-pre).

Although some of the previous work (e.g. Peitz et al. (2011)) reports comparable or even better scores, these results are not directly comparable to ours as they are reported on already segmented text while our results are achieved on unsegmented text.

Table 3.5: Results on English reference transcripts and ASR output test set. Models marked with [1] (DNN, DNN-A and CNN-2A) are the best models from (Che et al., 2016).

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*SER*
Ref.	DNN [1]	58.2	35.7	44.2	61.6	64.8	63.2	0	0	-	60.3	48.6	53.8	62.9
	DNN-A [1]	48.6	42.4	45.3	59.7	68.3	63.7	0	0	-	54.8	53.6	54.2	66.9
	CNN-2A [1]	48.1	44.5	46.2	57.6	69.0	62.8	0	0	-	53.4	55.0	54.2	68.0
	T-LSTM	49.6	41.4	45.1	60.2	53.4	56.6	57.1	43.5	49.4	55.0	47.2	50.8	74.0
	T-BRNN	64.4	45.2	53.1	72.3	71.5	71.9	67.5	58.7	62.8	68.9	58.1	63.1	51.3
	T-BRNN-pre	**65.5**	**47.1**	**54.8**	**73.3**	**72.5**	**72.9**	**70.7**	**63.0**	**66.7**	**70.0**	**59.7**	**64.4**	**49.7**
ASR	DNN [1]	47.2	32.0	38.1	59.0	60.9	60.0	0	0	-	54.4	45.6	49.6	73.3
	DNN-A [1]	41.0	40.9	40.9	56.2	64.5	60.1	0	0	-	49.2	51.6	50.4	79.2
	CNN-2A [1]	37.3	40.5	38.8	54.6	65.5	59.6	0	0	-	46.4	51.9	49.1	83.6
	T-LSTM	41.8	37.8	39.7	56.4	49.3	52.6	55.6	42.9	48.4	49.1	43.6	46.2	83.7
	T-BRNN	**60.0**	**45.1**	**51.5**	69.7	69.2	69.4	**61.5**	45.7	52.5	65.5	57.0	60.9	57.8
	T-BRNN-pre	59.6	42.9	49.9	**70.7**	**72.0**	**71.4**	60.7	**48.6**	**54.0**	**66.0**	**57.3**	**61.4**	**57.0**

Table 3.6: Confusion matrix of the T-BRNN-pre model on the English reference test set.

		Predicted			
		SPACE	COMMA	PERIOD	QUESTION
Actual	SPACE	10788	111	45	2
	COMMA	274	391	160	5
	PERIOD	125	91	582	5
	QUESTION	6	4	7	29

We also analyze the confusion matrix (Table 3.6) for the T-BRNN-pre model. The confusion matrix reveals that in the English dataset the commas and periods are roughly equally frequent. Compared to the Estonian dataset, the commas are much less frequent while periods are noticeably more common. This might indicate shorter and simpler sentences in the dataset in addition to the differences between the two languages. Prediction frequencies mostly match with actual frequencies, with the exception of commas. Most common error is comma deletion, followed by comma substitution with period. For periods the most common error is deletion, followed by substitution with comma, and finally insertion errors.

The comparison of the Estonian and English results reveals that comma restoration is a much more difficult task in English than it is in Estonian. This does not come as a surprise, as many commas in Estonian can be restored by following relatively simple rules based on the next word. Commas are also much more frequent in the Estonian dataset than they are in the English dataset, giving the model more training examples. Although there is a big difference in question mark restoration performance as well, it is hard to make conclusions as they are too rare in both test sets.

Table 3.7 shows an example of actual and generated (T-BRNN-pre) punctuation on the English reference test set. In this example, the majority of errors are deletion errors. All the deletion errors are comma deletions, which shows that commas are difficult for the model, but when it comes to the readability of the text, comma deletion errors have the smallest negative effect compared to sentence boundary punctuation deletions (periods and question marks). There are also three substitution errors in the example, but two of them are actually equally correct alternatives to periods. Overall, the generated punctuation does have a positive impact on the readability.

Table 3.7: Punctuation example. Correct, **deletion**, insertion, and *substitution* situations highlighted.

T-BRNN-pre	i 'm assuming there are not many people here who speak icelandic PERIOD so let me narrow the choices down to two PERIOD is it a happy word **SPACE** or a sad word QUESTION what do you say QUESTION okay *COMMA* some people say it 's happy PERIOD most people COMMA a majority of people **SPACE** say sad *COMMA* and it actually means sad PERIOD why do **SPACE** statistically **SPACE** a majority of people say that a word is sad **SPACE** in this case COMMA heavy in other cases *COMMA* in my theory **SPACE** language evolves in such a way that sounds match **SPACE** correspond with the subjective **SPACE** with the personal intuitive experience of the listener PERIOD
Actual	i 'm assuming there are not many people here who speak icelandic PERIOD so let me narrow the choices down to two PERIOD is it a happy word **COMMA** or a sad word QUESTION what do you say QUESTION okay *PERIOD* some people say it 's happy PERIOD most people COMMA a majority of people **COMMA** say sad *PERIOD* and it actually means sad PERIOD why do **COMMA** statistically **COMMA** a majority of people say that a word is sad **COMMA** in this case COMMA heavy in other cases *QUESTION* in my theory **COMMA** language evolves in such a way that sounds match **COMMA** correspond with the subjective **COMMA** with the personal intuitive experience of the listener PERIOD

Table 3.8: Ablation studies of the T-BRNN model. Relative change in percent in metrics on the English reference transcripts.

		No late fusion	No attention	No reverse RNN
COMMA	P	9.6	1	-1.5
	R	-21.1	-19.1	-25.2
	F_1	-8.3	-10.1	-15.1
PERIOD	P	-1.8	-1.4	-8.7
	R	3.5	-3.4	-32.1
	F_1	0.8	-2.4	-22.2
QUESTION	P	17.1	1.8	-14.4
	R	0	-29.9	-23.3
	F_1	8.2	-16.6	-19
OVERALL	P	4.7	0.6	-5.8
	R	-7.6	-11.2	-28.8
	F_1	-1.9	-5.6	-18.9
	SER	-1.2	5.4	34.2

3.3.6 Ablation studies

To better understand the individual contributions of late fusion, bidirectionality and attention, we trained additional versions of the T-BRNN model on English with the components removed:

- No late fusion — Equation 3.7 is removed from the T-BRNN model, Equation 3.6 is replaced with $\boldsymbol{s}_t = \text{GRU}(\boldsymbol{h}_t + \boldsymbol{a}_t, \boldsymbol{s}_{t-1})$, and Equation 3.8 is replaced with $\boldsymbol{y}_t = \text{Softmax}(\boldsymbol{s}_t \boldsymbol{W}_y + \boldsymbol{b}_y)$.
- No attention — Attention and late fusion (Equation 3.7) are removed from the T-BRNN model and Equation 3.8 is replaced with $\boldsymbol{y}_t = \text{Softmax}(\boldsymbol{s}_t \boldsymbol{W}_y + \boldsymbol{b}_y)$.
- No reverse RNN — Bidirectionality is removed from the T-BRNN model by replacing Equation 3.5 with $\boldsymbol{h}_t = \overrightarrow{\boldsymbol{h}}_t$.

As seen in Table 3.8, bidirectionality turned out to be the biggest factor, as removing the forward context caused the performance of all punctuation marks (especially periods and question marks) to drop. The drop in recall was much larger than in precision, indicating that the following context is a critical component for successfully recovering punctuation marks. Removing the attention (which also entails removing late fusion) had a much smaller, but still clear negative effect, affecting mostly question mark and comma

recall. Removing the late fusion alone, showed a clear performance decrease only in comma restoration recall. On the other hand, the overall precision improved.

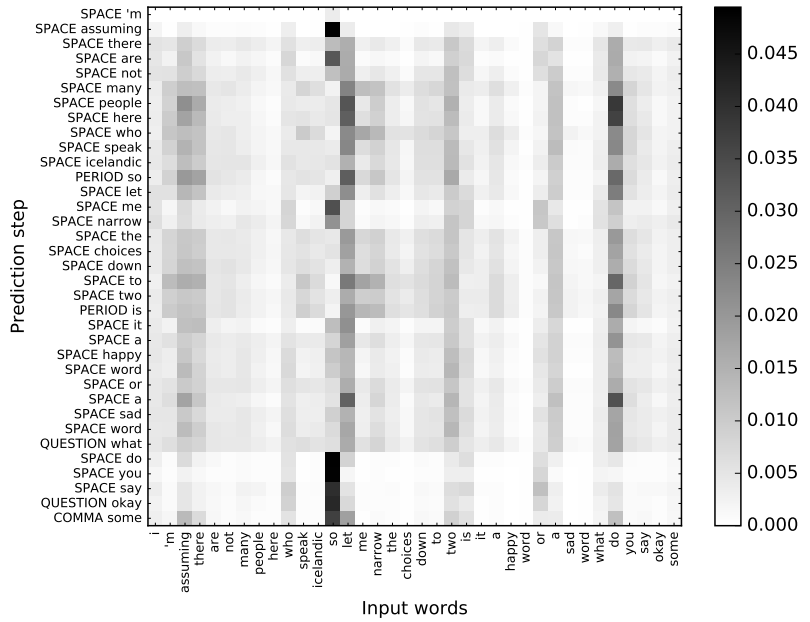
We also give examples of the attention weights of the T-BRNN-pre and *No late fusion* models in Figure 3.3. For clarity the figure shows only a 35×36 word slice of each attention matrix (the total size is 199×200). The vertical axis labels show the input word at that time step along with the punctuation mark that was predicted by the model to fill the slot before this word. Both models use attention to focus on a few keywords that remain almost constant through time steps. For both models, it is hard to interpret why the model considers exactly these words as attention worthy. The main thing that varies through time is the sharpness of the attention. The attention sharpening is especially evident in the *No late fusion* model where the otherwise smooth attention sharpens very clearly each time after a punctuation mark is predicted. T-BRNN-pre has sharper attention than the *No late fusion* model most of the time which changes minimally through time steps, even when punctuation is predicted, with the exception after predicting the first question mark. This behavior is much different from how the attention functions in sequence-to-sequence models in neural translation and summarization where focus shifts much more through time, but our model knows about the exact position in the encoded source at every step unlike sequence-to-sequence models. The almost constant behavior of attention in our model suggests that the model can be simplified by computing a single attention vector for the entire sequence (this removes $N - 1$ attention computations, where N is the number of prediction steps) and attention sharpness can be varied with a simpler and computationally less expensive subnetwork with one logistic sigmoid output for each prediction step.

The general conclusion from the ablation studies is that the attention and bidirectional recurrence are definitely useful components in a punctuation restoration model, but the late fusion should only be use when recall is valued over precision.

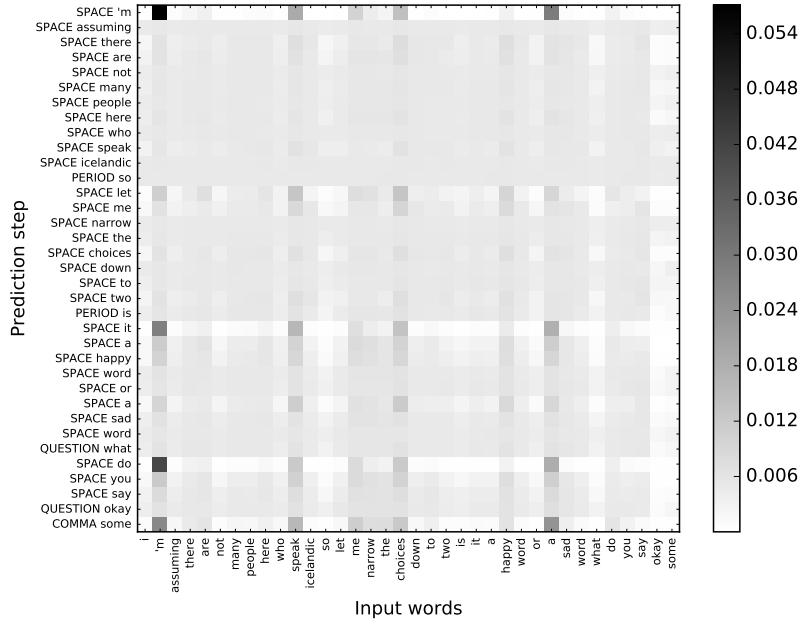
3.4 Conclusions

In this chapter we presented two recurrent neural network models for restoring punctuation marks in unsegmented transcribed speech. We presented first results on Estonian in that field and achieved state-of-the art results on the English TED Talks dataset. Also, to the best of our knowledge, the proposed LSTM model was the first published work on applying RNNs to the punctuation restoration problem.

Our results showed that the known benefits of RNN LMs over n-gram LMs did indeed transfer to hidden event language modeling as well — all



(a) T-BRNN-pre



(b) No late fusion

Figure 3.3: Attention weights examples on a section from English reference test set.

neural models outperformed the hidden event n-gram model by a large margin. Our models were able to utilize larger context, generalized better to unseen contexts and were more adapted to the target domain which all turned out to be useful in punctuation restoration.

We also showed that utilizing both large text corpora and a limited amount of prosody annotated transcripts can be effectively done with our proposed two-stage training method. The method allows for a separate tuning of the second stage hyperparameters to prevent overfitting to the annotated transcripts and as an additional benefit enables the adaptation of the model to the target domain. Supplementing textual features with prosodic features (in our case pause durations between words) brought largest improvements in period restoration as expected and was more useful the weaker the textual component of the model was.

Comparison of unidirectional and bidirectional RNNs showed that bidirectional models outperform the unidirectional ones by a large margin (although the unidirectional model can still be useful when low latency is crucial), suggesting that knowing a larger than one word context ahead is an important factor in determining punctuation. The attention mechanism also turned out to be a useful component in our BRNN model, although to a much smaller extent than bidirectionality. Our analysis also showed that the way the attention was used by our model can be approximated with a simpler and faster architecture in future work. The late fusion we used in our model to integrate the attention component output into the rest of the model improved the overall recall of the model, but reduced the precision, so the total amount of errors remained similar.

3.4.1 Future work

While our models performed well, there are still some weaknesses to address.

First, our models have a limited vocabulary and all words out of that vocabulary are mapped to a shared out-of-vocabulary token. This might hide valuable information from the model and hurt punctuation restoration performance in some contexts. Although, the problem can be somewhat mitigated with the use of additional text based features like part-of-speech tags (Cho et al., 2015a), it makes the model dependent on the availability and quality of corresponding model. A potentially better way to mitigate this problem is adding a character-level submodule to the model that is trained jointly with the rest of the model (Rei et al., 2016). Another option is to train a character based model that can predict embeddings for out-of-vocabulary words separately (Pinter et al., 2017). The latter has the advantage that it can be easily used with preexisting models.

Secondly, in this work we restored a limited amount of punctuation types and the only prosodic feature we used was pause duration between words.

Future research may include a richer set of prosodic features and predict a larger variety of punctuation marks such as dashes, colons, semicolons and parentheses. Parentheses and other paired punctuation marks might create new challenges for the models.

Thirdly, errors in the ASR system output had a large negative impact on punctuation restoration performance for all models. Improved robustness to ASR errors would produce more accurate predictions on transcribed speech as has been shown for neural headline generation (Yu et al., 2016).

Finally, the English TED Talks dataset that was used is relatively small. It is likely that more impressive results can be obtained with the help of larger training sets.

One potential weakness of the BRNN model is the fact that the amount of context diminishes towards the edges of the sequence, potentially making punctuation prediction there harder and less accurate than in the middle of sequence. Adding additional context-only words on both sides of the sequence might alleviate the problem (or, in other words: predicting punctuation only for a subsequence in the middle).

Another problem in our models was that we were unable to utilize previous predictions of the model for subsequent predictions without performance degradation. The same problem was found by Xu et al. (2016a). It is worth exploring whether using scheduled sampling (Bengio et al., 2015) and similar techniques improves the robustness to previous prediction mistakes and makes the quality of restored punctuation better.

In addition to fixing the shortcomings of current work, there are several other opportunities for future research.

While in this work we mainly focused on restoring punctuation in speech transcripts, it would be interesting to explore other use cases like automatic punctuation suggestions in mobile keyboards or word processor software.

We showed that RNNs are better hidden event language models than n-gram models, so a natural course would be to apply RNNs to other hidden event language modeling problems, such as, reconstruction of compound words (Alumäe, 2007). Joint punctuation and capitalization models can also be explored (Gravano et al., 2009).

The input layer weights or word embeddings in neural language models have shown very interesting and useful properties (Mikolov et al., 2013). One future direction could be studying the embeddings of neural hidden event language models and find out whether they also have useful properties or can be complementary to regular embeddings.

Finally, our two-step training method used only the last recurrent layer state as the input to the second stage model. More information can be extracted by learning a linear combination of more layers of the first stage model (Peters et al., 2018).

Chapter 4

Low-resource headline generation with neural networks

This chapter is based on the work in publication V.

Research questions:

- Can we improve headline generation performance if we utilize the available text in the corpus to a more full extent and how much does pre-training help?
- Most commonly used pre-training method used is pre-trained word embeddings. What is the effect of pre-training other components of the model? Does it help to pre-train more parameters?

4.1 Background

Neural headline generation (NHG) is the process of automatically generating a headline based on the text of the document using artificial neural networks.

Headline generation is a subtask of text summarization. While a summary may cover multiple documents, generally uses similar style to the summarized document, and consists of multiple sentences, headline, in contrast, covers a single document, is often written in a different style (Headlinese (Mardh, 1980)), and is much shorter (frequently limited to a single sentence).

Due to shortness and specific style, condensing the the document into a headline often requires the ability to paraphrase which makes this task a good fit for abstractive summarization approaches where neural networks based attentive encoder-decoder (Bahdanau et al., 2015) type of models have recently shown impressive results. Starting with simpler FFNN LM decoder combined with convolutional and attention based decoders (Rush et al., 2015),

more recent models have switched to RNN based encoders (Lopyrev, 2015) and decoders (Chopra et al., 2016). Lopyrev (2015) additionally proposes a simplified attention mechanism that performs better and uses a small subset of hidden layer neurons to compute attention weights. Further work includes many extensions to the encoder-decoder architecture or training objective to improve the quality of generated headlines or summaries. Shen et al. (2016) and Paulus et al. (2017) develop methods for optimizing ROUGE directly and to overcome the disparity between training and generation in conventional maximum likelihood optimized encoder-decoder models. Shen et al. (2016) use sentence-wise optimization with minimum risk training that enables optimizing ROUGE directly. Paulus et al. (2017) combine conventional maximum likelihood objective with reinforcement learning based objective to optimize ROUGE and language modeling loss simultaneously. They also propose intra-attention mechanisms in both encoder and decoder to ensure that different parts of the document are used and to reduce repeated phrases in the generated summary. Repetition problem is also explored by Suzuki and Nagata (2017). Another direction of research deals with improving the handling of out-of-vocabulary and rare words during generation by giving the models the ability to extract words from the source document in addition to generating new words from the limited decoder vocabulary. Gulcehre et al. (2016) create an extension for neural machine translation and summarization models to solve the problem of generating out-of-vocabulary words. The method uses a special submodel to switch between generating words from a shortlist vocabulary or copying them from the input based on attention weights (or location softmax in case the model does not use the attention mechanism). A very similar method by Gu et al. (2016), that is not limited to copying out-of-vocabulary words, uses a shared normalization term between extractive and generative output layer instead of the switching component. See et al. (2017) propose another similar solution for multi-sentence abstractive summaries with a switching mechanism like Gulcehre et al. (2016), but not restricted to out-of-vocabulary words and the generative and extractive probabilities are mixed. They also propose simple method to improve the summarization coverage by summing past attention weights and feeding the resulting vector as additional input to the attention component and adding coverage loss to the primary objective. Summarization coverage problem is also explored by Chen et al. (2016) who abstractively summarize documents with up to thousands of words. They use several distraction mechanisms to encourage the model to cover more content of the document. Several means to control the output length in encoder-decoder models, including automatically generated summaries, is proposed by Kikuchi et al. (2016). Ma et al. (2017) improve the quality of abstractive summaries by encouraging semantic similarity between the document and summary by

adding an additional loss term to the training objective. The additional loss term is based on cosine distance between the semantic representations of the document and summary (based on last states of encoder and decoder RNNs respectively). Some work focused on providing the model with additional information about the source document by either using additional features on the encoder side or utilizing the general topic of the document. Xu et al. (2016b) experiment with guiding the headline generation process with information about the topic of the document, by assigning each document into a topic using latent Dirichlet allocation and adapting most of the model for each topic. Takase et al. (2016) explore incorporating additional syntactic and semantic features into a neural headline generation model by the means of *abstract meaning representation*. Nallapati et al. (2016) enrich embeddings on the encoder side with part-of-speech and named entity tags, and TF-IDF statistics to better detect key entities in the document. They additionally extend the base model with the *large vocabulary trick* (Jean et al., 2015) where each mini-batch has a custom vocabulary based mostly on source document words extended with their 1-nearest-neighbors in the word embedding space and a hierarchical encoder and attention that operates both at the word and sentence level. A specific problem in generating headlines for ASR output where word errors in the source document can degrade the quality of generated headlines is explored by Yu et al. (2016). They extend their model with an error estimation submodel that enables the model to pay less attention to erroneous words in the input.

While state-of-the art results have been obtained by training NHG models on large datasets like Gigaword, access to such resources is often not possible, especially when it comes to low-resource languages. In this work we focus on maximizing performance on smaller datasets with different pre-training methods.

One of the reasons to expect pre-training to be an effective way to improve performance on small datasets, is that NHG models are generally trained to generate headlines based on just a few first sentences of the documents (Rush et al., 2015; Shen et al., 2016; Chopra et al., 2016; Nallapati et al., 2016). This leaves the rest of the text unutilized, which can be alleviated by pre-training subsets of the model on full documents. Additionally, the decoder component of NHG models can be regarded as a LM whose predictions are biased by the external information from the encoder. As a LM it sees only headlines during training, which is a small fraction of text compared to the documents. Supplementing the training data of the decoder with documents via pre-training might enable it to learn more about words and language structure.

Although, some of the previous work has used pre-training before (Nallapati et al., 2016; Paulus et al., 2017; Gulcehre et al., 2016), it is not fully

explored how much pre-training helps and what is the optimal way to do it. Another problem is, that in previous work only a subset of parameters (usually just embeddings) is pre-trained leaving the rest of the parameters randomly initialized.

The main contributions of this work are: LM pre-training for fully initializing the encoder and decoder (sections 4.2.1 and 4.2.2); combining LM pre-training with distant supervision (Mintz et al., 2009) pre-training using filtered sentences of the documents as noisy targets (i.e. predicting one sentence given the rest) to maximally utilize the entire available dataset and pre-train all the parameters of the NHG model (section 4.2.3); and analysis of the effect of pre-training different components of the NHG model (section 4.3.3).

4.2 Methods

The model that we use follows the architecture described by Bahdanau et al. (2015). Although originally created for neural machine translation, this architecture has been successfully used for NHG (e.g., by Shen et al. (2016); Nallapati et al. (2016) and in a simplified form by Chopra et al. (2016)).

The NHG model consists of:

- A bidirectional (Schuster and Paliwal, 1997) encoder with GRUs (Cho et al., 2014).
- A unidirectional GRU decoder with a deep output layer (Pascanu et al., 2014) using maxout units (Goodfellow et al., 2013).
- An attention mechanism and a decoder initialization layer that connect the encoder and decoder (Bahdanau et al., 2015).

During headline generation, the encoder reads and encodes the words of the document. Initialized by the encoder, the decoder then starts generating the headline one word at a time, attending to relevant parts in the document using the attention mechanism (Figure 4.1). During training the parameters are optimized to maximize the probabilities of reference headlines.

While generally at the start of training either the parameters of all the components are randomly initialized or only pre-trained embeddings (with dashed outline in Figure 4.1) are used (Nallapati et al., 2016; Paulus et al., 2017; Gulcehre et al., 2016), we propose pre-training methods for more extensive initialization. The general idea is to train simpler submodels of complex models on tasks that match the goals of these subnetworks and use the parameters of the trained submodels to initialize the parameters of the complex model before fine-tuning on the final task.

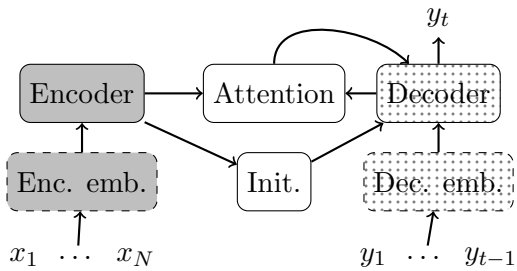


Figure 4.1: A high level description of the NHG model. The model predicts the next headline word y_t given the words in the document $x_1 \dots x_N$ and already generated headline words $y_1 \dots y_{t-1}$.

4.2.1 Encoder Pre-Training

When training a NHG model, most approaches generally use a limited number of first sentences or tokens of the document. For example Rush et al. (2015); Shen et al. (2016); Chopra et al. (2016) use only the first sentence of the document and Nallapati et al. (2016) use up to 2 first sentences. While efficient (training is faster and takes less memory as the input sequences are shorter) and effective (the most informative content tends to be at the beginning of the document (Nallapati et al., 2016)), this leaves the rest of the sentences in the document unused. Better understanding of words and their context can be learned if all sentences are used, especially on small training sets.

To utilize the entire training set, we pre-train the encoder on all the sentences of the training set documents. Since the encoder consists of two recurrent components – a forward and backward GRU – we pre-train them separately. First we add a softmax output layer to the forward GRU and train it on the sentences to predict the next word given the previous ones (i.e. we train it as a LM). After convergence on the validation set sentences, we take the embedding weights of the forward GRU and use them as fixed parameters for the backward GRU. Then we train the backwards GRU following the same procedure as with the forward GRU, with the exception of processing the sentences in a reverse order. When both models are fully trained, we remove the softmax output layers and initialize the encoder of the NHG model with the embeddings and GRU parameters of the trained LMs (highlighted with gray background in Figure 4.1).

A very similar encoder pre-training has also been proposed by Ramachandran et al. (2017) for neural machine translation and summarization. Besides the difference in the task, there are several other differences between their approach and ours. First, they leave all recurrent layers except for the

lowest one randomly initialized while we aim to pre-train all the parameters. Secondly, their method is designed for uni-directional encoders while we propose a method for bi-directional encoders. Finally, they propose using an additional language modeling loss during final training or fine tuning while our method leaves the loss function unchanged throughout all stages of training.

4.2.2 Decoder Pre-Training

Pre-training the decoder as a LM seems natural, since it is essentially a conditional LM. During NHG model training the decoder is fed only headline words, which is relatively little data compared to the document contents. To improve the quality of the headlines it is essential to have high quality embeddings that are a good semantic representation of the input words and to have a well trained recurrent and output layer to predict sensible words that make up coherent sentences. When it comes to statistical models, the simplest way to improve the quality of the parameters is to train the model on more data, but it also has to be the right kind of data (Moore and Lewis, 2010).

To increase the amount of suitable training data for the decoder we use LM pre-training on filtered sentences of the training set documents. For filtering we use the XenC tool by Rousseau (2013) with the cross-entropy difference filtering (Moore and Lewis, 2010). In our case the in-domain data is training set headlines, out-domain data is the sentences from training set documents, and the best cut-off point is evaluated on validation set headlines. The careful selection of sentences is mostly motivated by preventing the pre-trained decoder from deviating too much from Headlinese, but it also reduces training time.

Before pre-training we initialize the input and output embeddings of the LM for words that are common in both encoder and decoder vocabulary with the corresponding pre-trained encoder embeddings. We train the LM on the selected sentences until perplexity on the validation set headlines stops improving and then use it to initialize the decoder parameters of the NHG model (highlighted with dotted background in Figure 4.1).

A similar approach, without data selection and embedding initialization, has also been proposed by Ramachandran et al. (2017) and as possible future research by Alifimoff (2015). The method by Ramachandran et al. (2017) additionally differs from ours in the extent of pre-trained parameters — it leaves all recurrent layers except for the lowest one randomly initialized, so the problem of randomized inputs to the pre-trained output layer is alleviated with a residual connection (He et al., 2016) from the first recurrent layer to the output layer. Our method does not leave any parameters randomly initialized. Another difference is in the loss function during fine-tuning

— Ramachandran et al. (2017) add a language modeling loss to the main objective.

4.2.3 Distant Supervision Pre-Training

Approaches described in sections 4.2.1 and 4.2.2 enable full pre-training of the encoder and decoder, but this still leaves the connecting parameters (with white background in Figure 4.1) untrained.

As results in language modeling suggest, surrounding sentences contain useful information to predict words in the current sentence (Wang and Cho, 2016). This implies that other sentences contain informative sections that the attention mechanism can learn to attend to and general context that the initialization component can learn to extract.

To utilize this phenomenon, we propose using carefully picked sentences from the documents as pseudo-headlines and pre-train the NHG model to generate these given the rest of sentences in the document. Our pseudo-headline picking strategy consists of choosing sentences that occur within 100 first tokens of the document and were retained during cross-entropy filtering in section 4.2.2. Picking sentences from the beginning of the document should give us the most informative sentences, and cross-entropy filtering keeps sentences that most closely resemble headlines.

The pre-training procedure starts with initializing the encoder and decoder with LM pre-trained parameters (sections 4.2.1 and 4.2.2). After that, we continue training the attention and initialization parameters until perplexity on validation set headlines converges. We then use the trained parameters to initialize all parameters of the NHG model.

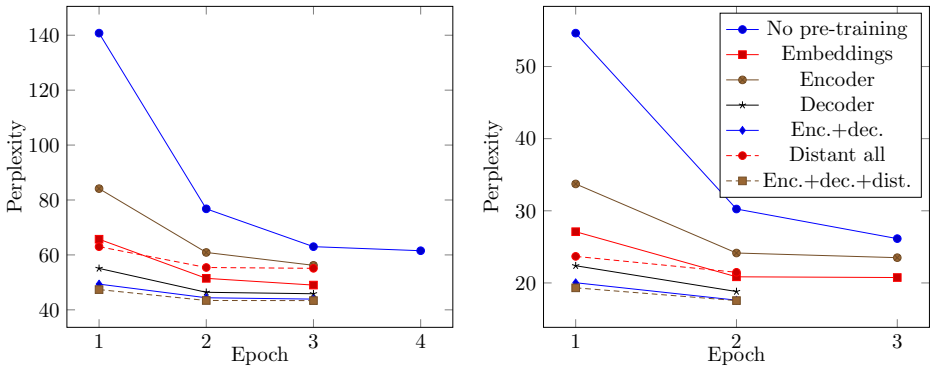
Distant supervision has been also used for multi-document summarization by Bravo-Marquez and Manriquez (2012).

4.3 Experiments

We evaluate the proposed pre-training methods in terms of ROUGE (Lin, 2004) and perplexity on two relatively small datasets (English and Estonian).

4.3.1 Training Details

All our models use hidden layer sizes of 256 and the weights are initialized according to Glorot and Bengio (2010). Although we use single recurrent layer models in our experiments, we see no reason why the applied pre-training methods should not generalize to multi-layer recurrent models, by training multi-layer language models during pre-training. The maxout (Goodfellow et al., 2013) hidden layer in the deep output layer (Pascanu et al., 2014) of the decoder takes a maximum across $k = 2$ inputs. The vocabularies consist of up to 50000 most frequent training set words that occur at least 3 times (the Estonian vocabulary includes two compound word



(a) English (b) Estonian

Figure 4.2: (Color online) Validation set perplexities of the NHG model with different pre-training methods.

concatenation symbols). The model is implemented in Theano (Bergstra et al., 2010; Bastien et al., 2012) and trained on GPUs using mini-batches of size 128. During training the weights are updated with Adam (Kingma and Ba, 2015) (parameters: α=0.001, β_1=0.9, β_2=0.999, ϵ=10^{-8} and λ=$1-10^{-8}$) and L_2-norm of the gradient is kept within a threshold of 5.0 (Pascanu et al., 2013). During headline generation we use beam search with beam size 5. For the Estonian model we postprocess the output by restoring compound words according to generated compound word concatenation symbols.

4.3.2 Datasets

We use the CNN/Daily Mail dataset (Hermann et al., 2015)[1] for experiments on English (EN). The number of headline-document pairs is 287227, 13368 and 11490 in training, validation and test set correspondingly. To the best of our knowledge, this is the first published work using the CNN/Daily Mail dataset for headline generation experiments. The preprocessing consists of tokenization, lowercasing, replacing numeric characters with #, and removing irrelevant parts (editor notes, timestamps etc.) from the beginning of the document with heuristic rules.

For Estonian (ET) experiments we use a similarly sized (341607, 18979 and 18977 training, validation and test split) dataset that also consist of news from two sources. During preprocessing, compound words are split, words are truecased and numbers are written out as words. We used Estnltk (Orasmaa et al., 2016) stemmer for ROUGE evaluations.

Both datasets can be considered small compared to commonly used

[1] `http://cs.nyu.edu/~kcho/DMQA/`

Table 4.1: Proportion of pre-trained parameters, amount of text from documents utilized and distance of closest pre-trained parameters from the output in the pre-training experiments.

	% pre-trained		% text		Distance
Model	EN	ET	EN	ET	EN & ET
No pre-training	0	0	12.2	40.9	-
Embeddings	64.6	62.4	100	100	2
Encoder	38.9	33.1	100	100	2
Decoder	57.9	64.1	27.4	50.1	0
Enc.+dec.	96.8	97.3	100	100	0
Distant all	100	100	5.5	24.1	0
Enc.+dec.+dist.	100	100	100	100	0

Gigaword dataset (Graff et al., 2003) which consists of 9.5M document-headline pairs out of which 4 million are generally used due to filtering.

4.3.3 Results and Analysis

Models are evaluated in terms of PPL and full length ROUGE (Lin, 2004). [2] In addition to pre-training methods described in sections 4.2.1-4.2.3, we also test: initializing only the embeddings using parameters from the LM pre-trained encoder and decoder (*Embeddings*); initializing the encoder and decoder, but leaving connecting parameters randomized (*Enc.+dec.*); pre-training the whole model from random initialization with distant supervision only (*Distant all*); and a baseline that is not pre-trained at all (*No pre-training*).

The statistics of all pre-training experiments are shown in Table 4.1. The table shows:

- The amount of parameters that are initialized with pre-trained values before training for the final task.
- The amount of words from the training set documents that are used for pre-training (*No pre-training* row shows the amount of text used during final training). 100% corresponds to full documents that have passed through a filtering of small amount of sentences in a pre-processing stage.
- The number of weight matrices between the output and the pre-trained parameters closest to it.

[2] We use ROUGE package with options: *-n 2 -m* for EN and *-n 2* for ET

Table 4.2: Perplexities on the test set with a 95% confidence interval (Klakow and Peters, 2002). All pre-trained models are significantly better than the *No pre-training* baseline.

Model	PPL (EN)	PPL (ET)
No pre-training	65.1 ±1.0	25.9 ±0.4
Embeddings	51.8 ±0.7	20.7 ±0.3
Encoder 4.2.1	59.3 ±0.9	23.5 ±0.4
Decoder 4.2.2	48.3 ±0.7	18.8 ±0.3
Enc.+dec.	46.2 ±0.7	17.7 ±0.3
Distant all	58.6 ±0.9	21.3 ±0.3
Enc.+dec.+dist. 4.2.3	**45.8 ±0.7**	**17.5 ±0.3**

All pre-training methods gave significant improvements in PPL (Table 4.2). The best method (*Enc.+dec.+dist.*) improved the test set PPL by 29.6-32.4% relative. Pre-trained NHG models also converged faster during training (Figure 4.2) and most of them beat the final PPL of the baseline already after the first epoch. General trend is that pre-training a larger amount of parameters and the parameters closer to the outputs of the NHG model improves the PPL more. *Distant all* is an exception to that observation as it used much less pre-training data (less than the *No pre-training* baseline during training) than other methods. Utilizing more text from the documents during pre-training does not show a very strong effect for PPL.

For ROUGE evaluations, we report ROUGE-1 and ROUGE-L (Tables 4.3a and 4.3b). In contrast with PPL evaluations, some pre-training methods either don't improve significantly or even worsen ROUGE measures. Another difference compared to PPL evaluations is that for ROUGE, pre-training parameters that reside further from outputs (embeddings and encoder) seems more beneficial. This might imply that a better document representation is more important to stay on topic during beam search while it is less important during PPL evaluation where predicting next target headline word with high confidence is rewarded and the process is aided by previous target headline words that are fed to the decoder as inputs. It is also possible, that a well trained decoder becomes too reliant on expecting correct words as inputs making it sensitive to errors during generation which would somewhat explain why *Enc.+dec.* performs worse than *Encoder* alone. This hypothesis can be checked in further work by experimenting with methods like scheduled sampling (Bengio et al., 2015) that should increase the robustness to mistakes during generation or by forcing the dynamics of the model to be more similar during training and sampling with Professor

Table 4.3: Recall, precision and F-score of ROUGE-1 and ROUGE-L on the English CNN/Daily Mail test sets. Best scores in bold. Results with statistically significant differences (95% confidence) compared to *No pre-training* underlined.

(a) English CNN/Daily Mail

Model	$R1_R$	$R1_P$	$R1_F$	RL_R	RL_P	RL_F
No pre-training	20.36	33.51	24.04	17.68	29.03	20.84
Embeddings	21.09	33.36	24.62	18.23	28.72	21.2
Encoder	21.25	34.1	24.97	18.45	29.5	21.61
Decoder	20.11	31.1	23.23	17.43	26.87	20.08
Enc.+Dec.	20.72	33.93	24.47	18.04	29.43	21.24
Distant all	20.32	31.54	23.49	17.59	27.25	20.29
Enc.+dec.+dist.	**21.34**	**34.81**	**25.19**	**18.53**	**30.14**	**21.82**

(b) Estonian news

Model	$R1_R$	$R1_P$	$R1_F$	RL_R	RL_P	RL_F
No pre-training	26.44	34.23	29.05	25.31	32.74	27.79
Embeddings	28.42	35.94	30.92	27.02	34.16	29.38
Encoder	**29.28**	**37.04**	**31.88**	**27.88**	**35.24**	**30.35**
Decoder	25.12	32.6	27.62	23.89	30.99	26.26
Enc.+dec.	27.18	34.58	29.64	25.79	32.78	28.11
Distant all	26.17	34.49	28.99	24.96	32.87	27.63
Enc.+dec.+dist.	27.74	35.46	30.32	26.35	33.67	28.79

Table 4.4: Successful examples of generated headlines on the CNN/Daily Mail dataset.

Document	a democratic congressman is at the head of a group of representatives trying to help undocumented immigrants avoid deportations with what they have called the family defender toolkit . the informational pamphlet includes a bilingual card - that some are calling a get out of deportation free card - that lists reasons a person should not be deported under expanded .
Reference	congressman is developing a get out of deportation toolkit to help undocumented immigrants if they are detained
No pre-training	congressman calls for undocumented immigrants
Embeddings	congressman calls for help from immigrants trying to help immigrants avoiding deportation
Encoder	republican congressman calls for immigrants trying to avoid deportation
Decoder	congressman who tried to stop deportations of immigrants
Enc.+Dec.	immigration congressman at the head of the head of the group who tries to avoid deportation
Distant all	congressman calls for deportation to immigrants who stay in the country
Enc.+dec.+dist.	congressman tries to help undocumented immigrants avoid deportation

Forcing (Goyal et al., 2016). Another option is to directly optimize ROUGE with *minimum risk training* (Shen et al., 2016) or reinforcement learning (Paulus et al., 2017). Unlike PPL, ROUGE tends to improve with fully utilizing the text from the documents during pre-training. Pre-training all parameters on all available text (*Enc.+dec.+dist.*) gives the best result on English and quite decent results on Estonian. Best models improve ROUGE by 0.85-2.84 points.

4.3.4 Examples

Some examples of the generated headlines on the CNN/Daily Mail dataset are shown in Tables 4.4 and 4.5.

Table 4.4 shows an example where the reference headline is an abstractive summary of the document. All pre-trained models also produce abstractive

Table 4.5: Unsuccessful examples of generated headlines on the CNN/Daily Mail dataset.

Document	a chihuahua and a bearded dragon showed off their interspecies friendship when they embarked upon a game of tag together . videoed in their front room , the dog named foxxy cleopatra and the reptile called ryuu can be seen chasing after one another around a coffee table . standing perfectly still while looking in the other direction , the bearded dragon initially appears disinterested as the chihuahua jumps around excitedly .
Reference	you re it!
No pre-training	is this the creepiest crawly?
Embeddings	meet the poodle!
Encoder	it's a knockout!
Decoder	the bearded dragon lizard: the bearded dragon lizard spotted in the middle of the street
Enc.+Dec.	oh, this is a lion!
Distant all	meet the dragon dragon: meet the dragon dragon
Enc.+dec.+dist.	is this the world's youngest lion?

summaries as headlines with varying quality. The model pre-trained to the largest extent (*Enc.+dec.+dist*) produces the best headline that is grammatically and factually correct, almost extractively summarizes the document, and is in Headlinese. Other models produce headlines that also at least get the topic right and include the relevant keywords, but fail to produce coherent sentences (e.g., *Enc.+Dec.*) or introduce errors that make the headline factually incorrect (e.g., *Distant all*).

Table 4.5 illustrates an example with a more creative reference headline. The reference headline is very short and consists entirely of stop words. This kind of headlines are often removed from the training and test sets. For example, Rush et al. (2015) remove all headline-document pairs where the headline has no non-stop-words in common with the document or contains a byline, colon or question mark. We do not filter any headline-document pairs as we expect our model to be able to generate all types of headlines. From the examples in Table 4.5 it is evident that most models are able to recognize based on the contents of the document that the reference headline is creative and also try to generate creative headlines. Unfortunately almost all generated headlines are either grammatically or factually incorrect. The

No pre-training model generates the only headline that can be regarded as correct (or not wrong) and as good as the reference headline. The *Decoder* and *Distant all* generate headlines that contain relevant keywords, but repeat themselves or get the facts wrong. This example illustrates a case where ROUGE and other automatic measures do not provide a good headline quality estimate and human evaluation is more appropriate.

4.4 Conclusions

We proposed three new NHG model pre-training methods that in combination enable utilizing the entire dataset and initializing all parameters of the NHG model. We also evaluated and analyzed pre-training methods and their combinations in terms of PPL and ROUGE on two languages.

The results revealed that better PPL does not necessarily translate to better ROUGE – PPL tends to benefit from pre-training parameters that are closer to outputs, but for ROUGE it is generally the opposite. Also, PPL benefited from pre-training more parameters while for ROUGE it was not always the case. Utilizing more text from the documents during pre-training generated larger gains in ROUGE than in PPL. Pre-training in general proved to be useful – our best results improved PPL by 29.6-32.4% relative and ROUGE F-scores by 0.98-2.83 points compared to a NHG model without pre-training.

These findings are similar to the ones from Ramachandran et al. (2017) for neural summarization where partial encoder pre-training turned out to be more effective than decoder pre-training and the achieved improvement with pre-training is even larger than ours (over 4.5 ROUGE points). A method similar to the distant supervision pre-training (without the data selection) was also shown to be helpful in a simultaneously published work on abstractive summarization by Hua and Wang (2017).

One problem with the proposed decoder pre-training method is the fact that it optimizes the maximum-likelihood objective which can make the model too sensitive to mistakes during generation as it was not exposed to them during pre-training and training. This explains why all pre-training methods reliably improved PPL but the decoder pre-training degraded the ROUGE score. Better training techniques such as scheduled sampling (Bengio et al., 2015) during maximum-likelihood optimization or better training objectives like *minimum risk training* (Shen et al., 2016), Professor Forcing (Goyal et al., 2016) or reinforcement learning (Paulus et al., 2017) can be explored for better decoder pre-training and final model training.

Our pre-training method for the encoder was based on training it as a language models. Previous work on LSTM RNN pre-training methods for sequence classification tasks has shown that in some cases initializing the embeddings and recurrent layers with autoencoder pre-training (encoding

the sequence into a vector with a RNN and decoding the same sequence with the same RNN from that vector) can be more effective than language model pre-training (Dai and Le, 2015). Autoencoder pre-training has also been effective in LSTM RNN sentence compression models (Sakti et al., 2015).

Current work focused on maximally utilizing available headlined corpora. One interesting future direction would be to additionally utilize potentially much more abundant corpora of documents without headlines for pre-training. Using large unlabeled corpora was also proposed by Shen et al. (2016) and experimented with by Ramachandran et al. (2017) for neural summarization. Although Ramachandran et al. (2017) showed that in summarization the gap between pre-training on the large unlabeled corpus and smaller labeled corpus is almost nonexistent, they only partially pre-trained the encoder and the decoder and the attention component was not pre-trained at all. Therefore the question whether using a larger unlabeled corpus for attention component distant supervision pre-training and full encoder-decoder pre-training improves the quality of generated headlines is still open. Another open question is the relationship between the dataset size and the effect of pre-training for neural headline generation. Ramachandran et al. (2017) showed that pre-trained neural machine translation models degrade less with reduced amount of labeled data.

In this work we used a data selection method for generating pseudo-headlines for distant pre-training. An alternative is to use extractive summarization (e.g. Lin et al., 2009; Kågebäck et al., 2014). The data selection method picked sentences that were a better match for Headlinese, but did not necessarily summarize the document. Although not all the headlines in the corpora were summarizing, the ability to summarize the content is probably much harder to learn than the ability to generate text in Headlinese. Abstractive summarization pseudo-headlines could potentially enable the model to learn to generate better headlines that summarize the document without much paraphrasing. The data selection and abstractive summarization method are not mutually exclusive and could be combined to use the benefits of both.

Since we did not exclude any headline-document pairs there were many headlines that were very abstractive, paraphrased a lot and sometimes did not have any words in common with the document. A human evaluation of the generated headlines could give a more accurate estimate of the effectiveness of the methods. An automatic metric that is less biased towards lexical similarity than ROUGE (e.g. Ng and Abrecht, 2015) can be considered as a simpler and faster alternative.

There are many recent developments in the field of neural summarization and headline generation that try to solve the problems with the quality of the generated output. For example, there are extensions that try to reduce

the number of factual errors and repetitions (See et al., 2017; Suzuki and Nagata, 2017) and give the model the ability to output out-of-vocabulary words (Nallapati et al., 2016; See et al., 2017; Gulcehre et al., 2016). Using and pre-training these extensions could help improve low-resource neural headline generation even further.

Chapter 5

Conclusions

This dissertation presented neural networks based methods and models for low-resource language modeling problems. The presented work can improve the quality of many systems that depend on language models, but need to function on low-resource languages or domains.

In Chapter 2 we presented two methods to improve language modeling on small domains or languages. The first method was an adaptation of an existing FFNN LM multi-domain component (Alumäe, 2013) for RNN LMs. The original multi-domain component turned out to be incompatible with the sequential processing in RNNs, but after analysis we were able to come up with a simple modification and obtain a 16% relative improvement in PPL on a medical domain corpus. The multi-domain component allowed us to train a joint model for all 10 domains instead of separate models. The second method allowed using the multi-domain component for adapting a neural network LM to a very small target domain, utilizing the same subset of data for both validation and adaptation. The adaptation method gave consistent improvements in PPL (2.2–3.3% relative), although it made no significant difference in WER.

Chapter 3 presented two RNN based punctuation restoration models that can be viewed as replacements for the n-gram based hidden event LMs. Both the unidirectional LSTM RNN and the more advanced bidirectional RNN with an attention mechanism gave a large improvement over the n-gram baseline in our experiments on an Estonian corpus. Experiments on an English corpus showed that the bidirectional model is also a huge improvement over the previous state of the art models based on deep and convolutional neural networks. To utilize prosodic features in addition to the textual features, the chapter presented a two-stage training method that replaces the output layer of a trained textual model with a new RNN which is trained on a much smaller prosody annotated dataset to utilize prosodic

features in conjunction with textual features computed by the first stage model. An additional benefit of two-stage training was the adaptation to target domain. The two-stage method significantly improved the performance of both models using interword pause duration feature alone, most noticeably in detecting sentence boundaries.

Chapter 4 proposed three pre-training methods for neural headline generation models to improve the performance on small datasets. The methods were each designed to pre-train a different component of the final model. Two methods were based on pre-training the encoder and decoder component as language models, while the third method used distant-supervision to pre-train the attention mechanism and other parameters connecting the encoder and decoder. When combined, the proposed methods enabled full utilization of the training data and useful initialization of all parameters of the model. Experiments on relatively low-resource Estonian and English datasets showed that all pre-training methods, either separately or in conjunction, significantly improved PPL. On the other hand, some methods and their combinations, that pre-trained the parameters closest to the output layer, degraded the performance in terms of ROUGE. However, the best results by a significant margin were still obtained with pre-trained models — a combination of all methods performed best on English and the encoder pre-training method on Estonian.

In the following sections we will determine the validity of the claims from the introduction and discuss some potential future research directions.

5.1 Validation of claims

Claim 1 proposed that when the dataset consists of texts from multiple small domains the performance of RNN LMs can be improved by using a multi-domain component (Alumäe, 2013) that was initially proposed for FFNN LMs. When it comes to the performance in terms of PPL the Claim 1 is clearly valid as can be seen from the experimental results in Chapter 2. However, to assess the practical usefulness of the multi-domain component in RNN LMs, integration into an ASR system and evaluation in terms of WER is required.

Claim 2 asserted that the multi-domain component is also a good fit for adapting neural network LMs to small domains. Chapter 2 supported this claim with consistent PPL improvements and showed that the small amount of domain-specific parameters is resistant to overfitting during adaptation even when we utilize the validation data for adaptation. On the other hand, the PPL improvements were small and did not translate into a reduction in WER. This does not automatically entail that the method would not be practical on other datasets or for other tasks, but further experimentation is needed. Also, a comparison with other adaptation methods instead of just

an unadapted baseline would help to better asses the effectiveness of the method.

Claim 3 stated that RNNs are an improvement over conventional hidden event LMs and can be used for training a joint model on both text and prosody annotated data that potentially have very different sizes. Chapter 3 presented two RNN text based models along with a two-stage training method to utilize prosody annotated data. Punctuation restoration experiments on Estonian demonstrated a large improvement over conventional hidden event LMs and both RNN models gained a significant additional improvement with the two-stage training method. These results provide a strong support for Claim 3.

Claim 4 proposed that pre-training all parameters of neural headline generation models improves the quality of generated headlines on small datasets and facilitates full utilization of training data. The combination of three pre-training methods proposed in Chapter 4 enabled pre-training the entire neural headline generation model, utilized all available training data and gave significant improvements in both PPL and ROUGE on two languages. However, the best results in terms of ROUGE on Estonian were achieved with the encoder pre-training alone, which initialized only 1/3 of the parameters, but still utilized all of the training data. Although this indicates that sometimes pre-training the entire model is not the best choice, it does not invalidate the Claim 4 as the fully pre-trained model gave the best results on English and significant improvements on Estonian.

5.2 Further work

The presented work leaves several potential directions for further research in addition to the ones stated in Section 5.1.

The multi-domain RNN LM proposed in Chapter 2 used a simple recurrent layer with logistic sigmoid units. Simple RNNs have been shown to have difficulty learning long-range dependencies (Bengio et al., 1994) even with *tanh* units, so logistic sigmoid units should be even less effective in that aspect. This brings up a question whether using more advanced units that are better at long-range dependencies like LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) would eliminate the need for a multi-domain component as the model may infer the domain from the long context. Another option, if the multi-domain component turns out to be useful in advanced RNN LMs, is do design a component that can be trained in a supervised manner, but can detect the domain automatically during test time (Shi et al., 2014) unlike our current version.

The multi-domain architecture based low-resource adaptation method can be further tested out on unsupervised or online adaptation (Souvignier and Kellner, 1998) where its resistance to overfitting can be useful.

The two RNN punctuation restoration models and the two-stage training method presented in Chapter 3 have room for improvements that can be addressed in further work. First and most important problem was the sensitivity to word errors in the ASR output. Training more robust models would greatly improve the practical value of the models for ASR. Using a larger variety of prosodic features or training the model to pay less attention to incorrectly recognized words (e.g. Yu et al., 2016) may reduce the reliance on the correctness of text. Another shortcoming was that the models had a limited input vocabulary and all out-of-vocabulary words were mapped to a shared token. One way to reduce the loss of information is to augment the models with a character based submodel (Rei et al., 2016). Next open problem is utilization of previous punctuation decisions. The proposed models do not use previous decisions as our preliminary experiments showed that wrong past decisions cause more wrong decisions in the future. Perhaps scheduled sampling (Bengio et al., 2015) and similar methods can reduce the accumulation of errors and make past predictions useful for the models. Finally, there are opportunities for additional experiments like restoring a larger variety of punctuation marks, analyzing the resulting word embeddings and applying the models to other hidden event language modeling problems.

Finally, the analysis of the pre-training methods for neural headline generation models in Chapter 4 revealed some areas for further improvement. First, the decoder pre-training method improved PPL but degraded the ROUGE scores. We believe that this happens due to the fact that during pre-training and training the next word is predicted based on correct word history while during generation at test time the decoder has to deal with its own previous outputs which may contain mistakes. Thus pre-training may make the model even more reliant on the correctness of word history. This problem has been recognized by others and several solutions such as scheduled sampling (Bengio et al., 2015), *minimum risk training* (Shen et al., 2016), Professor Forcing (Goyal et al., 2016) and reinforcement learning (Paulus et al., 2017) have been proposed. These methods may be a key to making decoder pre-training more effective. Next, the simple language modeling based pre-training of the encoder turned out to be consistently effective, but it has been shown that in some cases autoencoder based pre-training may produce better results (Dai and Le, 2015). Autoencoder pre-training would also eliminate the need to pre-train the bidirectional encoder as two separate language models. The distant-supervision based pre-training method for connecting components can also be improved. For a start, much more data from unlabeled sources can be utilized and different pseudo-headline selection methods (e.g., extractive summarization) can be experimented with. Also, like the decoder pre-training, the distant-supervision method can benefit from better training techniques or objectives. To get a more

accurate assessment of the effectiveness of pre-training a human evaluation is required. Reference headlines sometimes have no overlapping words with content and there are many correct ways to write headlines which makes the automatic scoring approximate.

References

Alifimoff, A. (2015). Abstractive sentence summarization with attentive deep recurrent neural networks. https://cs224d.stanford.edu/reports/aja2015.pdf.

Alumäe, T. (2007). Automatic compound word reconstruction for speech recognition of compounding languages. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 5–12. University of Tartu, Estonia, http://www.aclweb.org/anthology/W07-2403.

Alumäe, T. (2013). Multi-domain neural network language model. In *Interspeech 2013*, pages 2182–2186.

Alumäe, T. (2014). Recent improvements in Estonian LVCSR. In *Spoken Language Technologies for Under-Resourced Languages (SLTU 2014)*, Saint Petersburg, Russia.

Alumäe, T. and Kurimo, M. (2010). Domain adaptation of maximum entropy language models. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 301–306. Association for Computational Linguistics, http://www.aclweb.org/anthology/P10-2056.

Alumäe, T. and Tilk, O. (2016). Automatic speech recognition system for lithuanian broadcast audio. In *Human Language Technologies – The Baltic Perspective*, volume 289, pages 39–45. DOI: 10.3233/978-1-61499-701-6-39.

Bacchiani, M. and Roark, B. (2003). Unsupervised language model adaptation. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 1, pages I–224–I–227 vol.1. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2003.1198758.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR'2015, arXiv:1409.0473*.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.

Batista, F., Caseiro, D., Mamede, N., and Trancoso, I. (2007). Recovering punctuation marks for automatic speech recognition. In *Interspeech 2007*, Antwerp, Belgium.

Batista, F., Caseiro, D., Mamede, N., and Trancoso, I. (2008). Recovering capitalization and punctuation marks for automatic speech recognition: Case study for portuguese broadcast news. *Speech Communication*, 50(10):847–862.

Batista, F., Moniz, H., Trancoso, I., and Mamede, N. (2012). Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):474–485, ISSN: 1558-7916, DOI: 10.1109/TASL.2011.2159594.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press, `http://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf`.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, ISSN: 1045-9227, DOI: 10.1109/72.279181.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Bravo-Marquez, F. and Manriquez, M. (2012). A zipf-like distant supervision approach for multi-document summarization using wikinews articles. In Calderón-Benavides, L., González-Caro, C., Chávez, E., and Ziviani, N., editors, *String Processing and Information Retrieval*, pages 143–154, Berlin, Heidelberg. Springer Berlin Heidelberg, ISBN: 978-3-642-34109-0.

Broman, S. and Kurimo, M. (2005). Methods for combining language models in speech recognition. In *Ninth European Conference on Speech Communication and Technology*.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, ISSN: 0891-2017, `http://dl.acm.org/citation.cfm?id=176313.176316`.

Che, X., Wang, C., Yang, H., and Meinel, C. (2016). Punctuation prediction for unsegmented transcript based on word vector. In *The 10th International Conference on Language Resources and Evaluation (LREC)*.

Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016). Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2754–2760. AAAI Press, ISBN: 978-1-57735-770-4, `http://dl.acm.org/citation.cfm?id=3060832.3061006`.

Chen, X., Wang, Y., Liu, X., Gales, M. J., and Woodland, P. C. (2014). Efficient gpu-based training of recurrent neural network language models using spliced sentence bunch. In *Interspeech 2014*.

Cho, E., Kilgour, K., Niehues, J., and Waibel, A. (2015a). Combination of NN and CRF models for joint detection of punctuation and disfluencies. In *Interspeech 2015*.

Cho, E., Niehues, J., Kilgour, K., and Waibel, A. (2015b). Punctuation insertion for real-time spoken language translation. *Proceedings of the Eleventh International Workshop on Spoken Language Translation*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, DOI: 10.3115/v1/D14-1179.

Chopra, S., Auli, M., and Rush, M. A. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics, DOI: 10.18653/v1/N16-1012.

Christensen, H., Gotoh, Y., and Renals, S. (2001). Punctuation annotation using statistical prosody models. In *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc., `http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf`.

Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, 43(5):1470–1480.

Daume III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics, `http://www.aclweb.org/anthology/P07-1033`.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Elman, J. L. (1989). Representation and structure in connectionist models. Technical report, CALIFORNIA UNIV SAN DIEGO LA JOLLA CENTER FOR RESEARCH IN LANGUAGE.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Gandhe, A., Metze, F., and Lane, I. (2014). Neural network language models for low resource languages. In *Interspeech 2014*.

Gers, F. A., Schmidhuber, J. A., and Cummins, F. A. (2000). Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, ISSN: 0899-7667, DOI: 10.1162/089976600300015015.

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2003). Learning precise timing with LSTM recurrent networks. *J. Mach. Learn. Res.*, 3:115–143, ISSN: 1532-4435, DOI: 10.1162/153244303768966139.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR, `http://proceedings.mlr.press/v9/glorot10a.html`.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR, `http://proceedings.mlr.press/v15/glorot11a.html`.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, DOI: 10.1093/biomet/40.3-4.237.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1319–III–1327. JMLR.org, `http://dl.acm.org/citation.cfm?id=3042817.3043084`.

Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In Lee, D. D., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 4601–4609. Curran Associates, Inc.

Graff, D., Kong, J., Chen, K., and Maeda, K. (2003). English gigaword. *Linguistic Data Consortium, Philadelphia*, 4:1.

Gravano, A., Jansche, M., and Bacchiani, M. (2009). Restoring punctuation and capitalization in transcribed speech. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4741–4744. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2009.4960690.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics, DOI: 10.18653/v1/P16-1154.

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., and Bengio, Y. (2016). Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149. Association for Computational Linguistics, DOI: 10.18653/v1/P16-1014.

Hasan, M., Doddipatla, R., and Hain, T. (2014). Multi-pass sentence-end detection of lecture speech. In *Interspeech 2014*, pages 2902–2906.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. DOI: 10.1109/CVPR.2016.90.

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press, `http://dl.acm.org/citation.cfm?id=2969239.2969428`.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780, DOI: 10.1162/neco.1997.9.8.1735.

Hua, X. and Wang, L. (2017). A pilot study of domain adaptation effect for neural abstractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 100–106. Association for Computational Linguistics, `http://aclweb.org/anthology/W17-4513`.

Huang, J. and Zweig, G. (2002). Maximum entropy model for punctuation annotation from speech. In *Proceedings of the International Conference on Spoken Language Processing*, pages 917–920, Denver, CO, USA.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10. Association for Computational Linguistics, DOI: 10.3115/v1/P15-1001.

Kågebäck, M., Mogren, O., Tahmasebi, N., and Dubhashi, D. (2014). Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39. Association for Computational Linguistics, DOI: 10.3115/v1/W14-1504.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, ISSN: 0096-3518, DOI: 10.1109/TASSP.1987.1165125.

Khomitsevich, O., Chistikov, P., Krivosheeva, T., Epimakhova, N., and Chernykh, I. (2015). Combining prosodic and lexical classifiers for two-pass punctuation detection in a russian asr system. In Ronzhin, A., Potapova, R., and Fakotakis, N., editors, *Speech and Computer*, pages 161–169, Cham. Springer International Publishing, ISBN: 978-3-319-23132-7.

Kikuchi, Y., Neubig, G., Sasano, R., Takamura, H., and Okumura, M. (2016). Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338. Association for Computational Linguistics, DOI: 10.18653/v1/D16-1140.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Klakow, D. (1998). Log-linear interpolation of language models. In *ICSLP*, volume 5, pages 1695–1698. Australian Speech Science and Technology Association.

Klakow, D. and Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Commun.*, 38(1):19–28, ISSN: 0167-6393, DOI: 10.1016/S0167-6393(01)00041-3.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1. ISSN: 1520-6149, DOI: 10.1109/ICASSP.1995.479394.

Kolár, J. and Lamel, L. (2012). Development and evaluation of automatic punctuation for French and English speech-to-text. In *Interspeech 2012*, Portland, OR, USA.

Kolář, J., Shriberg, E., and Liu, Y. (2006). Using prosody for automatic sentence segmentation of multi-party meetings. In Sojka, P., Kopeček, I., and Pala, K., editors, *Text, Speech and Dialogue*, pages 629–636, Berlin, Heidelberg. Springer Berlin Heidelberg, ISBN: 978-3-540-39091-6.

Kolář, J., Švec, J., and Psutka, J. (2004). Automatic punctuation annotation in Czech broadcast news speech. In *SPECOM 2004*, Saint Petersburg, Russia.

Kombrink, S., Mikolov, T., Karafiát, M., and Burget, L. (2011). Recurrent neural network based language modeling in meeting recognition. In *Interspeech 2011*.

Kuhn, R. and Mori, R. D. (1990). A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, ISSN: 0162-8828, DOI: 10.1109/34.56193.

Kurimo, M., Enarvi, S., Tilk, O., Varjokallio, M., Mansikkaniemi, A., and Alumäe, T. (2017). Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987, ISSN: 1574-0218, DOI: 10.1007/s10579-016-9336-9.

Levy, T., Silber-Varod, V., and Moyal, A. (2012). The effect of pitch, intensity and pause duration in punctuation detection. In *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, pages 1–4. DOI: 10.1109/EEEI.2012.6376934.

Lin, C.-Y. (2004). *Text Summarization Branches Out*, chapter ROUGE: A Package for Automatic Evaluation of Summaries. `http://aclweb.org/anthology/W04-1013`.

Lin, H., Bilmes, J., and Xie, S. (2009). Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 381–386. DOI: 10.1109/ASRU.2009.5373486.

Lohk, A., Tilk, O., and Võhandu, L. (2013). How to create order in large closed subsets of wordnet-type dictionaries. *Eesti Rakenduslingvistika Ühingu aastaraamat*, 9:149–160, DOI: 10.5128/erya9.10.

Lopyrev, K. (2015). Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712*.

Lu, W. and Ng, H. T. (2010). Better punctuation prediction with dynamic conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 177–186. Association for Computational Linguistics, `http://www.aclweb.org/anthology/D10-1018`.

Ma, S., Sun, X., Xu, J., Wang, H., Li, W., and Su, Q. (2017). Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 635–640. Association for Computational Linguistics, DOI: 10.18653/v1/P17-2100.

Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (1999). Performance measures for information extraction. In *Proceedings of DARPA broadcast news workshop*, pages 249–252.

Mardh, I. (1980). *Headlinese : on the grammar of English front page headlines / by Ingrid Mardh.* Liberlaromedel/Gleerup Lund, ISBN: 9140047539.

Matusov, E., Mauser, A., and Ney, H. (2006). Automatic sentence segmentation and punctuation prediction for spoken language translation. In *IWSLT*, pages 158–165.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černocký, J. (2011a). Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 196–201. DOI: 10.1109/ASRU.2011.6163930.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech 2010*, pages 1045–1048.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S. (2011b). Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2011.5947611.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Černocký, J. (2011c). RNNLM-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, `http://www.aclweb.org/anthology/N13-1090`.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics, `http://aclweb.org/anthology/P09-1113`.

Moore, C. R. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224. Association for Computational Linguistics, `http://aclweb.org/anthology/P10-2041`.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, http://www.iro.umontreal.ca/~lisa/pointeurs/hierarchical-nnlm-aistats05.pdf.

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics, DOI: 10.18653/v1/nallapati2016abstractive.

Ng, J.-P. and Abrecht, V. (2015). Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930. Association for Computational Linguistics, DOI: 10.18653/v1/D15-1222.

Orasmaa, S., Petmanson, T., Tkachenko, A., Laur, S., and Kaalep, H.-J. (2016). Estnltk - nlp toolkit for estonian. In Chair), N. C. C., Choukri, K., Declerck, T., Grobelnik, M., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA), ISBN: 978-2-9517408-9-1.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/P02-1040.

Park, J., Liu, X., Gales, M. J., and Woodland, P. C. (2010). Improved neural network based language modelling and adaptation. In *Interspeech 2010*, pages 1041–1044.

Pascanu, R., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). How to construct deep recurrent neural networks. In *International Conference on Learning Representations 2014 (Conference Track)*. http://arxiv.org/abs/1312.6026.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, http://dl.acm.org/citation.cfm?id=3042817.3043083.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Peitz, S., Freitag, M., Mauser, A., and Ney, H. (2011). Modeling punctuation prediction as machine translation. In *IWSLT*, pages 238–245.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pinter, Y., Guthrie, R., and Eisenstein, J. (2017). Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics, `https://www.aclweb.org/anthology/D17-1010`.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi speech recognition toolkit. In *Proc. 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., and Visweswariah, K. (2008). Boosted MMI for model and feature-space discriminative training. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4057–4060. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2008.4518545.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, ISSN: 0740-7467, DOI: 10.1109/MASSP.1986.1165342.

Ramachandran, P., Liu, P., and Le, Q. (2017). Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391. Association for Computational Linguistics, `http://aclweb.org/anthology/D17-1039`.

Rei, M., Crichton, G., and Pyysalo, S. (2016). Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan. The COLING 2016 Organizing Committee, `http://aclweb.org/anthology/C16-1030`.

Rosenfeld, R. (1994). *Adaptive Statistical Language Modeling; A Maximum Entropy Approach*. PhD thesis, Carnegie-Mellon University. Ph.D. thesis.

Rousseau, A. (2013). Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, (100):73–82.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:9, DOI: 10.1038/323533a0.

Rush, M. A., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics, DOI: 10.18653/v1/D15-1044.

Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech 2014*.

Sakti, S., Ilham, F., Neubig, G., Toda, T., Purwarianti, A., and Nakamura, S. (2015). Incremental sentence compression using lstm recurrent networks. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 252–258. DOI: 10.1109/ASRU.2015.7404802.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, ISSN: 1053-587X, DOI: 10.1109/78.650093.

Schwenk, H., Bougares, F., and Barrault, L. (2014). Efficient training strategies for deep neural network language models. In *NIPS workshop on Deep Learning and Representation Learning*.

Schwenk, H. and Gauvain, J. L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–765–I–768. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2002.5743830.

Schwenk, H. and Gauvain, J.-L. (2005). Training neural network language models on very large corpora. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. `http://www.aclweb.org/anthology/H05-1026`.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics, DOI: 10.18653/v1/P17-1099.

Seide, F., Li, G., Chen, X., and Yu, D. (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 24–29. DOI: 10.1109/ASRU.2011.6163899.

Shen, S., Zhao, Y., Liu, Z., Sun, M., et al. (2016). Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904*.

Shi, Y., Larson, M., and Jonker, C. M. (2014). Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, ISSN: 0885-2308, DOI: 10.1016/j.csl.2014.11.004.

Souvignier, B. and Kellner, A. (1998). Online adaptation of language models in spoken dialogue systems. In *Fifth International Conference on Spoken Language Processing*, volume 6, page 2323.

Stolcke, A. (1998). Entropy-based pruning of backoff language models. *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Stolcke, A. (2002). SRILM – an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Stolcke, A. and Shriberg, E. (1996). Automatic linguistic segmentation of conversational speech. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 1005–1008 vol.2. DOI: 10.1109/ICSLP.1996.607773.

Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Hakkani, D., Plauche, M., Tur, G., and Lu, Y. (1998). Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Fifth International Conference on Spoken Language Processing*, Sydney, Australia.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *Interspeech 2012*, pages 194–197.

Suzuki, J. and Nagata, M. (2017). Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 291–297. Association for Computational Linguistics, `http://aclweb.org/anthology/E17-2047`.

Takase, S., Suzuki, J., Okazaki, N., Hirao, T., and Nagata, M. (2016). Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059. Association for Computational Linguistics, DOI: 10.18653/v1/D16-1112.

Tilk, O. and Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human Language Technologies – The Baltic Perspective*, volume 268, pages 149–152. IOS Press, DOI: 10.3233/978-1-61499-442-8-149.

Tilk, O. and Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687. ISSN: 1990-9770, `https://www.isca-speech.org/archive/interspeech_2015/i15_0683.html`.

Tilk, O. and Alumäe, T. (2016). Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051. DOI: 10.21437/Interspeech.2016-1517.

Tilk, O. and Alumäe, T. (2017). Low-resource neural headline generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 20–26. Association for Computational Linguistics, DOI: 10.18653/v1/w17-4503.

Tilk, O., Demberg, V., Sayeed, A., Klakow, D., and Thater, S. (2016). Event participant modelling with neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 171–182, Austin, Texas. Association for Computational Linguistics, DOI: 10.18653/v1/d16-1017.

Ueffing, N., Bisani, M., and Vozila, P. (2013). Improved models for automatic punctuation prediction for spoken and written text. In *Interspeech 2013*, Lyon, France.

Wang, T. and Cho, K. (2016). Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1319–1329. Association for Computational Linguistics, DOI: 10.18653/v1/P16-1125.

Wang, X., Ng, H. T., and Sim, K. C. (2012). Dynamic conditional random fields for joint sentence boundary and punctuation prediction. In *Interspeech 2012*, pages 1384–1387.

Wegmann, S., Zhan, P., Carp, I., Newman, M., Yamron, J., and Gillick, L. (1999). Dragon systems' 1998 broadcast news transcription system. In *Proc. 1999 DARPA Broadcast News Workshop*, pages 277–280.

Xu, C., Xie, L., Huang, G., Xiao, X., Chng, E., and Li, H. (2014). A deep neural network approach for sentence boundary detection in broadcast news. In *Interspeech 2014*, pages 2887–2891.

Xu, K., Xie, L., and Yao, K. (2016a). Investigating LSTM for punctuation prediction. In *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. DOI: 10.1109/ISCSLP.2016.7918492.

Xu, L., Wang, Z., Liu, Z., Sun, M., et al. (2016b). Topic sensitive neural headline generation. *arXiv preprint arXiv:1608.05777*.

Yu, D., Yao, K., Su, H., Li, G., and Seide, F. (2013). KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7893–7897. ISSN: 1520-6149, DOI: 10.1109/ICASSP.2013.6639201.

Yu, L. C., y. Lee, H., and s. Lee, L. (2016). Abstractive headline generation for spoken content by attentive recurrent neural networks with ASR error modeling. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 151–157. DOI: 10.1109/SLT.2016.7846258.

Zhang, D., Wu, S., Yang, N., and Li, M. (2013). Punctuation prediction with transition-based parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 752–760. Association for Computational Linguistics, `http://www.aclweb.org/anthology/P13-1074`.

Acknowledgments

I feel the greatest gratitude towards my supervisor Tanel Alumäe, who provided the guidance, constructive feedback and the productive work environment that allowed me to focus on the studies. I would also like to thank my second supervisor Leo Võhandu for introducing me to the world of research and inspiring me to start with the doctoral studies.

I am also grateful to my current and former colleagues at the Department of Software Science at TTÜ, with special thanks to Kairit Sirts for noticing my interest in neural networks and bringing me together with the right people to pursue that direction. Lya and Einar Meister have also been very supportive throughout the years.

I thank Vera Demberg and her colleagues at Saarland University for the opportunity to visit and work with them. I definitely learned many things I would not have learned at home.

Finally, I thank my family and friends for their support, and my partner Helen for her patience.

Parts of this work were financially supported by: the European Union through the European Regional Development Fund (project 3.2.1201.13-0010) and the Centre of Excellence in Estonian Studies; the Estonian Ministry of Education and Research target-financed research theme No. 0140007s12 and the National Program for Estonian Language Technology funded by them; and the Tallinn University of Technology project Estonian Speech Recognition System for Medical Applications.

Abstract

Language models encapsulate the knowledge about how smaller elements like words are strung together to form coherent sentences and texts in a language. This knowledge is useful in many applications such as automatic speech recognition, machine translation and predictive keyboards that are gaining importance in our daily lives at an accelerating pace.

Training good language models requires a lot of data which is usually limited by factors like the number of speakers of the language, the particular domain of use or some other aspects of the specific task. When the amount of data is small the methods for training the models have to become smarter. Neural networks based language models have been shown to be one of these smarter methods that generalize well and can be informed by a wider context than previous approaches. In this work we present and evaluate additional improvements for neural network language models in three resource-constrained areas.

First, we explore the problem of modeling language in small domains. We successfully adapt a FFNN LM multi-domain extension (Alumäe, 2013) for RNN LMs which enables using a single joint RNN for modeling multiple target domains that shares knowledge about common language patterns while also factoring in domain-specific variations. Additionally we show that the multi-domain component can be used for language model adaptation on small target domains where the small amount of domain-specific parameters is beneficial for preventing overfitting. Both methods show improvements in PPL compared to the unadapted versions, but require further work for determining whether these improvements translate into better performance of the system as a whole.

In the second part of the work we focus on detecting hidden events between words (punctuation in our case) with models that can utilize both unannotated texts and smaller prosody annotated datasets during training. We present two RNN models that are trained on unannotated texts — a unidirectional LSTM model and a bidirectional model with an attention mechanism — and a two-stage training method that enables adapting the models on a limited amount of prosody annotated data. The unidirectional

model is the first published work on applying RNNs to the punctuation restoration problem and the bidirectional model surpasses the previous state of the art model on English. Both models are a huge improvement over conventional hidden event LMs and the two-stage models with prosodic features improve the results further.

Lastly, we present and analyze pre-training methods for low-resource neural headline generation. The three methods we develop can be combined to fully utilize the text in the training data and pre-train all parameters of the model, while the previous approaches generally were trained on only a few first sentences of documents and pre-trained a subset of the parameters (usually just word embeddings) or used no pre-training at all. Experiments on two languages show that all pre-training methods reduce PPL, but the more appropriate ROUGE evaluation scores do not correlate much with PPL — methods that pre-train parameters close to the output are effective at reducing PPL but actually degrade the performance in terms of ROUGE. Nevertheless, pre-training the source side of the model consistently gives significant improvements over the model that is not pre-trained and the combination of all methods performs the best on one language.

Overall the results support our claims, although the ones related to the small domain language models require additional experiments to verify the practical benefits. We also present several ways to improve the proposed methods in future work.

Kokkuvõte

Keelemudelid peidavad endas infot selle kohta, kuidas sõnadest või muudest keeleühikutest keeleliselt korrektseid lauseid ja tekste moodustatakse. Selline info on hädavajalik paljudes igapäevaelus aina tähtsamaks muutuvates rakendustes nagu automaatne kõnetuvastus, masintõlge ja järgmist sisestust ennustavad klaviatuurid.

Hea keelemudeli treenimiseks kulub ohtralt treeningandmeid. Paraku on andmete hulk tihti erinevatel põhjustel piiratud. Piiravateks teguriteks võivad olla nii keele kõnelejate vähesus, eripärase keelestiiliga kasutusvaldkond kui ka mõni rakenduse tehniline iseärasus. Kui sobivate andmete juurde hankimine osutub keeruliseks, siis ei jää muud üle kui kasutada paremaid meetodeid ja mudeleid. Tehisnärvivõrkudel põhinevad keelemudelid on osutunud edukaks meetodiks, mis suudavad isegi väheste treeningandmete puhul paremini üldistada ning on võimelised haarama laiemat konteksti kui varasemad lähenemised. Käesolevas töös pakume välja mitmeid meetodeid ja mudeleid koos empiiriliste hinnangutega täiendamaks tehisnärvivõrkudel põhinevaid keelemudeleid kolmes piiratud ressurssidega valdkonnas.

Esimene uuritav probleem hõlmab eripärase keelestiiliga kasutusvaldkondade (domeenide) keele modelleerimist. Kohandame edukalt pärisuunalistele närvivõrkudele disainitud mitme domeeni korraga samas mudelis modelleerimist võimaldava komponendi rekurrentsete närvivõrkude jaoks sobivaks. Komponent võimaldab mudelil jagada infot üldiste keelemustrite ning sõnatähenduste kohta, olles samaaegselt võimeline modelleerima iga domeeni iseärasusi. Lisaks töötame välja meetodi, mille abil saab mitme domeeni komponenti kasutada üldise keelemudeli adapteerimiseks väiksele sihtdomeenile. Adapteerimisel osutub komponendi vähene domeeni jaoks eraldatud parameetrite arv kasulikuks omaduseks vältimaks vähestel andmetel ületreenimist. Mõlemad meetodid parandavad keelemudeli täpsust entroopias mõõdetuna, samas rakendusliku väärtuse tõestamine vajab täiendavaid eksperimente.

Teine osa tööst keskendub sõnadevaheliste peidetud sündmuste (antud juhul kirjavahemärkide) tuvastamisele. Töötame välja meetodi, mis võimaldab

sama mudelit treenida nii lihttekstide kui ka prosoodiliste tunnustega rikastatud andmete peal. Lihttekstide peal treenimiseks esitleme kahte rekurrentsetel närvivõrkudel põhinevat mudelit, millest esimene on ühesuunaline ning teine kahesuunaline koos tähelepanumehhanismiga. Pro-soodiliste tunnustega rikastatud andmete peal adapteerimiseks töötame välja kaheastmelise treeningmeetodi. Ühesuunalisel rekurrentsel närvivõrgul põhinev mudel on meile teadaolevalt esimene publitseeritud töö, kus kasutatakse rekurrentset närvivõrku kirjavahemärkide taastamiseks ning kahesuunaline mudel edestab suure edumaaga eelmist parimat mudelit ingliskeelse andmestiku peal. Mõlemad mudelid on suur edasiminek võrreldes tavapäraste n-grammidel põhinevate mudelitega ning kaheastmeline treenimine võimaldab edukalt prosoodilisi tunnuseid ära kasutada, parandades mõlema mudeli tulemusi veelgi.

Viimases osas töötame välja ja analüüsime erinevaid meetodeid automaatse pealkirjastamise mudeli eeltreenimiseks, parandamaks genereeritud pealkirjade kvaliteeti väheste treeningandmetega olukordades. Esitame kolm meetodit, mille kombineerimisel on võimalik treeningtekstid täies ulatuses ära kasutada ning mudeli kõiki parameetreid eeltreenida. Varasemad lähenemised piirdusid tavaliselt dokumentide paari esimese lause kasutamisega ning parimal juhul eeltreeniti ainult osa mudeli parameetritest (enamasti ainult sõnavektorid). Eksperimendid kahes erinevas keeles andmestike peal näitavad olulisi mudeli entroopia vähenemisi kõigi eeltreenimise meetodite puhul. Ülesande jaoks sobivamad ROUGE hinnangud aga ei korreleeru entroopiaga ning mõned meetodid, mis eeltreenivad väljundi lähedal paiknevaid parameetreid, toodavad kehvemaid tulemusi kui eeltreenimata mudel. Sellele vaatamata osutub eeltreenimine üldiselt efektiivseks meetodiks genereeritud pealkirjade kvaliteedi parandamisel. Sisendi poolel paiknevate parameetrite eeltreenimine parandab mõlemal juhul oluliselt tulemuste kvaliteeti ning kõigi meetodite kombinatsioon annab parimaid tulemusi ühe keele puhul.

Töö tulemused üldiselt toetavad esitatud väiteid, kuigi eripärase keelestiiliga kasutusvaldkondade keele modelleerimise meetodid vajavad täiendavaid eksperimente tõestamaks tulemuste praktilist väärtust. Pakume ka välja mitmeid tuleviku töö jaoks võimalikke suundi parandamaks esitatud meetodite nõrkusi.

Appendix A

Publication I

Tilk, O. and Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human Language Technologies – The Baltic Perspective*, volume 268, pages 149–152. IOS Press, DOI: 10.3233/978-1-61499-442-8-149

Human Language Technologies – The Baltic Perspective
A. Utka et al. (Eds.)

doi:10.3233/978-1-61499-442-8-149

Multi-Domain Recurrent Neural Network Language Model for Medical Speech Recognition

Ottokar TILK [a,1] and Tanel ALUMÄE [a]
[a] *Institute of Cybernetics at Tallinn University of Technology, Estonia*

Abstract. We evaluate back-off n-gram and recurrent neural network language models for an automatic speech recognition system for medical applications. We also propose an effective and simple multi-domain recurrent neural network architecture which enables training a joint model for all domains. The multi-domain recurrent neural network model outperforms all other compared models.

Keywords. Language modeling, recurrent neural network language model, multi-domain language model

Introduction

An important part of any automatic speech recognition (ASR) system is the language model (LM) which evaluates the probabilities of word sequences. The traditional approach for language modelling is the back-off n-gram LM which performs well but suffers from data-sparsity problem. Neural network (NN) LMs overcome this problem by projecting input words into continuous space and estimating word probabilities there [1]. While NN LMs enable larger contexts to be utilized than back-off n-gram LMs, they still are an n-gram approach (with fixed context length n-1). Recurrent neural network (RNN) LMs remove the fixed context length limitation and this seems to help as shown in [2] where RNN LMs outperform the feed-forward NN LMs.

LMs are typically trained on large text corpora of different domains with one or more of them being the target domain. Often the best model for the target domain is acquired by exploiting the inter-domain similarities. Back-off n-gram models exploit the similarities by finding optimal interpolation coefficients for combining all the domain-specific models into a target domain model. Maximum entropy and NN LMs can use the adaptation approach where the general model is carefully adjusted for the target domain [3,4]. Multi-domain NN LMs [5] can be trained for all domains simultaneously which is convenient if there are multiple target domains.

In this paper we train LMs for a medical ASR system using medical corpora. We compare n-gram and RNN LMs and propose a method to exploit the inter-domain similarities in the form of a novel multi-domain RNN LM.

[1] Corresponding Author. E-mail: ottokar.tilk@phon.ioc.ee

1. About the Medical Corpora

The medical corpora consist of radiology reports from 10 different domains (X-ray, computed tomography, ultrasound, etc.). All 10 domains are considered as target domains. The training set sizes of the domains vary in the range of 24.0K to 3.6M words with a total size of 10.8M words.

Our initial experiments with back-off n-gram LMs revealed some interesting properties of the medical corpora:

1. Including the general (non-medical) text corpora in the training of the medical LM is not practical;
2. Medical corpora consist of very different domains;
3. Exploiting the inter-domain similarities should give the best results.

Below we elaborate on these points in more detail.

What makes the medical applications model unique and difficult to train is that it has very specific vocabulary and style which is uncharacteristic to normal speech or text, thus limiting the usefulness of the general text corpora. This was reflected by the lack of improvement in terms of perplexity on the test set when we included the general text corpora in the training of the model.

Another challenge is that the medical dataset consists of texts from several very different domains. The differences are indicated by the very big (usually >0.9) optimal interpolation coefficients for the in-domain models when optimizing for the best combination of domain-specific n-gram language models for the target domain. For each domain there is a highly varying amount of training data available. Therefore, it is important for the method to be able to exploit the little similarities there are between the domains and to factor in the relatively big inter-domain differences while not overfitting the potentially small amount of in-domain data.

Our back-off n-gram LM experiments showed that linear interpolation of domain-specific models outperforms both a general model over all domains and using separate domain-specific models. This suggests that while the domains are different, they still share enough similarities to enable training a better model by exploiting the similarities in the whole set of medical corpora than training a separate model on each domain-specific corpus.

2. The Multi-Domain RNN LM Architecture

We implemented a multi-domain RNN LM inspired by the multi-domain NN LM from [5] because of its effectiveness, simplicity and very small number of domain-specific parameters. Small amount of domain-specific parameters enables the model to adapt to domains with little training data without the danger of overfitting. The method is also good at exploiting the similarities between the domains.

This method has not been used with RNN LMs before. Switching from feedforward to recurrent architecture requires finding a new way to apply the adaptation factors.

The first problem is deciding the location of the adaptation layer. The original multi-domain NN LM adds the adaptation layer between the projection and the hidden layer of the feedforward network. Our method puts the adaptation layer between the hidden

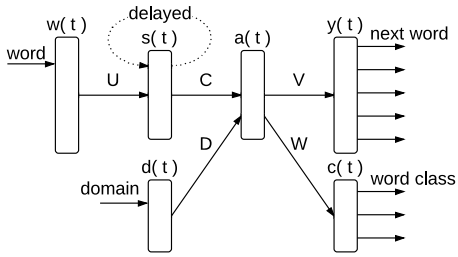


Figure 1. The Multi-Domain RNN LM.

Table 1. Comparison of models in terms of perplexity on the test set.

Model	PPL
n-gram	22.2
RNN	20.9
MD RNN	17.5
MD RNN + n-gram	15.6

and output layer of the recurrent network. This location requires the least amount of domain-specific parameters and enables the adaptation layer to double as a compression layer. Compression layer reduces the computational complexity of the model and was originally proposed in [6].

The second question is concerned with applying the domain-specific parameters in the adaptation layer. The multi-domain NN LM uses domain-specific multiplicative factors on the inputs to the adaptation layer. During experiments on NN LMs we discovered that this method requires shuffling the training samples to work properly. Using multiplicative factors on unshuffled training data was worse than using no adaptation at all. This seemed to be caused by the model parameters overfitting to the factors of more recently seen domains and becoming less compatible with domain factors seen earlier.

The shuffling requirement can not be satisfied with RNN LMs. For sequential training we propose a different method which, though benefits from shuffling, is much less sensitive to seeing long sequences of samples from the same domain. This method uses additive factors instead of multiplicative ones and can alternatively be thought of as a domain input (an idea also considered in [5]) or domain-specific bias to the adaptation layer. The adaptation layer state $a(t)$ at time step t in our approach is computed according to the following formula:

$$a(t) = f\big(s(t)C + d(t)D\big) \tag{1}$$

where $f(z)$ is the logistic sigmoid activation function; C is the weight matrix between the hidden state layer s and the adaptation layer a; D is the weight matrix between the domain input d and the adaptation layer a; $s(t)$ is the state of the hidden state layer and $d(t)$ is the domain input in the form of a one-of-N coded vector indicating the current domain (see Figure 1). The rest of the architecture is identical to the one described in [6].

Using addition instead of multiplication when applying the adaptation factors eliminates the adaptation factors from the gradients of the model parameters behind the adaptation layer. This way the parameters can fit to prediction error rather than adaptation parameters. Similar reasoning applies to factor gradients as well.

In our experiments on NN LMs, where we use shuffling of training samples, the additive factors show similar performance to multiplicative factors.

3. Experimental Results

We use a modified version of Mikolov's RNN LM toolkit [7] in our experiments. Both the simple and multi-domain RNN LM use the following set-up: number of classes in out-

put layer: 230; size of vocabulary: 52293; state layer size: 600; compression/adaptation layer size: 300; number of backpropagation through time steps: 3; learning rate: 0.1; L2 regularization coefficient: 1e-7. Additionally, the multi-domain RNN LM has 10 domain inputs. We use a single model for all domains in both RNN model experiments.

The baseline back-off n-gram model is a separate linearly interpolated model for each domain with interpolation weights optimized on the corresponding development set using the SRILM toolkit [8]. The order of the n-gram models is 4 and we use the modified Kneser-Ney discounting.

The results of the experiments can be seen in Table 1. A simple RNN LM brings a 6% relative perplexity improvement over interpolated back-off n-gram models. The multi-domain RNN LM increases the difference to 21% and linearly combining with the n-gram model raises it to 30%.

4. Conclusion

We trained and compared different types of LMs out of which the newly proposed multi-domain RNN LM turned out to be the most effective one in terms of perplexity. An additional benefit of the multi-domain architecture was the fact that it's a joint model for all domains thus enabling simple switching between target domains in contrast to the n-gram models where we had to use a different model for each domain.

In our future work we plan to perform ASR experiments to see how well the improvements in perplexity translate into improvements in word error rate. Testing the multi-domain RNN LM on other languages and domains is also in our plans.

Acknowledgments

This research was supported by the European Union through the European Regional Development Fund.

References

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*, pp. 137–186, Springer, 2006.

[2] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Černocký, "Empirical evaluation and combination of advanced language modeling techniques.," in *INTERSPEECH*, pp. 605–608, 2011.

[3] T. Alumäe and M. Kurimo, "Domain adaptation of maximum entropy language models," in *Proceedings of the ACL 2010 Conference Short Papers*, pp. 301–306, Association for Computational Linguistics, 2010.

[4] J. Park, X. Liu, M. J. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation.," in *INTERSPEECH*, pp. 1041–1044, 2010.

[5] T. Alumäe, "Multi-domain neural network language model," in *INTERSPEECH*, pp. 2182–2186, 2013.

[6] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, 2011.

[7] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Černocký, "RNNLM-recurrent neural network language modeling toolkit," in *Proc. of the 2011 ASRU Workshop*, pp. 196–201, 2011.

[8] A. Stolcke, "SRILM – an extensible language modeling toolkit.," in *Proc. Intl. Conf. on Spoken Language Processing*, pp. 901–904, 2002.

Appendix B

Publication II

Tilk, O. and Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687. ISSN: 1990-9770, https://www.isca-speech.org/archive/interspeech_2015/i15_0683.html

LSTM for Punctuation Restoration in Speech Transcripts

Ottokar Tilk, Tanel Alumäe

Institute of Cybernetics
Tallinn University of Technology, Estonia
ottokar.tilk@phon.ioc.ee, tanel.alumae@phon.ioc.ee

Abstract

The output of automatic speech recognition systems is generally an unpunctuated stream of words which is hard to process for both humans and machines. We present a two-stage recurrent neural network based model using long short-term memory units to restore punctuation in speech transcripts. In the first stage, textual features are learned on a large text corpus. The second stage combines textual features with pause durations and adapts the model to speech domain. Our approach reduces the number of punctuation errors by up to 16.9% when compared to a decision tree that combines hidden-event language model posteriors with inter-word pause information, having largest improvements in period restoration.

Index Terms: neural network, punctuation restoration

1. Introduction

The output of most automatic speech recognition (ASR) systems consists of raw word sequences, without any punctuation symbols. While this is sufficient for some tasks, such as indexing and retrieval as well as dictation where punctuation symbols might be entered explicitly by voice, most speech transcription applications benefit from automatically inserted punctuation symbols. This serves two purposes. First, it makes the ASR-based transcripts easier to read and understand for humans. Second, in many cases, it also makes downstream machine processing of the generated texts more accurate, as many natural language processing tools, such as sentiment analyzers, syntactic parsers, information extraction and machine translation systems, are trained on written texts that include punctuation, and thus expect them to be present also in the input texts.

There have been many previous studies on automatic punctuation restoration in speech transcripts. Probably the most widely used approach is based on the so-called hidden event language model (LM) which uses a traditional N-gram LM trained on texts that include punctuation tokens [1]. During decoding, the LM is used to recover the most probable sequence of words and hidden punctuation symbols. Punctuation restoration can also be treated as a sequence labelling task and solved using conditional random fields (CRFs) [2]. This model allows to combine various textual features, such as LM scores, token n-grams, sentence length and syntactic features which is found to give large improvements over purely lexical features [3]. Many approaches combine textual information with acoustic/prosodic features, such as pause length between words, phoneme lengths, pitch and energy, using a classifier such as decision tree [4] or maximum entropy model [5]. Often the purely lexical model trained on large text corpora is combined with a model containing acoustic features, trained on a smaller speech corpus. This is usually done using either (log-)linear interpolation of model predictions [4] or using the posteriors of the lexical model as additional features to the overall model [6].

This paper describes a punctuation restoration system for automatically transcribed Estonian broadcast speech that uses long short-term memory (LSTM)[7]. LSTM, a type of recurrent neural network (RNN), has been used for a variety of supervised sequence labelling tasks, including phoneme classification [8] and spoken language understanding [9]. We are not aware of any prior work using RNNs for punctuation restoration.

Neural networks provide a flexible architecture for constructing and synthesizing complex models. We take advantage of this flexibility by training a punctuation restoration model in two phases. First, a large text corpus is used to train a model that uses only words as features. Then a new model is trained on a smaller pause annotated corpus, using pause durations and first phase models uppermost hidden layer outputs as features. The resulting punctuation system is currently used in our publicly available Estonian speech transcription system[1]. Its performance on automatically transcribed data can be viewed on our constantly updated archive of Estonian broadcast speech [2][10]. The source code of the model is also publicly available [3].

The following section describes how we use LSTM for punctuation restoration. Section 3 presents evaluation data, evaluation metrics and experimental results. Section 4 concludes the paper.

2. Method

We focus on restoring two most important and frequent types of punctuation — commas and periods. Question marks, exclamation marks, semicolons and colons are mapped to periods and all other punctuation symbols are removed from the corpora. The model we use for this task is a LSTM RNN with forget gates [11] and peephole connections [12] in LSTM layers.

2.1. LSTM vs N-gram

The LSTM RNN has several advantages over the widespread hidden event N-gram LM.

First, one of the problems with N-gram models is the data sparsity issue. A better model should to be able to generalize to contexts that were not seen during training. As it is well known from language modeling, neural networks are much better at generalizing to unseen sequences, by learning distributed representations of words [13] or entire contexts as it is the case with RNNs [14]. This suggests that LSTM RNNs should be able to learn similar representations for contexts around similar punctuations and make accurate predictions even in unseen contexts.

[1] http://bark.phon.ioc.ee/webtrans/
[2] http://bark.phon.ioc.ee/tsab
[3] https://github.com/ottokart/punctuator

Another weakness of N-gram models is that their context size is limited to a fixed number of tokens. Although it has been shown that increasing the context size does not help as much as getting more data [15], it seems unjustified to expect that the relevant context size is the same for both types of punctuation and remains constant across the entire text. Therefore, one of our requirements is that the model should be able to dynamically decide on how long context is relevant. RNNs fit this requirement and are able to utilize arbitrary length sequences. Although in practice non-LSTM RNNs have difficulties in remembering long range dependencies due to vanishing gradients [16], this problem can be alleviated by using LSTM units.

Neural networks are very simple to augment with additional features, such as those derived from speech prosody, which gives them another advantage over N-gram models.

The main disadvantage of neural networks is their training speed, although it is not as huge problem as in language modeling as the output layer is small.

2.2. Input features

Our approach to punctuation restoration relies on both textual information and pause durations between words. Contrary to many previous works, we don't use any other prosodic features, such as F_0 contours, phoneme durations and energy values. This has two reasons: first, previous research [17, 4] has shown that pause duration is by far the most useful and robust prosodic feature for predicting punctuation symbols; second, it is easy to extract pause durations from word-aligned recognition output that can be generated using any decoder, without the need to reanalyze the audio signal. We also opted against using other linguistic features, such as part-of-speech tags, because we wanted our approach to be as portable to other languages as possible.

The textual component of our model decides the suitable punctuation for a slot based on the word after the slot and the history of all previously seen words. The word after the slot is given as one-hot encoded input vector and the information about preceding words is stored in the LSTM memory units. The next word is important for predicting both commas and periods correctly. In Estonian language there are many words that are almost always preceded by a comma and thus it is essential to know the following word. Also, in item listings it is indicative whether the following word is from the same category. For periods it helps to detect thematic changes and to recognize words which typically start a new sentence. Since there are many easy cases for commas, we can expect the textual model to be better at predicting commas than periods.

Pauses between words are most informative for predicting periods. This became apparent when we trained a simple model using a small window of pauses only as input features. The model achieved an F-score of 0.53 for periods, but was unable to predict commas. Although, in certain contexts, a small pause can be good indicator for a comma. Therefore, in conjunction with word features, the comma annotation performance should also improve, but we expect periods to benefit the most.

Combining textual and pause duration information should provide a model with balanced performance across different punctuations.

2.3. Two-stage model

Since the amount of pause annotated data is relatively small, we use a two-stage training procedure in which a purely textual model (T-LSTM) is trained first. The forward pass of the T-LSTM model is described in the following equations:

$$
\begin{aligned}
y_0(t) &= \tanh\big(W_0 x_0(t)\big) \\
y_1(t) &= LSTM\big(y_0(t)\big) \\
y_2(t) &= Softmax\big(W_2 y_1(t)\big)
\end{aligned}
$$

where x_0 is the one-hot encoded vector representing the input word following the punctuation slot, y and W are the layer activation vectors and weight matrices respectively where the subscript matches the layer index. The $LSTM$ is defined as in [18], except biases are omitted because we found they brought no noticeable improvement.

After obtaining the textual model, the final model utilizing text and pause duration (TA-LSTM-p) is trained in the second stage. As proposed in [19], we treat the last hidden layer outputs of the T-LSTM as high level features learned by the network, representing everything the model knows about the textual input. TA-LSTM-p then utilizes both pause durations and these high level features of text to make decisions about punctuations. To construct a TA-LSTM-p the output layer softmax classifier of T-LSTM is discarded and its last hidden layer features with the current slot pause duration are used as inputs to the TA-LSTM-p. The architecture of the TA-LSTM-p is identical to the T-LSTM with the exception of inputs and the omitted first hidden layer. During the second phase of training the model learns to use the now available pause duration information in conjunction with textual features. It also enables the model to adapt to the style of speech by learning a more suitable classifier for the target domain. The remaining T-LSTM layers are still used in the forward pass of the TA-LSTM-p, but are fixed during training. Stacking a new classifier on top of the T-LSTM features has two advantages when compared to e.g. adapting the existing T-LSTM parameters. First, the size of the model trained on the smaller pause annotated corpus can be easily adjusted to be optimal for the smaller corpus size — which as a side-effect can be also faster to train. Second, according to our experiments, stacking a new classifier performs better than adapting the existing parameters. While in the T-LSTM, the function of the LSTM layer is to remember the textual context, the LSTM layer in TA-LSTM-p serves to have information about previous pauses (e.g. whether the current pause is long enough to indicate punctuation or just characteristic to current context) and previous T-LSTM features. The forward pass of the TA-LSTM-p model takes the following form:

$$
\begin{aligned}
x_1(t) &= [y_1(t), p(t)] \\
y_3(t) &= LSTM\big(x_1(t)\big) \\
y_4(t) &= Softmax\big(W_4 y_3(t)\big)
\end{aligned}
$$

where x_1 is the input to the TA-LSTM-p model, which is obtained by concatenating the second hidden layer activations y_1 of T-LSTM and pause duration p of current slot. The output layer y_4 represents the probability distribution over possible punctuations — comma, period or no punctuation. During punctuation restoration the slot t is filled with the most probable punctuation according to y_4. Both T-LSTM and TA-LSTM-p are described in Figure 1.

3. Experiments

The LSTM model is trained similarly in both stages. Gradients are computed with back-propagation through time [20] over 5 time steps and the weights are updated using AdaGrad [21].

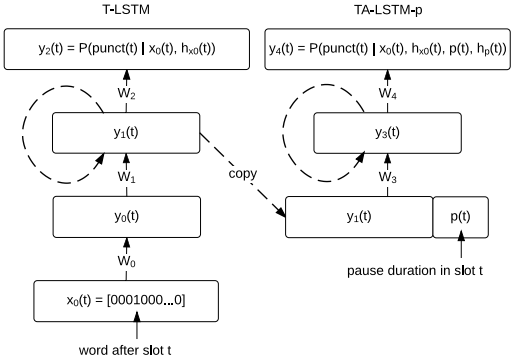


Figure 1: Description of the T-LSTM and TA-LSTM-p model. W_1 and W_3 represent the external input weights (for cell inputs and all gates) of the corresponding LSTM layers. The output of the T-LSTM model is a posterior probability distribution over punctuations $punct(t) = \{comma, period, \varnothing\}$ for slot t given word after the slot $x_0(t)$ and preceding word history $h_{x0}(t)$. TA-LSTM-p has a similar output, but is additionally conditioned on slot pause duration $p(t)$ and pause history $h_p(t)$.

Learning rate starts from 0.1 and when there is no sufficient improvement on the validation set we start to divide the learning rate by 2 at each epoch (adopted from [22]). Training is stopped when the shrinking learning rate no longer yields enough improvements or the maximum number of 20 epochs has been reached. Weights are initialized to random uniform values in a range of ± 0.005 and all hidden states to zeros. To speed up the training, the dataset is split into 100 sequences and these sequences are fed to the network in parallel as mini-batches.

The first and second hidden layer of the T-LSTM model consist of 100 tanh units and 100 single-cell LSTM blocks respectively. Input vocabulary consists of the 100K most frequent words in the training corpus plus two special symbols for unknown words and an end of input.

The TA-LSTM-p input size is 101 (T-LSTM features plus current slot pause duration). Hidden layer has 100 LSTM units.

We use two baselines: a hidden event LM and a decision tree that combines textual and inter-word pause information. The 4-gram hidden event LM uses a vocabulary of the same 100K words as the LSTM models plus punctuation symbols. The model is built by interpolating the models compiled from the individual text sources using coefficients optimized on the development set. The model is smoothed using Kneser-Ney discounting and n-gram probabilities accounting for less than 10^{-7} training set perplexity improvement are pruned. The second baseline is inspired from the models proposed in [4, 6]: a decision tree (4-gram+DT-p) is trained on the pause annotated corpus, using pause durations and the posterior probabilities of the punctuation symbols assigned by the hidden n-gram LM as features. This allows the decision tree to directly benefit from both inter-word pause information as well as a large text corpus which has no corresponding speech data available, similarly to the TA-LSTM-p model.

For a comparison with the 4-gram model, we also train a TA-LSTM-p model without pause duration inputs (i.e. just adapt it on the target domain data). This also helps us to assess the effectiveness of the TA-LSTM-p in utilizing pause duration information and compare it to the baselines DT-p approach. We refer to this adapted, purely textual model as TA-LSTM. We could have used the T-LSTM model for comparison, but we had two reasons to not do that — first, the N-gram model is adapted for target domain, thus for fairness the LSTM model should also be adapted; second, the T-LSTM model did not perform well on the speech data as the training data consisted mostly of written text (this might imply that local contexts don't vary between styles as much as longer ones).

To demonstrate the importance of textual features trained on a large corpus, we train an augmented T-LSTM model with additional pause duration input on pause annotated text only. This model is referred to as T-LSTM-p.

All models are evaluated on force-aligned manual transcriptions and the transcripts generated by the ASR system. In order to insert reference punctuation marks into automatic transcripts, the manual transcripts were aligned with the ASR output, using minimum cost edit distance, and the punctuation marks in the reference texts were propagated to the hypothesized texts.

Table 1: Number of tokens of textual data and the amount of hours of audio data used for training the punctuation model.

Source	*#Tokens*	*#Hours*
Newspapers	203M	
Web	74M	
Fiction	35M	
Magazines	29M	
Parliament	15M	
Social media	28M	
Lecture speech	283K	39.3
Broadcast news	127K	32.1
Broadcast conversations	505K	74.0
Total	*386M*	*145.4*

3.1. Data

Textual and audio data used for training the punctuation model is summarized in Table 1. The audio data is force-aligned using the speech recognition system described below. Inter-word pause durations, used for training the 4-gram+DT-p, TA-LSTM-p and T-LSTM-p model, are then captured from the alignments.

As development data, we use two hours of broadcast news and 4.4 hours of broadcast conversations (radio talkshows and telephone interviews). The test set contains two hours of broadcast news and 5.6 hours of broadcast conversations.

3.2. Speech recognition system

The ASR system that is used for aligning punctuation model training data and producing the ASR hypotheses for the evaluation data is described in [23], although some details have been improved since then. Speech recognition is implemented using the Kaldi toolkit [24]. The acoustic models are trained from around 178 hours of speech from various sources. We use multisplice DNN-based acoustic models that take an i-vector of the current speaker as additional input to the DNNs for unsupervised speaker adaptation.

The ASR LM is compiled from the same data that is used for training the text-based punctuation model (Table 1). A 4-gram LM is built by interpolating models trained on the individual sources, using a vocabulary of 200K compound-split words. The final LM is heavily pruned to be usable for decoding. One-

Table 2: Results on manually transcribed reference and ASR output test set.

	Reference text							*ASR output*						
Model	*P(C)*	*R(C)*	*F(C)*	*P(P)*	*R(P)*	*F(P)*	*Err*	*P(C)*	*R(C)*	*F(C)*	*P(P)*	*R(P)*	*F(P)*	*Err*
4-gram	0.78	0.60	0.68	0.47	0.26	0.33	9.67	0.70	0.54	0.61	0.38	0.19	0.25	11.32
4-gram+DT-p	0.76	0.70	0.73	0.64	0.60	0.62	8.22	0.66	0.62	0.64	0.52	0.48	0.50	10.97
T-LSTM-p	0.79	0.63	0.70	**0.69**	0.60	0.64	8.24	0.70	0.57	0.63	**0.57**	0.49	0.53	10.46
TA-LSTM	0.74	**0.72**	0.73	0.63	0.43	0.51	8.20	0.65	**0.65**	0.65	0.49	0.32	0.39	11.02
TA-LSTM-p	**0.82**	0.70	**0.76**	0.68	**0.77**	**0.72**	**6.83**	**0.71**	0.62	**0.66**	0.55	**0.61**	**0.58**	**10.01**

pass recognition is used, followed by rescoring of the lattices with a larger LM.

The word error rate (WER) of the system is around 17%.

3.3. Metrics

All models are evaluated in terms of four different metrics. First we give an overall classification error rate Err defined as incorrectly punctuated slots divided by the total number of slots. For a more detailed overview we also report precision, recall and F-score of commas and periods.

3.4. Results on reference text

Left side of Table 2 shows the results on manually transcribed reference text. The 4-gram model has decent performance in restoring commas but fails miserably when it comes to periods where especially the poor recall of 0.26 stands out. This might indicate that periods depend on longer context than the 4-gram model is able to utilize and require better generalization.

The LSTM model trained on purely textual features (TA-LSTM) achieves a 15.2% relative decrease in error rate over 4-gram model on reference text. While there are improvements in comma restoration, most of the gains come from better period restoration performance. This confirms that longer context and better generalization of LSTMs is indeed helpful when it comes to punctuation restoration, particularly for periods. TA-LSTM performance may be also partially attributed to the superiority of the adaptation scheme used (the 4-gram model uses linear interpolation). Just as with the 4-gram model, the TA-LSTM model is still worse at period restoration than comma restoration, although the difference is noticeably smaller.

Adding pause duration features yields noticeable reductions in error rate — TA-LSTM-p reduces error by 16.7% relative over TA-LSTM and 4-gram+DT-p improves by 15.0% relative over 4-gram model. As expected, the biggest improvement in TA-LSTM-p performance comes from the large jump in period restoration recall, followed by a noticeable rise in comma restoration precision. The TA-LSTM-p model has a relatively balanced performance over punctuation marks where both commas and periods have similar F-scores while the precision is larger for commas but recall is higher for periods. 4-gram+DT-p compared to 4-gram also mainly improves in period restoration, but shows a smaller improvement in comma restoration.

When comparing T-LSTM-p, a model trained on the small pause annotated corpus only, to the TA-LSTM-p model, it becomes clear that using textual features trained on a large corpus helps a lot. Biggest improvement is in period recall, again hinting that good representations of long contexts are important for period restoration.

The TA-LSTM-p model beats the 4-gram+DT-p baseline by 16.9% relative. It has much higher period recall and higher precision for both punctuation marks.

3.5. Results on ASR output

Experiments on ASR output (Table 2, right) show similar trends as the reference text, although the differences between models are much smaller. TA-LSTM reduces error rate by 2.7% relative compared to 4-gram. 4-gram+DT-p improves over 4-gram by 3.1% and TA-LSTM-p over TA-LSTM by 9.2%. Comparing TA-LSTM-p to the 4-gram+DT-p baseline shows relative error rate reduction of 8.8%.

It is clear that all models suffer from the errors made by the ASR system. It is also evident that LSTM models suffer more and period restoration performance declines more than for commas. This might be another indicator that restoring periods requires larger context (which LSTM models are able to use) and as context size grows the more likely it is to encounter errors made by the ASR system. As LSTM models suffer more from ASR WER, they also have a higher potential for improvements when the WER of the ASR systems improves.

Another thing to note is that alignment method used to propagate reference punctuations to ASR output is not error free either. Manual inspection of punctuations restored by TA-LSTM-p model revealed that many decisions made by the model that were counted as mistakes were actually better than the expected punctuations. While flawed, our current ASR output test set should still give a rough performance estimate.

4. Conclusions

This paper presented a novel LSTM RNN model for restoring periods and commas in speech transcripts. The model was trained in two stages where in the first stage textual features were learned and in the second stage pause durations and textual features were combined and the model was adapted to the target domain. Based on experiments with Estonian broadcast speech, the model showed a balanced performance for both punctuation types and reduced the number of restoration errors by 16.9% on reference text and by 8.8% on ASR output when compared to a decision tree that combines the output of a hidden event LM with pause durations. Most of the gains came from largely improved period restoration.

Future research includes other languages, restoring additional types of punctuation and using larger amount of prosodic features. Also, current LSTM model only peeked one word after the punctuation slot. It would be interesting to find out whether looking further into the future context improves the performance. Bidirectional models [25] can also be considered.

5. Acknowledgements

The work was supported by the European Union through the European Regional Development Fund, project 3.2.1201.13-0010 and by the Estonian Ministry of Education and Research target-financed research theme No. 0140007s12.

6. References

[1] A. Stolcke, E. Shriberg, R. A. Bates, M. Ostendorf, D. Hakkani, M. Plauch, G. Tr, and Y. Lu, "Automatic detection of sentence boundaries and disfluencies based on recognized words," in *ICSLP 1998*, Sydney, Australia, 1998.

[2] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *EMNLP 2010*, Cambridge, MA, USA, 2010.

[3] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text," in *Interspeech 2013*, Lyon, France, 2013.

[4] J. Kolář, J. Švec, and J. Psutka, "Automatic punctuation annotation in Czech broadcast news speech," in *SPECOM 2004*, Saint Petersburg, Russia, 2004.

[5] J. Huang and G. Zweig, "Maximum entropy model for punctuation annotation from speech," in *ICSLP 2002*, Denver, CO, USA, 2002.

[6] J. Kolár and L. Lamel, "Development and evaluation of automatic punctuation for French and English speech-to-text," in *Interspeech 2012*, Portland, OR, USA, 2012.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[8] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP 2013*, Vancouver, BC, Canada, 2013, pp. 6645–6649.

[9] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *SLT 2014*, South Lake Tahoe, NV, USA, 2014.

[10] T. Alumäe and A. Kitsik, "TSAB – web interface for transcribed speech collections," in *Interspeech 2011*, Florence, Italy, 2011, pp. 3335–3336.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[13] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[14] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *ICASSP 2011*, 2011, pp. 5528–5531.

[15] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *ICASSP 2009*, 2009, pp. 4741–4744.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] H. Christensen, Y. Gotoh, and S. Renals, "Punctuation annotation using statistical prosody models," in *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding*, 2001.

[18] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech 2014*, 2014.

[19] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *ASRU 2011*, 2011, pp. 24–29.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 9, 1986.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[22] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "RNNLM – recurrent neural network language modeling toolkit," *ASRU 2011*, pp. 196–201, 2011.

[23] T. Alumäe, "Recent improvements in Estonian LVCSR," in *SLTU 2014*, Saint Petersburg, Russia, 2014.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. e. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *ASRU 2011*, Dec. 2011.

[25] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

Appendix C

Publication III

Tilk, O. and Alumäe, T. (2016). Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051. DOI: 10.21437/Interspeech.2016-1517

INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration

Ottokar Tilk[1], *Tanel Alumäe*[2]

[1]Institute of Cybernetics, Tallinn University of Technology, Estonia
[2]Raytheon BBN Technologies, Cambridge, MA, USA
ottokar.tilk@phon.ioc.ee, tanel.alumae@phon.ioc.ee

Abstract

Automatic speech recognition systems generally produce unpunctuated text which is difficult to read for humans and degrades the performance of many downstream machine processing tasks. This paper introduces a bidirectional recurrent neural network model with attention mechanism for punctuation restoration in unsegmented text. The model can utilize long contexts in both directions and direct attention where necessary enabling it to outperform previous state-of-the-art on English (IWSLT2011) and Estonian datasets by a large margin.
Index Terms: neural network, punctuation restoration

1. Introduction

Most automatic speech recognition (ASR) systems output an unpunctuated sequence of words. Restoring the punctuation greatly improves the readability of transcripts and increases the effectiveness of subsequent processing, like machine translation, summarization, question answering, sentiment analysis, syntactic parsing and information extraction.

Punctuation restoration and a related task of segmentation or sentence boundary detection have been extensively studied.

Some previous approaches have used textual features only, enabling applications where audio is not available. Various methods have been used, like n-gram models [1], conditional random fields (CRFs) [2, 3], transition-based dependency parsing [4], deep and convolutional neural networks [5]. Some have treated the punctuation restoration as a machine translation task, translating from unpunctuated text to punctuated text [6, 7].

On the other hand, there are methods that rely entirely on prosodic or audio based features, such as pause durations between words, pitch and intensity [8, 9]. For example, a combination of two neural networks has been used, where the first network classifies input as speech or punctuation and the second one predicts the punctuation type [9].

Both approaches have benefits — text based approach does not require audio and has generally shown better results on reference transcripts, while prosody based models are more robust to ASR system errors — but the combination of the two brings further improvements [10, 11]. Pause durations between words have been shown to be particularly helpful when combined with textual features [8, 12]. Approaches for combining textual and prosodic features can be roughly divided into two categories — a single model that utilizes both types of features, and separate models that are combined in various ways.

Single model approach has been used, for example, with maximum entropy model [13, 14, 15, 16], statistical finite state model [8], boosting-based classifier [10] and long short-term memory (LSTM) recurrent neural network [17].

A common way to combine models is to pass the outputs of the textual model along with prosodic features to the main model that makes the final punctuation decision. For example, language model posteriors can be treated as features by a decision tree [18], CRFs [19] or adaptive boosting algorithm [11]. Another option is to use prosodic posteriors as features for a model that combines them with textual features, like in [20] where deep neural network based prosodic model posteriors were used as additional features in a text based CRFs classifier. Prosodic and textual model posteriors can also be interpolated [18, 12, 21, 10] or passed to a third model as features [22].

Combination of a separate textual and prosodic component makes it straightforward to achieve a greater quality textual model, as it is not limited to the availability of corresponding audio and can be separately trained on a much larger amount of text [22]. Single model methods can achieve the same goal through adaptation or 2-stage training, where the model is initially trained on a large text corpus using textual features alone, and then adapted on a smaller corpus where both textual and prosodic features are available [17].

In [23], a multi-pass approach additionally refined a prosody and text based CRFs result, by taking into account the distance from the closest sentence boundary in both directions.

In this work we use two approaches. On the English dataset we use text only, as prosodic features were unavailable to us and the previous best result that we compare with. On the Estonian dataset we use textual features in combination with a prosodic feature. Similarly to the previous best method [17], the only prosodic feature we use is the pause duration between words, but other features can also be easily incorporated into this model. We use a single model that is trained in two stages to maximally utilize both text and prosody. The two-stage approach is similar to the one used in [17] where in the first stage a large written text corpus is utilized for training textual features, and then these features are combined with the pause duration feature in the second stage when the model is trained on a smaller pause annotated corpus. Although some of the previous work (e.g. [6]) reported results on already segmented text, our results are achieved on unsegmented text.

The novelty of our approach is that this is, to the best of our knowledge, the first use of bidirectional recurrent neural networks (BRNN) [24] in combination with an attention mechanism [25] for punctuation restoration in unsegmented text. The source code of the model is publicly available [1].

The next section describes our approach in detail. Section 3 describes training strategies, models, data, metrics and results. Section 4 concludes the paper.

[1]https://github.com/ottokart/punctuator2

http://dx.doi.org/10.21437/Interspeech.2016-1517

2. Method

Our model is a bidirectional recurrent neural network (BRNN) [24] which enables it to make use of unfixed length contexts before and after the current position in text.

In the recurrent layers we use gated recurrent units (GRU) [26] that are well suited for capturing long range dependencies on multiple time scales. These units have similar benefits as LSTM [27] units while being simpler.

We incorporated an attention mechanism [25] into our model to further increase its capacity of finding relevant parts of the context for punctuation decisions. For example the model might focus on words that indicate a question, but may be relatively far from the current word, to nudge the model towards ending the sentence with a question mark instead of a period.

To fuse together the model state at current input word and the output from the attention mechanism we use a late fusion approach [28] adapted from LSTM to GRU. This allows the attention model output to directly interact with the recurrent layer state while not interfering with its memory.

Next we describe in detail how our model processes the inputs to produce the outputs. At time step t the model outputs probabilities for punctuations $\boldsymbol{y}_t$ to be placed between the previous word $\boldsymbol{x}_{t-1}$ and current input word $\boldsymbol{x}_t$. As there is no punctuation before the first word $\boldsymbol{x_1}$, the model predicts punctuations only for words $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$, where $\boldsymbol{x}_T$ is a special *end-of-sequence* token.

The sequence of one-hot encoded input words $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is first processed by a bidirectional layer consisting of two recurrent layers with GRU units, where one recurrent layer processes the sequence in forward direction and the other in reverse direction. Both recurrent layers are preceded by a shared embedding layer with weights $\boldsymbol{W}_e$. The state $\overrightarrow{\boldsymbol{h}}_t$ at time step t of the forward recurrent layer is

$$\overrightarrow{\boldsymbol{h}}_t = GRU(\boldsymbol{x}_t \boldsymbol{W}_e, \overrightarrow{\boldsymbol{h}}_{t-1}) \tag{1}$$

where GRU is the gated recurrent unit activation function as described in [26] with the exception of added biases. We use $tanh$ as the new hidden state nonlinearity ϕ. The state $\overleftarrow{\boldsymbol{h}}_t$ of the reverse recurrent layer is computed similarly except the input word sequence $\boldsymbol{X}$ is processed in reverse order. The bidirectional state $\boldsymbol{h}_t$ is then constructed by concatenating the states of the forward and backward layers at time t:

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t] \tag{2}$$

So this layer learns representations for each input word x_t that depend on both the preceding and following context, hopefully helping the model to better identify question indicating words as this often depends on the context (e. g. "This is *what* I do." vs. "*What* do you do?"). Also, this gives the model more information to determine whether the current word starts a new sentence or not.

The bidirectional layer is followed by a unidirectional GRU layer with an attention mechanism. This layer processes the bidirectional states sequentially and keeps track of the current position in text, while the attention mechanism can focus on relevant bidirectional context aware word representations before and after the current position. The state $\boldsymbol{s}_t$ of the layer

$$\boldsymbol{s}_t = GRU(\boldsymbol{h}_t, \boldsymbol{s}_{t-1}) \tag{3}$$

is late fused with the attention model output $\boldsymbol{a}_t$ which is computed based on the previous state $\boldsymbol{s}_{t-1}$ and bidirectional layer states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$ as described in [25]. The late fused state $\boldsymbol{f}_t$

$$\boldsymbol{f}_t = \boldsymbol{a}_t \boldsymbol{W}_{fa} \circ \sigma(\boldsymbol{a}_t \boldsymbol{W}_{fa} \boldsymbol{W}_{ff} + \boldsymbol{h}_t \boldsymbol{W}_{fh} + \boldsymbol{b}_f) + \boldsymbol{h}_t \tag{4}$$

is fed to the output layer producing the punctuation probabilities $\boldsymbol{y}_t$ at time step t

$$\boldsymbol{y}_t = Softmax(\boldsymbol{f}_t \boldsymbol{W}_y + \boldsymbol{b}_y) \tag{5}$$

The model described above is used for single stage training and as the first stage in two-stage training. Graphical description of the model can be seen in Figure 1.

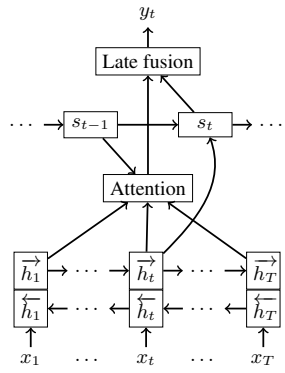


Figure 1: *Description of the model predicting punctuation $\boldsymbol{y}_t$ at time step t for the slot before the current input word $\boldsymbol{x}_t$.*

For two-stage training, to incorporate pause duration and adapt to target domain, the second stage discards the first stage output layer and replaces it with a new recurrent GRU layer

$$\boldsymbol{z}_t = GRU([\boldsymbol{f}_t, p_t], \boldsymbol{z}_{t-1}) \tag{6}$$

which takes the concatenation of the late fusion state $\boldsymbol{f}_t$ and the pause duration p_t before word $\boldsymbol{x}_t$ as input. Vector $\boldsymbol{z}_t$ is passed to a newly initialized output layer, similar to 5. Only the newly added parameters are trained during second stage training while the first stage parameters are kept fixed. This worked better than adapting all the parameters. The reason for that might be that as the second stage training corpus is smaller, it does not contain all the words that are in the model vocabulary. Therefore some word embeddings are not updated while the rest of the model changes, causing these embeddings to become less compatible with the model.

3. Experiments

3.1. Training details

During training the weights are updated using AdaGrad [29] with a learning rate of 0.02. The L_2-norm of the gradient is kept within a threshold of 2 by normalizing it every time this threshold is exceeded [30].

Negative log-likelihood of the punctuation sequence is minimized during training. During testing the punctuation with highest probability according to model output is chosen. We also experimented with giving the previously predicted punctuations as an input to the model and using beam search to find

the best sequence of predictions, but this caused the model to accumulate mistakes and performed worse.

The first stage of two-stage training is finished when the validation perplexity gets worse for the first time. The second stage of two-stage training and single stage training is completed when the validation perplexity has not improved in the last 5 epochs. Weights are initialized according to the normalized initialization from [31] and biases are initialized to zeros. All hidden layers consist of 256 units.

The models are implemented using Theano [32, 33] and trained on GPUs. The input sequence is partitioned into 200 word long slices. Each slice always begins with the first word of a sentence. If a slice ends with an unfinished sentence, then the unfinished sentence is copied to begin the next slice. The output sequence is one element shorter as no punctuation is placed before the first word. Slices are also used during testing, but unlike during training, the sentence boundaries predicted by the model are used. To reduce training time, the slices are shuffled before each epoch and arranged into mini-batches of 128 slices.

The Estonian model has an input word vocabulary of 100K most frequent words in the training corpus, plus the *end-of-sequence* and *out-of-vocabulary* token. The vocabulary of the English model is constructed by taking all words that occur at least twice in the training corpus, resulting in a vocabulary of 27 244 words and the 2 special tokens.

The output vocabulary consists of the predicted punctuations (comma, period and question mark) and a *no punctuation* token. Other punctuation symbols are either mapped to one of the punctuations in our output vocabulary or removed from corpora. For Estonian dataset, exclamation marks, semicolons and colons are mapped to periods and all other punctuation symbols are removed. In the English dataset exclamation marks and semicolons are mapped to periods, while colons and dashes are mapped to commas.

3.2. Models

On the English dataset we use the model described in Figure 1 (T-BRNN). Since the models in [5] used pre-trained word vectors, we also train one T-BRNN model with embeddings initialized to the same pre-trained word vectors [2] (T-BRNN-pre) for comparison.

The Estonian dataset has out-of-domain and pause annotated data available. Therefore we train our model using the two-stage approach — first training on the large out-of-domain corpus and then adapting on the pause annotated corpus. We train the two-stage Estonian model both with (TA-BRNN-p) and without (TA-BRNN) utilizing pause durations. Analogous models (TA-LSTM-p with pauses and TA-LSTM without pauses) were also trained in [17].

The model that holds the previous best result on Estonian has publicly available source code, so we train the first stage part of it on English for comparison (T-LSTM). We used the same hyperparameters for T-LSTM that were used in [17]. Two-stage training requires out-of-domain data which was not used by [5] and would give our models an unfair advantage.

3.3. Datasets

3.3.1. Estonian

The Estonian dataset we use consists of two parts — a 334M word out-of-domain written text (e. g. newspapers and WWW) corpus and a 1M word in-domain pause annotated speech transcripts (broadcast news and conversations, lectures) corpus. The development and test set consist of 27K and 30K words respectively. The best result so far on this dataset was obtained by [17] and the details of the dataset can be found there.

3.3.2. English

Experiments on English are performed on the IWSLT dataset which consists of TED Talks transcripts. The current best result on this dataset was achieved by [5]. We use the same training, development and test set to train and test our models. The training and development set consist of 2.1M and 296K words respectively and come from the IWSLT2012 machine translation track training data. IWSLT2011 reference and ASR test set are used for testing and contain about 13K words each. More detailed description of the dataset can be found in [5].

3.4. Metrics and results

All models are evaluated in terms of per punctuation and overall precision, recall and F_1-score. We also report the overall slot error rate (SER), as F_1-score has been shown to have some undesirable properties [34]. All comparisons in this section are in terms of absolute differences.

On the Estonian test sets (Table 1), it is clear that our newly proposed BRNN models outperform the previous best LSTM based models despite having to deal with an additional type of punctuation. Our best model (TA-BRNN-p) achieves an overall F_1-score improvement by 2.5% on reference text and 1.8% on ASR output, when compared to the the previous best (TA-LSTM-p). SER is reduced by 4.4% and 2.6% on reference and ASR text respectively. The improvements are even larger and the comparison more fair when we map all question marks to periods (Q=P). Detailed metrics show that the TA-BRNN-p model is better than TA-LSTM-p in all aspects except comma restoration precision, but the difference is small. The gap between the text-only models (TA-BRNN and TA-LSTM), that did not use the pause duration information during second stage training, is even bigger — F_1-score improves by $3.8-4.5\%$ and SER by $3.4 - 6.4\%$. The improvements with the text only TA-BRNN model seem to mostly come from its much higher recall for periods (by $18.1 - 21.2\%$ higher than TA-LSTM) without sacrificing precision. As the previous best model (TA-LSTM-p) used only the next word and the preceding context and many commas in Estonian depend on a very local context (the next word), we conclude that the improved period restoration is the benefit of our model's ability to flexibly utilize the entire context in both directions.

The results on English test sets (Table 2) show even larger differences. The overall F_1-score improves by 8.9% on reference text and by 10.5% on ASR output when comparing our T-BRNN model to the best baseline (DNN-A). The best baseline in terms of SER is the DNN model from [5] and the T-BRNN model reduces it by 11.6% on reference text and by 15.5% on ASR output. The T-BRNN model shows improvements in all metrics for all punctuation types. The biggest difference is in the question mark restoration performance, as the models from [5] were unable to restore any question marks thanks to a limited fixed size context (3 words before and 2 words after the slot) that rarely included the question indicating words that often are in the beginning of the sentence. Our newly proposed T-BRNN model and the T-LSTM model from [17] were both able to restore question marks, as their preceding context length is not limited to a fixed size. The T-LSTM

[2] http://nlp.stanford.edu/projects/glove/

Table 1: *Results on Estonian reference transcripts and ASR output test set. T-LSTM-p, TA-LSTM and TA-LSTM-p are the best models from [17], TA-BRNN and TA-BRNN-p are our models, and (Q=P) indicates that question marks have been mapped to periods.*

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		Pr.	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*SER*
Ref.	T-LSTM-p [17]	78.5	63.3	70.1	68.9	59.8	64.0	-	-	-	75.6	62.3	68.3	52.8
	TA-LSTM [17]	74.5	72.2	73.3	62.8	42.9	51.0	-	-	-	71.9	63.9	67.7	52.5
	TA-LSTM-p [17]	**82.3**	69.9	75.6	67.7	76.8	72.0	-	-	-	77.3	71.9	74.5	43.7
	TA-BRNN	75.1	**75.5**	75.3	65.6	64.1	64.8	**63.6**	43.8	51.9	72.5	71.9	72.2	46.1
	TA-BRNN-p	81.6	75.4	**78.4**	72.5	77.0	74.7	59.1	**48.7**	**53.4**	78.6	75.4	77.0	39.3
	TA-BRNN (Q=P)	75.1	**75.5**	75.3	67.4	64.7	66.0	-	-	-	73.0	72.4	72.7	45.6
	TA-BRNN-p (Q=P)	81.6	75.4	**78.4**	**73.8**	**77.3**	**75.5**	-	-	-	**79.2**	**76.0**	**77.6**	**38.7**
ASR	T-LSTM-p [17]	69.9	57.3	63.0	57.3	49.0	52.8	-	-	-	66.2	55.0	60.1	68.5
	TA-LSTM [17]	64.5	64.6	64.6	48.8	32.2	38.8	-	-	-	61.3	55.5	58.2	72.1
	TA-LSTM-p [17]	**71.1**	62.5	66.5	54.9	61.2	57.8	-	-	-	65.7	62.1	63.8	65.5
	TA-BRNN	63.9	**67.9**	65.8	54.0	50.3	52.1	48.8	29.9	37.0	61.3	62.6	62.0	68.7
	TA-BRNN-p	69.1	66.8	**68.0**	59.7	**61.5**	60.6	**51.2**	**31.3**	**38.9**	66.3	64.9	65.6	62.9
	TA-BRNN (Q=P)	63.9	**67.9**	65.8	54.9	50.2	52.4	-	-	-	61.6	62.9	62.2	68.4
	TA-BRNN-p (Q=P)	69.1	66.8	**68.0**	**60.6**	61.1	**60.8**	-	-	-	**66.6**	**65.2**	**65.9**	**62.6**

Table 2: *Results on English reference transcripts and ASR output test set. DNN, DNN-A and CNN-2A are the best models from [5], T-LSTM is first stage model from [17] that we trained on the English dataset, and T-BRNN and T-BRNN-pre are our models.*

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		Pr.	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*SER*
Ref.	DNN [5]	58.2	35.7	44.2	61.6	64.8	63.2	0	0	-	60.3	48.6	53.8	62.9
	DNN-A [5]	48.6	42.4	45.3	59.7	68.3	63.7	0	0	-	54.8	53.6	54.2	66.9
	CNN-2A [5]	48.1	44.5	46.2	57.6	69.0	62.8	0	0	-	53.4	55.0	54.2	68.0
	T-LSTM [17]	49.6	41.4	45.1	60.2	53.4	56.6	57.1	43.5	49.4	55.0	47.2	50.8	74.0
	T-BRNN	64.4	45.2	53.1	72.3	71.5	71.9	67.5	58.7	62.8	68.9	58.1	63.1	51.3
	T-BRNN-pre	**65.5**	**47.1**	**54.8**	**73.3**	**72.5**	**72.9**	**70.7**	**63.0**	**66.7**	**70.0**	**59.7**	**64.4**	**49.7**
ASR	DNN [5]	47.2	32.0	38.1	59.0	60.9	60.0	0	0	-	54.4	45.6	49.6	73.3
	DNN-A [5]	41.0	40.9	40.9	56.2	64.5	60.1	0	0	-	49.2	51.6	50.4	79.2
	CNN-2A [5]	37.3	40.5	38.8	54.6	65.5	59.6	0	0	-	46.4	51.9	49.1	83.6
	T-LSTM [17]	41.8	37.8	39.7	56.4	49.3	52.6	55.6	42.9	48.4	49.1	43.6	46.2	83.7
	T-BRNN	**60.0**	**45.1**	**51.5**	69.7	69.2	69.4	**61.5**	45.7	52.5	65.5	57.0	60.9	57.8
	T-BRNN-pre	59.6	42.9	49.9	**70.7**	**72.0**	**71.4**	60.7	**48.6**	**54.0**	**66.0**	**57.3**	**61.4**	**57.0**

model showed the lowest period restoration scores, indicating that looking further than one word into the following context is important for sentence boundary detection. The T-BRNN model showed improvements despite the fact that DNN, DNN-A and CNN-2A used an external source of information in the form of pre-trained word vectors (trained on 6B tokens). When we use the same word vectors as an initialization for our word embeddings, then we get further improvements and achieve our best result (T-BRNN-pre).

The comparison of the Estonian and English results reveals that comma restoration is a much more difficult task in English than it is in Estonian. This does not come as a surprise, as many commas in Estonian can be restored by following relatively simple rules based on the next word. Although there is a big difference in question mark restoration performance as well, it is hard to make conclusions as they are too rare in both test sets.

To better understand the individual contributions of bidirectionality and attention, we trained additional models on English with either of the components removed. Bidirectionality turned out to be the biggest factor, as removing the forward context caused the performance of all punctuation marks (especially periods and question marks) to drop. Removing attention had much smaller effect, hurting mostly question mark restoration.

4. Conclusions

This paper presented a bidirectional recurrent neural network with attention mechanism for restoring commas, periods and question marks in unsegmented transcribed speech. Both a purely textual approach and an approach combining textual features with prosodic information were used. Experiments on Estonian and English showed improvements for all punctuation types compared to the state-of-the-art. The overall F_1-score was improved by $1.8 - 10.5\%$ absolute and slot error rate was reduced by $2.6 - 15.5\%$. The biggest improvements were achieved when comparing text-only models.

Future research includes the use of a richer set of prosodic features, training an English model on a larger dataset, and exploring joint punctuation and capitalization models.

5. Acknowledgements

This study was supported by the National Program for Estonian Language Technology funded by the Estonian Ministry of Education and Research, and by the European Regional Development Fund (Centre of Excellence in Estonian Studies). The authors would also like to thank Xiaoyin Che and the other authors of [5] for the support with the IWSLT dataset.

6. References

[1] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *ICASSP 2009*, 2009, pp. 4741–4744.

[2] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *EMNLP 2010*, Cambridge, MA, USA, 2010.

[3] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text," in *Interspeech 2013*, Lyon, France, 2013.

[4] D. Zhang, S. Wu, N. Yang, and M. Li, "Punctuation prediction with transition-based parsing." in *ACL (1)*, 2013, pp. 752–760.

[5] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *The 10th International Conference on Language Resources and Evaluation (LREC)*, 2016.

[6] S. Peitz, M. Freitag, A. Mauser, and H. Ney, "Modeling punctuation prediction as machine translation." in *IWSLT*, 2011, pp. 238–245.

[7] E. Cho, J. Niehues, K. Kilgour, and A. Waibel, "Punctuation insertion for real-time spoken language translation," *Proceedings of the Eleventh International Workshop on Spoken Language Translation*, 2015.

[8] H. Christensen, Y. Gotoh, and S. Renals, "Punctuation annotation using statistical prosody models," in *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding*, 2001.

[9] T. Levy, V. Silber-Varod, and A. Moyal, "The effect of pitch, intensity and pause duration in punctuation detection," in *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*. IEEE, 2012, pp. 1–4.

[10] J. Kolář, E. Shriberg, and Y. Liu, "Using prosody for automatic sentence segmentation of multi-party meetings," in *Text, Speech and Dialogue*. Springer, 2006, pp. 629–636.

[11] J. Kolár and L. Lamel, "Development and evaluation of automatic punctuation for French and English speech-to-text," in *Interspeech 2012*, Portland, OR, USA, 2012.

[12] J. Kolář, J. Švec, and J. Psutka, "Automatic punctuation annotation in Czech broadcast news speech," in *SPECOM 2004*, Saint Petersburg, Russia, 2004.

[13] J. Huang and G. Zweig, "Maximum entropy model for punctuation annotation from speech," in *ICSLP 2002*, Denver, CO, USA, 2002.

[14] F. Batista, D. Caseiro, N. Mamede, and I. Trancoso, "Recovering punctuation marks for automatic speech recognition," in *Interspeech 2007*, Antwerp, Belgium, 2007.

[15] ——, "Recovering capitalization and punctuation marks for automatic speech recognition: Case study for portuguese broadcast news," *Speech Communication*, vol. 50, no. 10, pp. 847–862, 2008.

[16] F. Batista, H. Moniz, I. Trancoso, and N. Mamede, "Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 2, pp. 474–485, 2012.

[17] O. Tilk and T. Alumäe, "LSTM for punctuation restoration in speech transcripts," in *Interspeech 2015*, Dresden, Germany, 2015.

[18] A. Stolcke, E. Shriberg, R. A. Bates, M. Ostendorf, D. Hakkani, M. Plauch, G. Tr, and Y. Lu, "Automatic detection of sentence boundaries and disfluencies based on recognized words," in *ICSLP 1998*, Sydney, Australia, 1998.

[19] X. Wang, H. T. Ng, and K. C. Sim, "Dynamic conditional random fields for joint sentence boundary and punctuation prediction." in *INTERSPEECH*, 2012, pp. 1384–1387.

[20] C. Xu, L. Xie, G. Huang, X. Xiao, E. Chng, and H. Li, "A deep neural network approach for sentence boundary detection in broadcast news." in *INTERSPEECH*, 2014, pp. 2887–2891.

[21] E. Matusov, A. Mauser, and H. Ney, "Automatic sentence segmentation and punctuation prediction for spoken language translation." in *IWSLT*. Citeseer, 2006, pp. 158–165.

[22] O. Khomitsevich, P. Chistikov, T. Krivosheeva, N. Epimakhova, and I. Chernykh, "Combining prosodic and lexical classifiers for two-pass punctuation detection in a russian asr system," in *Speech and Computer*. Springer, 2015, pp. 161–169.

[23] M. Hasan, R. Doddipatla, and T. Hain, "Multi-pass sentence-end detection of lecture speech." in *INTERSPEECH*, 2014, pp. 2902–2906.

[24] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR2015, arXiv:1409.0473*, 2015.

[26] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[28] T. Wang and K. Cho, "Larger-context language modelling," *arXiv preprint arXiv:1511.03729*, 2015.

[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[30] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[32] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.

[33] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[34] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel *et al.*, "Performance measures for information extraction," in *Proceedings of DARPA broadcast news workshop*, 1999, pp. 249–252.

Appendix D

Publication IV

Kurimo, M., Enarvi, S., Tilk, O., Varjokallio, M., Mansikkaniemi, A., and Alumäe, T. (2017). Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987, ISSN: 1574-0218, DOI: 10.1007/s10579-016-9336-9

Lang Resources & Evaluation (2017) 51:961–987
DOI 10.1007/s10579-016-9336-9

ORIGINAL PAPER

Modeling under-resourced languages for speech recognition

Mikko Kurimo[1] · Seppo Enarvi[1] · Ottokar Tilk[2] · Matti Varjokallio[1] · André Mansikkaniemi[1] · Tanel Alumäe[2]

Published online: 10 February 2016

Abstract One particular problem in large vocabulary continuous speech recognition for low-resourced languages is finding relevant training data for the statistical language models. Large amount of data is required, because models should estimate the probability for all possible word sequences. For Finnish, Estonian and the other fenno-ugric languages a special problem with the data is the huge amount of different word forms that are common in normal speech. The same problem exists also in other language technology applications such as machine translation, information retrieval, and in some extent also in other morphologically rich languages. In this paper we present methods and evaluations in four recent language modeling topics: selecting conversational data from the Internet, adapting models for foreign words, multi-domain and adapted neural network language modeling, and decoding with subword units. Our evaluations show that the same methods work in more than one language and that they scale down to smaller data resources.

✉ Seppo Enarvi
seppo.enarvi@aalto.fi

Mikko Kurimo
mikko.kurimo@aalto.fi

Ottokar Tilk
ottokar.tilk@phon.ioc.ee

Matti Varjokallio
matti.varjokallio@aalto.fi

André Mansikkaniemi
andre.mansikkaniemi@aalto.fi

Tanel Alumäe
tanel.alumae@phon.ioc.ee

[1] Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland

[2] Institute of Cybernetics, Tallinn University of Technology, Tallinn, Estonia

Keywords Large vocabulary speech recognition · Statistical language modeling · Subword units · Data filtering · Adaptation

1 Introduction

In agglutinative languages, for example Finnish and Estonian, the number of different word forms is huge, because of derivation, inflection and compounding. This is problematic for statistical language modeling that tries to build probabilistic models of word sequences. While modeling the morphology in these languages is complex, modeling the pronunciation of words is rule-based with few exceptions. Thus, splitting words into subwords, such as morphemes or statistical morphs, is a viable and useful tool in applications like automatic speech recognition. However, statistical modeling of morphology, lexicon and word sequences still requires a considerable amount of relevant training data. For under-resourced agglutinative languages, such as variations of Sami and other small fenno-ugric languages, the collection of relevant training data is a significant challenge for language technology development. In this paper we study this resource problem by performing simulations in Finnish and Estonian which include similar morphological properties, but have sufficient resources for carrying out evaluations.

The technical focus of this paper is in large-vocabulary continuous speech recognition (LVCSR) that is essential for automatic processing of dictations, interviews, broadcasts, and all audio-visual recordings. In LVCSR we target on four language modeling topics where we have recently been able to show significant progress: selecting conversational language modeling data from the Internet, adapting pronunciation and language models (LMs) for foreign words, multi-domain and adapted neural network language modeling for improving performance in target topic and style, and decoding with subword lexical units.

For many languages today, large amounts of textual material can be extracted from the World Wide Web. These texts, however, generally provide rather poor match to the targeted style of the language. On the other hand, producing enough accurately transcribed matching training data is expensive. We have faced this problem when developing speech recognition systems for conversational Finnish and Estonian. Huge amounts of Finnish and Estonian data can be crawled from the Internet, but careful filtering is required to obtain a model that matches spontaneous conversations. Several methods have been proposed for selecting segments from an inconsistent collection of texts, so that the selected segments are in some sense similar to in-domain development data (Klakow 2000; Moore and Lewis 2010; Sethy et al. 2006). However, these methods rely on proper development data, but for our Finnish and Estonian tasks there are little carefully transcribed spontaneous conversations available.

A particular problem in lexical modeling is the frequent use of foreign words, which do not follow the same morphological and pronunciation rules as the native words. This becomes a major problem for speech recognition, because a single misrecognized word can severely degrade the modeling of the whole sentence, and the proper names, in particular, are often the most important key words of the

content. In many automatic speech recognition (ASR) applications the correct recognition of foreign words relies on hand-made pronunciation rules that are added to the native lexicon. This is a time-consuming solution. An alternative is to automatically generate pronunciation rules for foreign words. Data-driven grapheme-to-phoneme (G2P) converters are often used for this purpose (Bisani and Ney 2008). Focused pronunciation adaptation for foreign words has been previously implemented by automatically detecting the most likely foreign words with letter n-gram models and then generating pronunciation rules for them with language-specific G2P converters (Maison et al. 2003; Lehecka and Svec 2013). Discriminative pruning of G2P pronunciation variants for foreign proper names has also been applied, to reduce the effect of lexical confusion (Adde and Svendsen 2011).

The state-of-the-art in statistical language modeling has been pushed forward by the application of neural networks (Bengio et al. 2003). Neural network models, projecting word sequences into a continuous space, are capable of modeling more complex dependencies, and improve generalization and discrimination. Neural network language models (NNLMs) have also been shown to be useful when training data is very limited (Gandhe et al. 2014). Recently, the methods to improve performance in targeted speaking styles and topics have improved—starting with weighted sampling (Schwenk and Gauvain 2005) to more recent work in adaptation (Park et al. 2010), multi-domain models (Alumäe 2013; Tilk and Alumäe 2014) and curriculum learning (Shi et al. 2014). We put our focus on multi-domain models and adaptation in this article.

Subword LMs have many advantages in agglutinative languages with limited data resources. A relatively small lexicon can sufficiently cover an almost unlimited number of words, while still producing models that are capable of accurately predicting words. However, in some cases, the system can also produce words that are very rare or even nonsense. To avoid this we have proposed a new decoder (Varjokallio and Kurimo 2014a), that can efficiently build and use a search network of millions of acceptable words. Thus, new words can be easily added whenever there is a need to recognize some important words that do not exist in the training data.

In our work we mainly present LVCSR evaluations in Finnish and Estonian. Although these two are significantly smaller and less resourced than the main languages of the world, we have fairly good benchmarking tasks to evaluate. For the smaller agglutinative languages, such as Northern Sami, we can not provide such evaluations. However, by artificially reducing Finnish and Estonian training data, we can make simulations that may reveal useful properties of the language modeling methods we propose. The evaluation material in both languages can be divided into broadcast news that suffer from large vocabulary and foreign proper names, and conversations that suffer from the small amount of relevant training data.

2 Methods

2.1 Methods for segmenting words into subwords

Most of the methods described below rely on segmenting the vocabulary into subword units, to address the problems originating from the huge number of

different words in Finnish and Estonian. Unless otherwise stated, we have used Morfessor (Creutz and Lagus 2002) for deriving these segmentations.

The selection algorithms presented in Sects. 2.2.2 and 2.2.3 need to estimate models from development data, which is less than 100,000 words. We found a Morfessor model to be problematic for the selection algorithms, because with so little training data Morfessor commonly segments unseen words into single letters that are missing from the LM, which has a significant effect when scoring unseen sentences.

Therefore, in Sects. 2.2.2 and 2.2.3, we created the subwords using the multigram training algorithms from the freely available software package (Varjokallio and Kurimo 2014b), which avoids setting for any fixed segmentation altogether. By training a multigram model (Deligne and Bimbot 1997) using the forward–backward estimation procedure, the segmentation of words into subwords is probabilistic and all segmentation paths are considered in the model. The multigram formulation is also closely related to Markov models. The model may be written as a unigram model, where the probabilities correspond to fractional frequencies as estimated by the forward–backward training. The model can be used for segmentation of unseen words into subwords, and computation of the probability of any sentence, eliminating the OOV issue.

It should be noted that Morfessor segmentations can still significantly benefit automatic speech recognition of agglutinative languages, even when less than 50,000 words of training data is used (Leinonen 2015).

In the decoding experiments in Sect. 3.5, Morfessor was used for the language models trained on the smaller subset. On the larger subset, the subword vocabulary was selected to code the training corpus with high unigram likelihood (Varjokallio et al. 2013). This segmentation approach is suitable for reasonably large text corpora.

2.2 Methods for selecting conversational data from the Internet

When modeling under-resourced languages, Internet is often the first place to look for training data. However, the noisy web data requires careful filtering. Several methods exist for selecting LM training data that matches the targeted style of the language, but their computational cost can be high, and the sparsity of development data may pose difficulties especially with agglutinative languages. Furthermore, conversational Finnish is written down phonetically, meaning that also phonetic variation increases vocabulary size and data sparsity (Enarvi and Kurimo 2013a).

We have developed tools for effectively applying suitable criteria to select useful segments for language modeling from large data sets, when working with only a handful of development data and a morphologically rich language. The source code is available in GitHub.[1] The selection criteria that we have implemented are summarized below. The first two define a score for a text segment, based on which the segments are filtered independently of each other. The third one defines a

[1] https://github.com/senarvi/senarvi-speech/tree/master/filter-text.

criterion for adding a text segment to current selection set: The data is scanned sequentially and each segment is selected if it improves the selection set.

- *devel-lp* A model is estimated from the unfiltered training data, and with a segment removed. The decrease in development data log probability when a segment is removed, is the score of the segment. This is the selection criterion used by Klakow (2000).
- *xe-diff* A model is estimated from the development data, and from the same amount of unfiltered training data. The score of a segment is the difference in cross-entropy given by these two models. This is the selection criterion used by Moore and Lewis (2010).
- *devel-re* A text segment is added to the selection set, if including it reduces relative entropy with respect to the development data. This is the criterion used by Sethy et al. (2006).

The implementation of each filtering criterion is explained below. In practice, when the language is agglutinative, the only way is to build the LMs from subword units, or the high number of out-of-vocabulary (OOV) words makes reliable estimation of the probabilities impossible (Enarvi and Kurimo 2013a). To make the implementations as fast as possible, unigram subword models are used. Limiting to unigrams does not seem to be harmful, since higher-order LMs tend to overlearn small development sets (Klakow 2000).

2.2.1 *Implementation of devel-lp filtering*

The filtering method presented by Klakow (2000) optimizes the perplexity (or equally log probability) of a model computed from the filtered data, on development data. A naive implementation scores each text segment by removing the text segment from the training data, training a language model, and computing the log probability of the development data. This is compared to the log probability given by an LM trained on all training data, and the difference is the score of the text segment. Models are estimated only from the training data, which makes this approach especially suitable for the situation when we have very limited amount of development data. OOV words or subwords are less of a problem when all the models are estimated from a large data set. Consequently, this was the only one of these filtering methods that we applied in Enarvi and Kurimo (2013a).

The naive implementation requires training as many LMs as there are text segments. Even though the computation can be done in parallel, a number of optimizations were needed to make the algorithm scale to tens of millions of text segments. First we note that the log probability given by the LM trained on all training data is constant, so we can equivalently define the score of a text segment as the log probability when a text segment is removed from the training data. The only statistics needed for the computation of unigram probabilities are subword counts. As we only compute probabilities on the development data, we only need the counts of the subwords that exist in the development data, $\{c_1^T \ldots c_N^T\}, C^T$, which are

collected only once. For each text segment, the counts, $\{c_1^S \ldots c_N^S\}, C^S$ are collected and the score of the segment is computed as

$$\sum_{i=1}^{N} \log\left(\frac{c_i^T - c_i^S}{C^T - C^S}\right) c_i^D, \tag{1}$$

where c_i^D is the number of times the subword appears in the development data. Thus the running time of the algorithm is proportional to the number of text segments times the number of unique subwords in the development data.

2.2.2 *Implementation of xe-diff filtering*

In the method proposed by Moore and Lewis (2010), two language models are estimated, one from the development data and another from the same amount of unfiltered training data. The score of a text segment is the difference in cross-entropy given by these two models. The method requires only computation of the two LM probabilities for each text segment. Thus, the running time is proportional to the number of words in the unfiltered training data.

2.2.3 *Implementation of devel-re filtering*

The idea behind the filtering method proposed by Sethy et al. (2006) is to match the distribution of the filtered data with the distribution of the development data in terms of relative entropy. First a language model is estimated from the development data, and the same amount of unfiltered training data is used to initialize a model of the selection set. Then the text segments are processed sequentially. It is computed how much relative entropy would change, with respect to the development data model, if a segment was included in the selection set. If the change is negative, the text segment is included and the selection set model is updated.

We used the revised version of the algorithm that uses skew divergence in place of Kullback–Leibler (KL) divergence (Sethy et al. 2009). Skew divergence contains parameter α, whose value 1 corresponds to KL divergence, and smaller values smooth the maximum-likelihood model of the selection set. We first select the same amount of text as there is in the initial model and then recompute the model from only the selected data.

Sethy et al. present an optimization that runs proportional to the number of words in the unfiltered training data. However, the sequential algorithm itself cannot be parallelized. The authors note that the algorithm is greedy and running it several times with random permutations of the text segments improves the result. They also suggest skipping sentences that have already been included in more than two passes, in order to gain new data faster. We did not enforce that requirement, enabling us to run multiple passes simultaneously. It should be noted that also the generation of a random permutation can be time consuming and I/O intensive, especially when the data set is too large to be loaded into memory, and multiple parallel processes access the same data.

2.3 Methods for adapting models for foreign words

In ASR applications the correct recognition of foreign proper names (FPNs) is a difficult challenge. The problem of recognizing foreign words is especially a problem for smaller languages where influence from other languages is bigger and FPN occurrence more frequent. For Finnish subword-based ASR, foreign names constitute one of the largest error sources (Hirsimäki and Kurimo 2009).

The challenge in recognizing foreign names stems from a combination of many factors. The most obvious is pronunciation modeling. Pronunciation rules that cover native words usually give unreliable results for foreign words. Foreign names are often rare and topic-specific. Background LMs usually give unreliable estimates for FPNs. A third factor that is quite specific to subword LMs is oversegmentation (base form of the word is split into many different parts). Oversegmentation of foreign words complicates the mapping of non-standard pronunciation rules to separate subword units.

Previously, FPN recognition for Finnish subword-based speech recognition has been improved using a two-pass adaptation framework, as illustrated in Fig. 1 (Mansikkaniemi and Kurimo 2013). Based on first-pass ASR output the language model and lexicon are both adapted in an unsupervised manner. In-domain articles which best match the first-pass output are selected based on latent semantic indexing (LSI). From the selected articles an in-domain LM (P_I) is trained and adapted with the background LM (P_B). In this work linear interpolation is used with a fixed interpolation weight (Eq. 2, $\lambda = 0.1$).

$$P_{adapt}(w|h) = \lambda P_I(w|h) + (1 - \lambda)P_B(w|h) \tag{2}$$

Lexicon adaptation is performed by first screening for foreign word candidates in the in-domain texts. All words starting with an uppercase letter are selected as foreign word candidates. From the candidate list, the most likely foreign words are chosen using the product of two factors, letter n-gram perplexity $ppl(word)$ and topic similarity $sim(word)$, as a score (Eq. 3). $ppl(word)$ is the perplexity given by letter n-gram model estimated from a native word list collected beforehand, on word *word* in the in-domain article. $sim(word)$ is defined as the cosine similarity between the first-pass output and the article where *word* occurs.

$$score(word) = ppl(word) * sim(word) \tag{3}$$

The most likely foreign names (with the highest score) are selected and added to the vocabulary. Adapted pronunciation rules for each FPN are generated using a data-driven G2P model (Bisani and Ney 2008). Optionally subword restoration is applied for oversegmented FPN candidate words.

In this work we study how well this adaptation framework can be transferred from Finnish to a related language, Estonian. The phoneme sets of the two languages are quite similar. This gives the option of sharing the foreign word G2P model. The original G2P model was trained on 2000 foreign names retrieved from a Finnish text corpus. The hand-crafted pronunciation rules were made with a Finnish

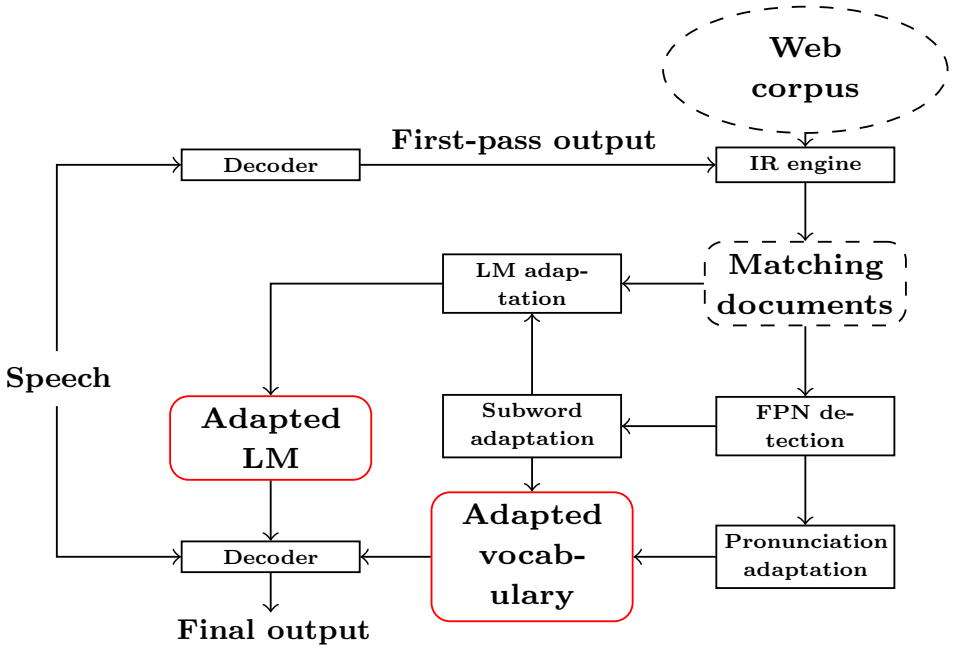


Fig. 1 Adaptation framework for foreign proper name adaptation. Adapted LM and vocabulary are used in second-pass recognition

phoneme set and Finnish speakers in mind. The pronunciation rules generated from the G2P model can with some minor modifications be converted to an Estonian phoneme set.

A problem with G2P generated pronunciation variants when trying to improve FPN recognition is that many of the variants actually degrade the recognition of native words. In combination with the adaptation framework, we will also evaluate a lattice-based discriminative pronunciation pruning method (Enarvi and Kurimo 2013b). The pruning tools are available in GitHub.[2] The algorithm removes those FPN pronunciation variants from the final adapted dictionary that have a negative effect on the total word error rate. Pronunciation variants that have a positive effect on recognition are used to retrain the G2P model by appending them to the foreign word lexicon. This discriminative training procedure is iterated a number of times on the development set before a final G2P model and a list of harmful pronunciations is obtained. The updated G2P model and the list of harmful pronunciations are then used on the evaluation set.

To the authors' knowledge no previous work has used this type of lattice-based discriminative pronunciation pruning for both excluding harmful pronunciation variants and re-training the G2P model with beneficial pronunciation variants.

[2] https://github.com/senarvi/senarvi-speech/tree/master/filter-dictionary.

2.4 Methods for multi-domain and adapted neural network language modeling

When developing a LM for a specific domain it is often the case that the amount of available in-domain data (the data belonging to the target domain) is not sufficient for a good model. This is even more of a problem when dealing with under-resourced languages. The scarcity of in-domain data makes it necessary to include out-of-domain sources in the training of the LM. Usually the amount of available out-of-domain data is much bigger than in-domain data. Therefore the LM needs to favour the in-domain data somehow to perform well in the target domain.

NNLMs (Bengio et al. 2003) can achieve this goal in several ways:

- *Weighted sampling* During training the in-domain data is sampled with higher probability than out-of-domain data [e.g. use all in-domain data and only a random subset of out-of-domain data in each epoch (Schwenk and Gauvain 2005)].
- *Curriculum learning* The order in which the training data is presented to the network is planned in such a way that more general samples are seen in the beginning of the training while domain-specific samples are kept towards the end of the training so they have more influence on the final model (Shi et al. 2014).
- *Adaptation* After training the model on out-of-domain data it is adapted for the in-domain data. The adaptation can be done, for example, by adding an adaptation layer and training it on in-domain data while keeping the other parameters fixed (Park et al. 2010).
- *Multi-domain models* Most parameters are shared between domains to allow exploiting the inter-domain similarities. A tiny fraction of parameters is reserved to be domain-specific and is switched according to the active domain to take into account the domain-specific differences (Alumäe 2013; Tilk and Alumäe 2014). Unlike with adaptation, the domain-specific and general parameters are trained jointly and the same model can be used in all domains.

In this article we use the adaptation and multi-domain approaches.

For multi-domain approach we use a simplified version of the multi-domain NNLM from Alumäe (2013). The architecture of our model is shown in Fig. 2. It

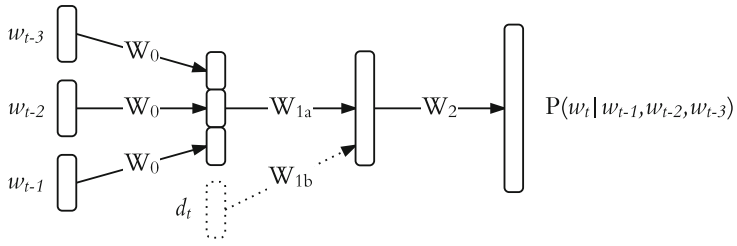


Fig. 2 Description of the NNLM architecture. *Dotted lines* stress the parts of the network that are characteristic only to the multi-domain and adapted models. The inputs (context word indices w_{t-1}, w_{t-2}, w_{t-3} and the domain index d_t) are one-of-N encoded vectors

differs from the architecture described in Alumäe (2013) by omitting the extra linear adaptation layer and applying the multiplicative adaptation factors directly to the pre-activation signal of the hidden layer rectified linear units (ReLU). The hidden layer activations are computed as shown in Eq. 4 where y_0 and y_1 are projection and hidden layer activations respectively, W_{1a} and b_{1a} are hidden layer and W_{1b} and b_{1b} are domain adaptation weights and biases respectively. W_{1b} consists of domain-specific row-vectors (domain vectors) while b_{1b} is shared across domains. To prevent the adaptation factors from shrinking the inputs to ReLU from the start of training, the weights W_{1b} or bias b_{1b} can be initialized to ones (we used the latter in our experiments).

$$y_1 = ReLU(y_0 W_{1a} \circ (d_t W_{1b} + b_{1b}) + b_{1a}) \tag{4}$$

This kind of hidden layer enables each domain to influence the structure of sparsity in the output layer inputs (i.e. which hidden layer units are more or less likely to be exactly zero for each domain) in addition to modulating the nonzero outputs. One can consider the NNLM as a log-linear model on top of an automatically learned feature vector obtained by transforming the input through nonlinear transformations in lower layers as in Seide et al. (2011). In this perspective the multi-domain model can influence the relevance of the log-linear model input features in the context of different domains. Our experience shows that the simplified model performs just as well or even marginally better than the original one with an additional layer.

The multi-domain model requires the availability of in-domain data in the training set. With limited-resource domains it is possible that there is not enough target domain data for separate training, validation and test set. This means that there might be no in-domain data left for the training phase. We propose an adaptation approach which uses exactly the same model architecture as the multi-domain model to overcome this problem. The advantage of using the multi-domain architecture for adaptation is its resistance to overfitting due to the very small amount of domain specific parameters that need to be trained on the target domain data. The amount of domain-specific parameters is limited to a single vector with a number of elements equal to the hidden layer size (usually several hundred or thousand), which is tiny compared to the total amount of parameters in the network (usually in millions). Thus, the training error on validation data gives a good estimate of the performance on unseen data and all the available in-domain data (except the test data) can be used for adaptation.

The adaptation procedure is as follows:

1. Train a general model on out-of-domain training data using the in-domain validation data for early stopping and hyperparameter selection;
2. After the general model is ready, add the domain-specific parameters W_{1b}, b_{1b} and modify the hidden layer activation according to Eq. 4;
3. Train only the domain-specific parameters added in the previous step on the in-domain validation data until convergence, while keeping the rest of the parameters fixed.

Initially, we believed that to effectively utilize the domain vectors, the network should have a multi-domain architecture from the start and be trained as such on non-target domains. However, the preliminary experiments revealed that this is not true. The adapted model works just as well if all the multi-domain architecture specific elements are added right before training the target domain parameters.

This procedure raised a question whether the multi-domain model can also be improved by combining all the in-domain data from both training and validation set and using it to fine-tune the target domain vector as a final step of training. Unfortunately, our preliminary experiments showed that this does not significantly improve the perplexity of the test set.

2.5 Methods for decoding with subword units

The normal approach to language modeling in ASR is to train n-gram LMs over sequences of words. For morphologically rich languages this is often problematic, because the number of OOV words may be high. This is especially the case for less-resourced languages, as considered here. Thus, words are not necessarily the best units for language modeling. By training the n-gram models over sequences of subwords, it is possible to assign probabilities to previously unseen word forms. In our final task we compare different combinations of lexical units and decoders.

A common approach to LVCSR decoding is the dynamic token-passing search (Young et al. 1989), where tokens are propagated in a graph containing paths for the allowed recognition output with the corresponding Hidden-Markov-Model (HMM) state sequences. A token contains at least the accumulated likelihood scores, information about the current n-gram state and the recognition history. Many standard techniques (Ney and Ortmanns 2000) like hypothesis recombination, beam pruning and LM lookahead are needed to make the search efficient. Cross-word pronunciation modeling (Sixtus and Ney 2002) is also important for the speech recognition accuracy in tasks dealing with continuous speech. In Fig. 3, the first

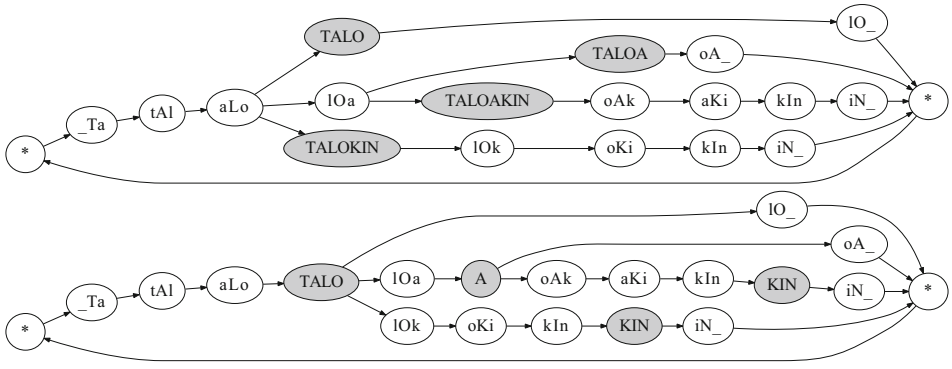


Fig. 3 Example decoding graphs for word n-grams (*above*) and subword n-grams (*below*), for the same 4-word recognition vocabulary. *Grey nodes* depict the n-gram identifiers

Table 1 Finnish speech data sets

Data set	Words	Hours
fi-std-train	131,005	31.4
fi-conv-train	200,415	15.2
fi-conv-eval	6268	0.73
fi-news-dev	35,439	5.38
fi-news-eval	37,196	5.58

Table 2 Estonian speech data sets

Data set	Words	Hours
ee-conv-train	1,251,638	165
ee-conv-eval	25,942	2.90
ee-news-dev	15,961	2.13
ee-news-eval	15,335	2.03

graph is a conceptual example of a standard word decoder utilizing triphone HMMs and word n-grams. Silence and cross-word modeling is omitted from the image.

In the case of subword n-grams, the same search principles may be applied, but the graph should be constructed differently. Here we consider subword decoders, which are general in the sense, that arbitrary segmentations of words to subwords are allowed. With subword n-grams, it is possible to allow all possible concatenations of subwords (Pylkkönen 2005), which enables unlimited recognition vocabulary, as all word forms may be created by concatenating the subwords. The requirement for this construction is that the pronunciation of each subword is defined. For the languages considered here, the pronunciation may be easily derived from the grapheme form of the subword.

Another recently suggested possibility is to use subword n-grams, but still restrict the recognition vocabulary to the desired set of words (Varjokallio and Kurimo 2014a). In Fig. 3, the second graph is a conceptual example of a decoding graph, which is constructed in this way. As also in this case the n-gram model has probabilities for all word forms, unseen words may be segmented with the n-gram model, and the corresponding paths added to the graph. This opens up new possibilities for augmenting and adapting the vocabulary, especially in cases, when the training data does not cover enough word forms. For analysis purposes, the recognition performance of the word n-gram and the subword n-gram estimates may be compared for the same recognition vocabulary. This is useful in assessing, whether the improvement in using subwords models is caused by the better n-gram estimates or the reduced OOV rate.

3 Experiments

3.1 Data

The speech data sets used in our experiments are listed in Tables 1 and 2. Finnish acoustic models for all experiments (except the conversational speech experiment) were trained on the Finnish Speecon database (Iskra et al. 2002), from which 31 h of clean dictated wideband speech from 310 speakers (*fi-std-train*) was used for training. Estonian acoustic models for the conversational speech and neural network language modeling experiments were trained on the full *ee-conv-train* set. It consists of a small amount of spontaneous Estonian conversations, but mostly less spontaneous radio broadcasts and lecture recordings. Estonian acoustic models for the foreign proper name adaptation and subword decoding experiments were trained on a 30 h subset of the *ee-conv-train* set, consisting of only broadcast news recordings.

Finnish conversational speech experiments were carried out on data collected at Aalto University by recording and transcribing pair-wise conversations between students. Finnish acoustic models for web text filtering experiments were trained on the *fi-conv-train* set. It consists of student conversations, transcribed radio shows, FinDialogue part of the FinINTAS (Lennes 2009) corpus, and free spontaneous speech from Finnish SPEECON (Iskra et al. 2002) corpus. The extent to which the speech is spontaneous varies between the recordings, as well as the dialect and style. The evaluation set *fi-conv-eval* consists of transcribed radio conversations and student conversations from unseen speakers. *ee-conv-eval* consists of transcribed conversations from the Phonetic Corpus of Estonian Spontaneous Speech.[3]

Text data sets are listed in Tables 3 and 4. Training data for conversational LMs were crawled from four Estonian conversation sites (*ee-web-1* to *ee-web-4*) and six Finnish sites (*fi-web-1* to *fi-web-6*). These sites contain active discussions in various topics, such as technology, sports, relationships, and culture. The most important tool we have used is the Python library Scrapy. The web data filtering experiments required two development sets for each language. *ee-conv-dev1* and *ee-conv-dev2* consist of transcripts from the Phonetic Corpus of Estonian Spontaneous Speech. *fi-conv-dev1* and *fi-conv-dev2* contain partly the same data that was used in acoustic model training: student conversations, transcribed radio shows, and FinDialogue.

Foreign proper name adaptation experiments were conducted on broadcast news data. The development and evaluation sets *fi-news-dev* and *fi-news-eval* were used in the Finnish experiment and the sets *ee-news-dev* and *ee-news-eval* in the Estonian experiment. The *fi-general* set from the Finnish Text Collection[4] corpus was used for Finnish baseline LM training. It contains texts from books, magazines and newspapers. For Estonian baseline LM training, the full *ee-newspapers* and *ee-news-train* sets were used, and a random 75 % subset of *ee-webnews*.

NNLM experiments for Finnish were carried out on the development and evaluation sets *fi-news-dev* and *fi-news-eval*, which consist of Finnish broadcast news recordings collected in 2011 and 2012. For training the LMs, three data

[3] http://www.keel.ut.ee/et/foneetikakorpus.

[4] https://research.csc.fi/-/finnish-text-collection.

Table 3 Sizes of finnish text data sets after preprocessing

Data set	Words
fi-web-1	766,918
fi-web-2	1,035,043
fi-web-3	561,489
fi-web-4	25,175,069
fi-web-5	46,207,390
fi-web-6	2,618,084,259
fi-conv-dev1	98,956
fi-conv-dev2	8,853
fi-general	153,535,459
fi-webnews	12,675,262
fi-newswire	31,809,529

Table 4 Sizes of Estonian text data sets after preprocessing

Data set	Words
ee-web-1	28,490,011
ee-web-2	4,189,681
ee-web-3	273,413,272
ee-web-4	30,599,060
ee-conv-dev1	187,436
ee-conv-dev2	21,202
ee-newspapers	20,423,775
ee-webnews	76,235,530
ee-news-train	133,171

sources were used: a random subset of 23 million words from *fi-general*, a corpus of texts from Finnish web news portals (*fi-webnews*), and a corpus of newswire texts from a Finnish news agency STT (*fi-newswire*). The Estonian experiment was based on the development and evaluation sets *ee-news-dev* and *ee-news-eval* that contain broadcast news speech from 2005. For language modeling we used three data sources: newspaper texts (*ee-newspapers*), texts from web news portals (*ee-webnews*) and broadcast news transcripts (*ee-news-train*).

Finnish LMs for subword decoding experiments were trained on two subsets from *fi-general*. The larger subset contained 50M word tokens with 2.2M distinct word types and the smaller 10M word tokens with 850k word types. Estonian LMs were trained on the *ee-webnews*, *ee-newspapers* and *ee-news-train* data sets. A larger model was trained on all the training data of around 80M words with 1.6M distinct word types and a smaller model from a 10M word subset with 550k word types.

3.2 Experiments in selecting conversational data from the Internet

In this section we experiment how the most important filtering criteria perform when filtering large amounts of Internet data, when there is only very little in-domain development data available. Our motivation has been development of automatic speech recognition for conversational Finnish and Estonian. We have a small amount of transcribed Finnish and Estonian conversations that are enough for development and evaluation. For LM training data we crawled large amounts of multi-domain data from Internet conversation sites. The segments used as the unit of filtering are conversation site messages.

For the baseline experiments, the sizes of the largest data sets were limited by random selection. In total the number of words in Finnish training data was reduced to 9.9 % and in Estonian data to 49 % of the original. *devel-lp* and *xe-diff* methods define a score for each text segment. The filtering threshold is optimized to minimize the perplexity of a bigram subword model on the second development set (*fi-conv-dev2* or *ee-conv-dev2*). *devel-re* does not define a score for each segment. Instead, whether a segment is included depends on what has been included earlier. We found running multiple passes with random permutations of the input text segments to be crucial for collecting enough data. The number of passes is limited by the high computational cost. We ran 100 passes, but also tried using data from only so many passes that unigram subword model perplexity on the second development set was minimized. We selected the value 0.975 for the smoothing parameter α, based on observations of the original author (Sethy et al. 2009), without trying to optimize the value.

Filtering was performed, and the filtering threshold and the number of passes was optimized, on each data set (conversation site) separately. However, sets *fi-web-1* to *fi-web-3* were pooled together during filtering, and the set *fi-web-6* was split into 48 parts during *devel-lp* and *xe-diff* filtering.

The experiments were carried out using Aalto ASR system (Hirsimäki et al. 2009) and GMM-HMM-based acoustic models. Language models were 4-gram word models interpolated from models of individual data sets. The vocabulary was created after filtering by selecting 200,000 top words based on weighted word counts in order to maximize the likelihood of the combined development data. The number of n-grams in every LM was reduced by pruning all n-grams whose removal caused less than 5×10^{-10} increase in training data perplexity.

3.2.1 Results

Results for web text filtering are shown in Table 5. Large phonetic variation in conversational Finnish creates challenges when measuring recognition accuracy. As most of the words can be pronounced in several slightly different ways, and the words are written out as they are pronounced, it would be harsh to compare recognition against the verbatim phonetic transcription. Thus word forms that are simply phonetic variation have been added as alternatives in the reference transcriptions.

Table 5 Filtered data sizes and speech recognition results. The best results in terms of WER are in bold type

Algorithm	Finnish		Estonian	
	Words	WER (%)	Words	WER (%)
Baseline	266M	55.6	167M	53.4
devel-lp	192M	54.3	82.2M	**52.7**
xe-diff	169M	54.4	38.9M	53.2
devel-re [passes = 1]	5.08M	57.9	13.1M	54.4
devel-re [passes = 50]	53.9M	54.6	93.8M	53.1
devel-re [passes = 100]	79.5M	54.2	125M	53.1
devel-re [optimized]	75.9M	**54.1**	117M	53.1

devel-re selection resulted in the smallest data size. The amount of data that will be selected depends on the size of the development set. The small development set used in these experiments caused only a minimal amount of data to be selected during the first *devel-re* pass, resulting in poor word error rate. Combining selected data from 100 passes improved word error rate to 54.2 % with Finnish data. The other methods gave very similar results in terms of WER, but more than double the amount of data. However, running 100 passes was computationally very demanding.

Optimizing the number of passes of *devel-re* filtering, in terms of perplexity on held-out development data, gave still a slight improvement. The resulting 54.1 % WER is good, given that only web data was used to build the LM. In our previous state-of-the-art of conversational Finnish ASR, we obtained 57.5 % WER using only web data, and 55.6 % when combined with other corpora, while using only other than web data WER was 59.8 % (Enarvi and Kurimo 2013a). One can conclude that significant improvement can be gained by using web data, in the absence of accurately transcribed conversational corpora. However, in this paper we have also used better acoustic models.

Overall, filtering Estonian data did not improve speech recognition compared to the baseline as much as with Finnish data. The best result, 52.7 % WER, was given by *devel-lp* filtering. Compared to the Finnish language results, the advantage to the other methods was surprisingly clear. *devel-re* method gained new data faster than in the Finnish language experiments, probably due to the larger development set, and as many passes were not needed. We are not aware of any earlier research on recognition of spontaneous Estonian conversations.

3.3 Experiments in adapting models for foreign words

Foreign proper name adaptation experiments are conducted with the adaptation framework described in the methods section (Fig. 1). The occurrence of foreign names in the data sets is of importance since we are focusing adaptation efforts on improving their recognition. For Finnish, FPN rate is 4.3 % for the development set (*fi-news-dev*) and 3.5 % for the evaluation set (*fi-news-eval*). For Estonian, FPN rate

is 1.6 % for the development set (*ee-news-dev*) and 1.7 % for the evaluation set (*ee-news-eval*).

Experiments are run on the Aalto ASR system (Hirsimäki et al. 2009) and GMM-HMM-based acoustic models. For Finnish, a Kneser–Ney smoothed varigram LM (n = 12) with a 45k subword lexicon was trained on the LM training data using variKN language modeling toolkit (Siivola et al. 2007) and Morfessor (Creutz and Lagus 2002). A letter bigram model was trained on the same LM training data for the foreign name detection algorithm.

A subword-based baseline LM for Estonian was trained, similarly to Finnish using Morfessor and variKN toolkit. The resulting model was a Kneser–Ney smoothed varigram LM (n = 8) with a 40k subword lexicon. A letter bigram model for foreign name detection was trained on a word list extracted from the LM training data.

First set of experiments are run with the baseline LMs to retrieve the first-pass ASR output. After that unsupervised LM adaptation experiments are run. The background LM is adapted with 6000 of the best matching articles compared to the ASR output. The retrieval corpus is a collection of articles retrieved from the Web. The Finnish retrieval corpus consists of 44,000 articles (*fi-webnews*). The Estonian retrieval corpus consists of 80,000 articles (25 % subset of *ee-webnews*).

In the third adaptation layer we apply vocabulary adaptation. Foreign proper name candidates are selected based on the letter-gram perplexity and cosine similarity score. A threshold is set so that only 30 % of the best scoring FPN candidates are selected for adaptation. Furthermore an additional constraint is set so that the number of new words added can not exceed 4 % of the original vocabulary size. Four new pronunciation rules are generated for each selected FPN candidate and added to the lexicon. The pronunciation rules are generated with a data-driven G2P model which has been trained on 2000 foreign names found in Finnish texts. The same G2P model is used for both Finnish and Estonian. Subword restoration is applied on oversegmented FPN candidate words to enable one-to-one mapping between pronunciation rule and vocabulary unit.

In the final adaptation layer we implement discriminative pronunciation pruning based on the ASR output lattices when using the adapted LM and lexicon. Harmful FPN pronunciation variants that degrade overall recognition accuracy by five word errors or more are excluded in the next run. Beneficial FPN pronunciation variants that decrease word error by one word or more are added to the 2000 word foreign name lexicon. A new G2P model is re-trained with the updated lexicon. This procedure is iterated a couple of times on the development set before get a final list of harmful pronunciation variants and an updated G2P model which are then used on the evaluation set.

3.3.1 Results

Results of the FPN adaptation experiments are presented in Table 6. Performance is measured in average word error rate (WER) and foreign proper name error rate (FER).

Table 6 FPN adaptation results for Finnish and Estonian. Baseline results are followed by results for unsupervised LM adaptation (Adapted LM), combination of unsupervised LM and vocabulary adaptation (Adapted LM + VOC), and iterations of discriminative pronunciation pruning (Adapted LM + VOC [pruned, iter = x]). On the evaluation sets discriminative pronunciation pruning is tested with the pruning data and models obtained after the third iteration on the development set (Adapted LM + VOC [pruned, dev. iter = 3])

Adaptation	*fi-news-dev*		*ee-news-dev*	
	WER (%)	FER (%)	WER (%)	FER (%)
Baseline	29.6	73.5	19.2	51.8
Adapted LM	28.6	66.4	18.9	51.4
Adapted LM + VOC	28.6	61.8	19.5	50.7
Adapted LM + VOC [pruned, iter = 1]	28.5	60.6	19.4	50.0
Adapted LM + VOC [pruned, iter = 2]	28.4	60.6	19.4	49.3
Adapted LM + VOC [pruned, iter = 3]	28.4	60.5	19.4	49.3
Baseline	30.5	71.6	19.6	49.3
Adapted LM	29.7	64.8	19.2	47.1
Adapted LM + VOC	29.6	60.0	19.5	46.0
Adapted LM + VOC [pruned, dev. iter = 3]	29.6	59.0	19.4	46.0

First set of experiments were run on the Finnish development set (*fi-news-dev*). Compared to the baseline model, unsupervised LM adaptation reduces average WER with 3 % and FER with 10 %. Vocabulary adaptation (pronunciation and subword adaptation) reduces FER with another 7 % but average WER remains unchanged, compared to only using unsupervised LM adaptation. After three iterations discriminative pronunciation pruning is able to further reduce WER with 1 % and FER with 2 %. It does seem that pronunciation pruning, in excluding some of the most harmful pronunciation variants, is able to correct the misrecognition of some native words.

For the Finnish evaluation set (*fi-news-eval*) results are similar compared to the development set, when applying unsupervised LM and vocabulary adaptation. Average WER is reduced with around 3 % compared to the baseline LM. Vocabulary adaptation reduces FER with 7 % compared to only using unsupervised LM adaptation. Discriminative pronunciation pruning was tested with the list of harmful pronunciation variants and re-trained G2P model obtained after three iterations on the development set. In terms of average WER, which remains unchanged, results are not as good as on the development set. There is probably not enough overlap between harmful pronunciation variants introduced in the development set that are also relevant for the evaluation set. We might see a more significant impact over larger data sets. The re-trained G2P model reduces FER with around 2 %. The change is small but it does indicate that it is possible to improve G2P modeling through discriminative pronunciation pruning on development data.

For the Estonian broadcast news development set, unsupervised LM adaptation reduced average WER with nearly 2 % and FER with under 1 %. Vocabulary

adaptation increases average WER, but reduces FER with over 1 %, compared to using only unsupervised LM adaptation. Discriminative pronunciation pruning does manage to improve recognition of foreign names with almost 3 % but average WER is still higher than compared to only using unsupervised LM adaptation.

Results for the Estonian evaluation set are quite similar to the development set. Unsupervised LM adaptation reduces WER with 2 % and FER with 4 %. Again, vocabulary adaptation degrades recognition of native words. Average WER increases but FER is reduced with 2 %. Discriminative pronunciation pruning (data and models obtained from the development set's third iteration) does lower average WER slightly but FER is not further improved.

There seems to be more acoustic confusion added to Estonian ASR when augmenting the lexicon with G2P generated pronunciation variants. It is not clear whether this is because of the low FPN rate in Estonian speech data or if the Finnish G2P model has negative effects on the recognition of some native Estonian words. Discriminative pronunciation pruning is not able to significantly lessen the effect of lexical confusion.

3.4 Experiments in multi-domain and adapted neural network language modeling

In multi-domain and adapted NNLM experiments we evaluate the models in terms of perplexity (PPL) and WER. The models are evaluated on two broadcast news data sets: a Finnish data set consisting of subwords (morphs) and an Estonian data set consisting of compound-split words. The PPL scores are calculated on their respective lexical units, WER scores are computed on words.

Our baseline LM is a back-off 4-gram model with modified Kneser–Ney discounting constructed over all available training data. Surprisingly, interpolating domain-specific models results in an inferior model.

It has been recently verified that NNLMs perform better than back-off n-gram models on under-resourced languages (Gandhe et al. 2014). One of our goals is to check whether the multi-domain and adapted NNLMs bring additional improvements and what is the relationship between their relative improvement and training set size.

Four experiments are performed on both languages. We start by training all the models on all available text data and continue by halving the training data for each consecutive experiment by taking every second line of the previous data set. NNLM hidden and projection layer size is divided by $\sqrt{2}$ every time the training data is halved. The initial hidden layer size is 500 for Finnish and 1400 for Estonian NNLM; initial projection layer size is 3×100 for Finnish and fixed to 3×128 for Estonian. Both Finnish and Estonian models use a shortlist (Schwenk and Gauvain 2004) of 1024 most frequent units (compound-split words or subwords respectively) plus an additional end of sentence token. The input vocabulary consists of 199,861 most frequent compound-split words and 50,410 most frequent subwords for Finnish and Estonian data set respectively. Both input vocabularies contain an additional token for the beginning of sentence and unknown units. When interpolating the n-gram and NNLM model outputs we use an equal weight of

0.5 for both models. Out-of-shortlist units are evaluated only by the n-gram model. All NNLMs are trained with backpropagation and mini-batch stochastic gradient descent using batch size of 200 samples and learning rate of 0.1 until the best model according to validation perplexity is not within the last 5 epochs. We use our NNLM adaptation method on Finnish data set, because there we have no in-domain training data. Estonian data set has in-domain training data, so we use the multi-domain NNLM there.

In speech recognition experiments recognition lattices were generated using systems based on the Kaldi toolkit (Povey et al. 2011), and the lattices were rescored using the NNLMs. Finnish acoustic models are triphones, built using fMLLR-based speaker-adaptive training (SAT) and optimized using the boosted MMI criterion (Povey et al. 2008). Lattices are obtained after two decoding passes: first pass uses speaker-independent models, and the second pass fMLLR-transformed features with SAT-based models. Estonian acoustic models are hybrid deep neural networks based hidden Markov models (DNN-HMMs) that use speaker identity vectors (i-vectors) as additional input features to the DNNs in parallel with the regular acoustic features, thus performing unsupervised transcript-free speaker adaptation (Saon et al. 2013). The output hypotheses of the speech recognition systems consist of subword units for Finnish and compounds-split words for Estonian. These were converted to word hypotheses using a hidden event LM that treats a word break (for Finnish) or an inter-compound unit (for Estonian) as a hidden word that needs to be recovered. More details about the Estonian system are available in Alumäe (2014).

3.4.1 Results

The results of PPL and WER evaluations on the test set can be seen in Table 7. All NNLMs consistently outperform back-off n-gram models in PPL and WER. Utilizing NNLMs in addition to n-gram models gives a similar effect as using about twice as much training data: the PPL improves 7.1–17.5 % relative, statistically significant WER improvement is about 2.1–4.9 % relative. The type of lexical units used in vocabulary and baseline WER (largely determined by the acoustic model quality) don't seem to affect the relative WER improvement brought by NNLMs. Both, the multi-domain and adapted, NNLMs consistently beat the simple NNLM in PPL evaluation (0.6–7.1 % relative). Unfortunately this makes no significant difference in WER for neither case. This holds true for all languages and training set sizes we tested.

The small PPL gap and no significant WER improvement between the simple and multi-domain NNLM architecture seems to indicate that the single static domain vector has too little capacity to alter the model sufficiently to reflect all the domain differences. This problem can be solved by either reducing the domain sizes—by clustering them into subdomains for example—or by using adaptation with more capacity and influence over the model.

Table 7 LM test set PPL and WER with different sized training sets. Comparison with the n-gram baseline in parentheses. *a-nnlm* is the adapted and *md-nnlm* is the multi-domain NNLM

		1	1/2	1/4	1/8
Finnish					
PPL	n-gram	197	222	256	298
	nnlm + n-gram	183 (−7.1%)	205 (−7.7%)	236 (−7.8%)	274 (−8.1%)
	a-nnlm + n-gram	177 (−10.2%)	200 (−9.9%)	230 (−10.2%)	268 (−10.1%)
WER	n-gram	33.3	34.0	34.9	35.7
	nnlm + n-gram	32.3 (−3.0%)	32.9 (−3.2%)	33.4 (−4.3%)	34.3 (−3.9%)
	a-nnlm + n-gram	32.4 (−2.7%)	32.8 (−3.5%)	33.4 (−4.3%)	34.3 (−3.9%)
Estonian					
PPL	n-gram	223	252	301	366
	nnlm + n-gram	198 (−11.2%)	216 (−14.3%)	257 (−14.6%)	315 (−13.9%)
	md-nnlm + n-gram	184 (−17.5%)	208 (−17.5%)	250 (−16.9%)	313 (−14.5%)
WER	n-gram	9.2	9.6	10.3	10.8
	nnlm + n-gram	9.0 (−2.2%)	9.4 (−2.1%)	9.8 (−4.9%)	10.3 (−4.6%)
	md-nnlm + n-gram	9.0 (−2.2%)	9.4 (−2.1%)	9.9 (−3.9%)	10.3 (−4.6%)

3.5 Experiments in decoding with subword units

In this section we experiment with different combinations of lexical units and decoders. N-gram LMs used modified Kneser–Ney smoothing and were trained using the VariKN package (Siivola et al. 2007). Maximum order of the n-grams was 3 for word n-grams and 6 for subword n-grams. Relatively large n-gram models with respect to the corpus sizes were used in all the experiments. Word error rates for the models trained on the larger training corpora may be found in Table 8 and for the smaller training corpora in Table 9.

The first observation from the results is that effectively very large vocabularies are needed to obtain good ASR performance on the broadcast news task for both languages, irrespective of the way of modeling. If more was known about the topics to be recognized, more limited vocabularies could be utilized. Accurate topic modelling, however, would likely require more resources than assumed to be available here. The results also show, that the standard dynamic token-passing decoding can effectively operate with very large vocabularies, if care is taken in the implementation (Soltau and Saon 2009; Varjokallio and Kurimo 2014a).

In terms of error rates, including all the word forms from the LM training data to the vocabulary seems to give reasonable initial results. In the Finnish experiments, word n-grams and subword n-grams performed equally well with these very large vocabularies in both the settings. The OOV-rates were still 3.2 and 5.3 %, indicating some mismatch between the training corpus and the recognition task. In the Estonian experiments, the subword n-grams outperformed the word n-grams with

the same vocabulary in both the settings. It thus seems, that subword n-grams provide better probability estimates in some cases. The OOV-rates in the Estonian experiments were 1.2 and 2.5 %.

We also experimented with a subword decoder, which enables an unlimited recognition vocabulary and did simulated experiments, where the recognition vocabulary was augmented by the remaining OOV words and in the smaller corpus setting using the vocabulary from the larger corpus instead. The words were segmented using the n-gram model and added to the decoding graph. The subword n-gram model was not modified.

In the large corpus setting, the relative error rate reductions for the unlimited recognition vocabulary were 2.8 and 3.2 %, compared to the best restricted vocabulary recognizer. The corresponding numbers for the closed vocabulary experiment were 3.4 and 4.5 %. The results show, that the OOV words were still causing many recognition errors. In this case opting for unlimited vocabulary recognition was quite effective in bridging the gap between the initial and the closed vocabulary.

In the small corpus setting, the relative improvements for unlimited vocabulary recognition were 4.5 % for Finnish and 5.3 % for Estonian. By using the vocabulary from the large corpus, the corresponding results were 3.5 % for Finnish and 4.8 % for Estonian. Adding the remaining OOV-words further improved WER by 3.5 and 3.9 %. In this setting, it may be seen that the OOV-rate had quite a big impact on the recognition rates. Also, the difference between the unlimited and the closed vocabulary results increased, indicating that the quality of the n-gram estimates started to suffer.

Table 8 Word error rates for the models trained on the larger training corpora

Units	Finnish		Estonian	
	Vocabulary size	WER (%)	Vocabulary size	WER (%)
Words	2.2M	32.1	1.6M	16.2
Subwords	2.2M	32.1	1.6M	15.6
Subwords	–	31.2	–	15.1
Subwords	2.2M + OOV	31.0	1.6M + OOV	14.9

Table 9 Word error rates for the models trained on the smaller training corpora

Units	Finnish		Estonian	
	Vocabulary size	WER (%)	Vocabulary size	WER (%)
Words	850k	35.2	550k	19.6
Subwords	850k	35.2	550k	18.7
Subwords	–	33.6	–	17.7
Subwords	2.2M	34.0	1.6M	17.8
Subwords	2.2M + OOV	32.8	1.6M + OOV	17.1

In unlimited vocabulary recognition, also some non-words will be recognized. This may be an annoyance in some ASR use cases. The rate of the non-words will depend much on the task at hand. The results further show, that a restricted vocabulary which is closed or nearly closed, should give the best recognition results. In this case also non-words will be avoided. The question then becomes, in which cases is this a realistic goal? The subword n-gram decoder with a restricted vocabulary opens some new possibilities towards this end, as the vocabulary may be augmented without having all the word forms in the training text corpus. Other data sources, like dictionaries and morphological analyzers (generators), can be used to enrich the vocabulary. This could be especially helpful for less-resourced languages, for which sufficiently large text corpora are mostly not available. It has been estimated, that with entry generators (Linden 2009), a native linguist may annotate 300–400 new words in an hour to a morphological analyzer lexicon. For the initial lexicon, around 5000 annotated words may suffice. Also in use cases, where the ASR system will be used repeatedly, it may be possible to cover the most important missing words over time.

4 Conclusion

In this work several recently developed language modeling methods were evaluated in LVCSR. The evaluations were performed in two agglutinative languages, Finnish and Estonian. Although language technology in these two languages have not been very widely developed, most of the benchmarking tasks we used are almost directly comparable to previous work. For the smaller agglutinative languages that are extremely under-resourced, such as Northern Sami, proper evaluations are still impossible. However, by verifying the same evaluations in parallel for both Finnish and Estonian, and by artificially reducing the training data, we managed to make simulations that are realistic for less resourced languages. This allows us to conclude how to collect new data and what methods are suitable for languages with a limited amount of language model training data.

The first task we evaluated was LM training data collection. Although training data for planned speech is relatively easy to collect e.g. from news wire, conversational speech pose a more difficult problem. The best training data would be real conversations, but they are expensive to transcribe. However, we managed to demonstrate a reasonable performance by clever filtering of Internet discussion forums. Reducing data size is essential, not only from the perspective of improving LM accuracy, but also because it makes modeling easier. The most compact training set can be obtained by relative entropy minimization based filtering. The vast reduction in data size may enable new approaches to language modeling, such as NNLMs.

The second evaluation was dealing with the pronunciation and language modeling of foreign words. It is very typical for small languages to borrow new words from English and other large languages. However, the pronunciation of these words do not usually follow the same pronunciation rules as native words and the pronunciation used in practice is often unpredictable. Furthermore foreign words are

often topic-specific and poorly estimated by the baseline LM. Our results indicate the we can successfully improve recognition of foreign words with unsupervised LM and vocabulary adaptation. However, generating multiple pronunciation variants for foreign names negatively affects the recognition of some native words. Discriminative pronunciation pruning did improve recognition slightly over the development sets but the pruned models didn't have as much effect on unseen data in the form of the evaluation sets. It is possible that discriminative pronunciation pruning is more effective over larger data sets. We evaluated a shared resource by using a G2P model originally trained for Finnish on Estonian. Results indicate that the model does improve recognition of foreign words in Estonian as well but the added lexical confusion which impacts the recognition of native words seems to be worse than in Finnish. Improving pruning methods and testing over larger data sets need to be done in the future to better understand the feasibility of G2P model sharing between languages.

The results of the third evaluation show that the proposed multi-domain and adapted NNLMs consistently outperform the n-gram baseline and simple NNLMs in terms of PPL. The proposed model provides statistically significant WER improvements compared to the n-gram baseline, but fails to improve upon simple NNLMs. The results appear to be similar in both multi-domain and adaptation modes. Finding better and more clever methods, rather than just more data, to improve the target-domain performance is important for under-resourced languages, because it is not expected that sufficient amount of in-domain data can be collected for any particular topic or style alone. In our future work we plan to address the lack of WER improvements of multi-domain and adapted models over simple NNLMs by exploring sub-domain level multi-domain models and more powerful adaptation methods.

The last evaluation concerned the different combinations of lexical units and decoding approaches. For agglutinative languages, such as Finnish, Estonian and Sami, subword LMs have many advantages. In the broadcast news experiments, n-gram models trained over subwords performed equally well or better than word n-grams with the same recognition vocabulary. Further advantage is that the subword n-grams are able to assign probabilities to unseen word forms. Decoding with unlimited vocabulary improved recognition accuracy for both languages. Using subword n-grams but still opting for a restricted vocabulary is also a viable alternative, which avoids the recognition of non-sense words. We expect that the ability of quickly adding new words for the search network may become useful if there are important OOV words that the system should recognize better. Also, the results indicated, that in the cases where the recognition vocabulary is closed or nearly closed, better results will be reached with a restricted vocabulary. Much depends on the recognition task and the available resources, if this is a realistic goal.

The next step in our project is to gather and build the resources for constructing and evaluating LVCSR in Northern Sami, where all the results of this paper should become useful. The word error rates from conversational Finnish and Estonian speech recognition experiments are still above 50 %. One area where we still clearly need to improve is acoustic modeling. Accurately transcribed spontaneous conversations are hard to find, so we have had to combine data from many small

corpora of varying quality. More intelligent combination of these data sources by model adaptation or neural network models would certainly help, and will be done in the future.

Acknowledgments This work was partially funded by the Estonian Ministry of Education and Research target-financed research theme no. 0140007s12, by the Tallinn University of Technology project Estonian Speech Recognition System for Medical Applications, by the Academy of Finland under the Grant Number 251170 [Finnish Centre of Excellence Program (2012–2017)], and by Finnish Cultural Foundation. We acknowledge the computational resources provided by Aalto Science-IT project.

References

Adde, L., & Svendsen, T. (2011). Pronunciation variation modeling of non-native proper names by discriminative tree search. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4928–4931). Prague, Czech Republic.

Alumäe, T. (2013). Multi-domain neural network language model. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH 2013)* (pp. 2182–2186).

Alumäe, T. (2014). Recent improvements in Estonian LVCSR. In *Spoken Language Technologies for Under-Resourced Languages*. St. Petersburg, Russia.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

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.

Creutz, M., & Lagus, K. (2002). Unsupervised discovery of morphemes. In *Proceedings of the ACL 2002 workshop on morphological and phonological learning, MPL '02*, Vol. 6 (pp. 21–30). Association for Computational Linguistics, Stroudsburg, PA, USA. doi:10.3115/1118647.1118650.

Deligne, S., & Bimbot, F. (1997). Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, *23*(3), 223–241.

Enarvi, S., & Kurimo, M. (2013a). Studies on training text selection for conversational finnish language modeling. In *Proceedings of the 10th International Workshop on Spoken Language Translation (IWSLT 2013)* (pp. 256–263). Heidelberg, Germany.

Enarvi, S., & Kurimo, M. (2013b). A novel discriminative method for pruning pronunciation dictionary entries. In *Proceedings of the 7th International Conference on Speech Technology and Human–Computer Dialogue* (pp. 113–116). Cluj-Napoca, Romania.

Gandhe, A., Metze, F., & Lane, I. (2014). Neural network language models for low resource languages. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*.

Hirsimäki, T., & Kurimo, M. (2009). Analysing recognition errors in unlimited-vocabulary speech recognition. In *Proceedings of the North American Chapter of the Association for Computational Linguistics—Human Language Technologies 2009 Conference (NAACL 2009)* (pp. 193–196). Boulder, Colorado, USA.

Hirsimäki, T., Pylkkönen, J., & Kurimo, M. (2009). Importance of high-order n-gram models in morph-based speech recognition. *IEEE Transactions on Audio, Speech & Language Processing*, *17*(4), 724–732. doi:10.1109/TASL.2008.2012323.

Iskra, D. J., Grosskopf, B., Marasek, K., van den Heuvel, H., Diehl, F., & Kießling, A. (2002). SPEECON—speech databases for consumer devices: Database specification and validation. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*. Canary Islands, Spain.

Klakow, D. (2000). Selecting articles from the language model training corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, Vol. 3 (pp. 1695–1698). IEEE Computer Society. doi:10.1109/ICASSP.2000.862077.

Lehecka, J., & Svec, J. (2013). Improving speech recognition by detecting foreign inclusions and generating pronunciations. *Text, Speech, and Dialogue, Lecture Notes in Computer Science*, *8082*, 295–302.

Leinonen, J. (2015). Automatic speech recognition for human–robot interaction using an under-resourced language. Aalto University, School of Electrical Engineering, Department of Signal Processing and Acoustics, Espoo.

Lennes, M. (2009). Segmental features in spontaneous and read-aloud Finnish. In V. de Silva & R. Ullakonoja (Eds.), *Phonetics of Russian and Finnish. General introduction. Spontaneous and read-aloud speech* (pp. 145–166). Bern: Peter Lang GmbH.

Linden, K. (2009). Entry generation for new words by analogy for morphological lexicons. *Northern European Journal of Language Technology*, *1*, 1–25.

Maison, B., Chen, S., & Cohen, P. S. (2003). Pronunciation modeling for names of foreign origin. In *Proceedings of the 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 429–434).

Mansikkaniemi, A., & Kurimo, M. (2013). Unsupervised topic adaptation for morph-based speech recognition. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH 2013)* (pp. 2693–2697). Lyon, France.

Moore, R. C., & Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers, ACLShort '10* (pp. 220–224). Association for Computational Linguistics, Stroudsburg, PA, USA. http://dl.acm.org/citation.cfm?id=1858842.1858883.

Ney, H., & Ortmanns, S. (2000). Progress in dynamic programming search for LVCSR. *Proceedings of the IEEE*, *88*(8), 1224–1240.

Park, J., Liu, X., Gales, M. J. F., & Woodland, P. C. (2010). Improved neural network based language modelling and adaptation. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (pp. 1041–1044).

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. (2011). The Kaldi speech recognition toolkit. In *Proceedings of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Hilton Waikoloa Village, Big Island, Hawaii: IEEE Signal Processing Society.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., & Visweswariah, K. (2008). Boosted MMI for model and feature-space discriminative training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2008)* (pp. 4057–4060). IEEE.

Pylkkönen, J. (2005). An efficient one-pass decoder for Finnish large vocabulary continuous speech recognition. In *Proceedings of the 2nd Baltic Conference on Human Language Technologies*.

Saon, G., Soltau, H., Nahamoo, D., & Picheny, M. (2013). Speaker adaptation of neural network acoustic models using i-vectors. In *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 55–59). IEEE.

Schwenk, H., & Gauvain, J. L. (2004). Neural network language models for conversational speech recognition. In *Proceedings of the 8th International Conference on Spoken Language Processing (INTERSPEECH 2004)*.

Schwenk, H., & Gauvain, J. L. (2005). Training neural network language models on very large corpora. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 201–208). Association for Computational Linguistics.

Seide, F., Li, G., Chen, X., & Yu, D. (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proceedings of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 24–29). IEEE.

Sethy, A., Georgiou, P. G., & Narayanan, S. (2006). Text data acquisition for domain-specific language models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06* (pp. 382–389). Association for Computational Linguistics, Stroudsburg, PA, USA. http://dl.acm.org/citation.cfm?id=1610075.1610129.

Sethy, A., Georgiou, P. G., Ramabhadran, B., & Narayanan, S. S. (2009). An iterative relative entropy minimization-based data selection approach for n-gram model adaptation. *IEEE Transactions on Audio, Speech, and Language Processing*, *17*(1), 13–23.

Shi, Y., Larson, M., & Jonker, C. M. (2014). Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*. doi:10.1016/j.csl.2014.11.004.

Siivola, V., Hirsimäki, T., & Virpioja, S. (2007). On growing and pruning Kneser–Ney smoothed N-gram models. *IEEE Transactions on Speech, Audio and Language Processing*, *15*(5), 1617–1624.

Sixtus, A., & Ney, H. (2002). From within-word model search to across-word model search in large vocabulary continuous speech recognition. *Computer Speech and Language*, *16*(2), 245–271.

Soltau, H., & Saon, G. (2009). Dynamic network decoding revisited. In *Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 276–281).

Tilk, O., & Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human language technologies—The Baltic perspective*, Vol. 268 (pp. 149–152). Amsterdam: IOS Press.

Varjokallio, M., & Kurimo, M. (2014a). A word-level token-passing decoder for subword n-gram LVCSR. In *Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT)* (pp. 495–500). South Lake Tahoe, California and Nevada.

Varjokallio, M., & Kurimo, M. (2014b). A toolkit for efficient learning of lexical units for speech recognition. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*. Reykjavik, Iceland.

Varjokallio, M., Kurimo, M., & Virpioja, S. (2013). Learning a subword vocabulary based on unigram likelihood. In *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Olomouc, Czech Republic.

Young, S. J., Russell, N. H., & Thornton, J. H. S. (1989). *Token passing: A simple conceptual model for connected speech recognition system*. Tech. rep., Cambridge University Engineering Department.

Appendix E

Publication V

Tilk, O. and Alumäe, T. (2017). Low-resource neural headline generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 20–26. Association for Computational Linguistics, DOI: 10.18653/v1/w17-4503

Low-Resource Neural Headline Generation

Ottokar Tilk and **Tanel Alumäe**
Department of Software Science, School of Information Technologies,
Tallinn University of Technology, Estonia
ottokar.tilk@ttu.ee, tanel.alumae@ttu.ee

Abstract

Recent neural headline generation models have shown great results, but are generally trained on very large datasets. We focus our efforts on improving headline quality on smaller datasets by the means of pre-training. We propose new methods that enable pre-training all the parameters of the model and utilize all available text, resulting in improvements by up to 32.4% relative in perplexity and 2.84 points in ROUGE.

1 Introduction

Neural headline generation (NHG) is the process of automatically generating a headline based on the text of the document using artificial neural networks.

Headline generation is a subtask of text summarization. While a summary may cover multiple documents, generally uses similar style to the summarized document, and consists of multiple sentences, headline, in contrast, covers a single document, is often written in a different style (Headlinese (Mårdh, 1980)), and is much shorter (frequently limited to a single sentence).

Due to shortness and specific style, condensing the the document into a headline often requires the ability to paraphrase which makes this task a good fit for abstractive summarization approaches where neural networks based attentive encoder-decoder (Bahdanau et al., 2015) type of models have recently shown impressive results (e.g., Rush et al. (2015); Nallapati et al. (2016)).

While state-of-the art results have been obtained by training NHG models on large datasets like Gigaword, access to such resources is often not possible, especially when it comes to low-resource languages. In this work we focus on maximizing performance on smaller datasets with different pre-training methods.

One of the reasons to expect pre-training to be an effective way to improve performance on small datasets, is that NHG models are generally trained to generate headlines based on just a few first sentences of the documents (Rush et al., 2015; Shen et al., 2016; Chopra et al., 2016; Nallapati et al., 2016). This leaves the rest of the text unutilized, which can be alleviated by pre-training subsets of the model on full documents. Additionally, the decoder component of NHG models can be regarded as a language model (LM) whose predictions are biased by the external information from the encoder. As a LM it sees only headlines during training, which is a small fraction of text compared to the documents. Supplementing the training data of the decoder with documents via pre-training might enable it to learn more about words and language structure.

Although, some of the previous work has used pre-training before (Nallapati et al., 2016; Alifimoff, 2015), it is not fully explored how much pre-training helps and what is the optimal way to do it. Another problem is, that in previous work only a subset of parameters (usually just embeddings) is pre-trained leaving the rest of the parameters randomly initialized.

The main contributions of this paper are: LM pre-training for fully initializing the encoder and decoder (sections 2.1 and 2.2); combining LM pre-training with distant supervision (Mintz et al., 2009) pre-training using filtered sentences of the documents as noisy targets (i.e. predicting one sentence given the rest) to maximally utilize the entire available dataset and pre-train all the paramters of the NHG model (section 2.3); and analysis of the effect of pre-training different components of the NHG model (section 3.3).

Proceedings of the Workshop on New Frontiers in Summarization, pages 20–26
Copenhagen, Denmark, September 7, 2017.

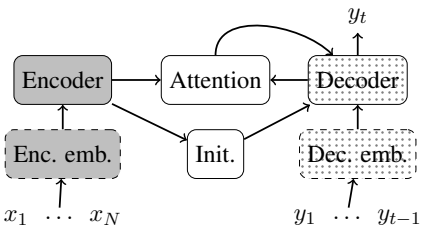


Figure 1: A high level description of the NHG model. The model predicts the next headline word y_t given the words in the document $x_1 \dots x_N$ and already generated headline words $y_1 \dots y_{t-1}$.

2 Method

The model that we use follows the architecture described by Bahdanau et al. (2015). Although originally created for neural machine translation, this architecture has been successfully used for NHG (e.g., by Shen et al. (2016); Nallapati et al. (2016) and in a simplified form by Chopra et al. (2016)).

The NHG model consists of: a bidirectional (Schuster and Paliwal, 1997) encoder with gated recurrent units (GRU) (Cho et al., 2014); a unidirectional GRU decoder; and an attention mechanism and a decoder initialization layer that connect the encoder and decoder (Bahdanau et al., 2015).

During headline generation, the encoder reads and encodes the words of the document. Initialized by the encoder, the decoder then starts generating the headline one word at a time, attending to relevant parts in the document using the attention mechanism (Figure 1). During training the parameters are optimized to maximize the probabilities of reference headlines.

While generally at the start of training either the parameters of all the components are randomly initialized or only pre-trained embeddings (with dashed outline in Figure 1) are used (Nallapati et al., 2016; Paulus et al., 2017; Gulcehre et al., 2016), we propose pre-training methods for more extensive initialization.

2.1 Encoder Pre-Training

When training a NHG model, most approaches generally use a limited number of first sentences or tokens of the document. For example Rush et al. (2015); Shen et al. (2016); Chopra et al. (2016) use only the first sentence of the document and Nallapati et al. (2016) use up to 2 first sentences. While efficient (training is faster and takes less memory as the input sequences are shorter) and effective (the most informative content tends to be at the beginning of the document (Nallapati et al., 2016)), this leaves the rest of the sentences in the document unused. Better understanding of words and their context can be learned if all sentences are used, especially on small training sets.

To utilize the entire training set, we pre-train the encoder on all the sentences of the training set documents. Since the encoder consists of two recurrent components – a forward and backward GRU – we pre-train them separately. First we add a softmax output layer to the forward GRU and train it on the sentences to predict the next word given the previous ones (i.e. we train it as a LM). After convergence on the validation set sentences, we take the embedding weights of the forward GRU and use them as fixed parameters for the backward GRU. Then we train the backwards GRU following the same procedure as with the forward GRU, with the exception of processing the sentences in a reverse order. When both models are fully trained, we remove the softmax output layers and initialize the encoder of the NHG model with the embeddings and GRU parameters of the trained LMs (highlighted with gray background in Figure 1).

2.2 Decoder Pre-Training

Pre-training the decoder as a LM seems natural, since it is essentially a conditional LM. During NHG model training the decoder is fed only headline words, which is relatively little data compared to the document contents. To improve the quality of the headlines it is essential to have high quality embeddings that are a good semantic representation of the input words and to have a well trained recurrent and output layer to predict sensible words that make up coherent sentences. When it comes to statistical models, the simplest way to improve the quality of the parameters is to train the model on more data, but it also has to be the right kind of data (Moore and Lewis, 2010).

To increase the amount of suitable training data for the decoder we use LM pre-training on filtered sentences of the training set documents. For filtering we use the XenC tool by Rousseau (2013) with the cross-entropy difference filtering (Moore and Lewis, 2010). In our case the in-domain data is training set headlines, out-domain data is the sentences from training set documents, and the best cut-off point is evaluated on validation set head-

lines. The careful selection of sentences is mostly motivated by preventing the pre-trained decoder from deviating too much from Headlinese, but it also reduces training time.

Before pre-training we initialize the input and output embeddings of the LM for words that are common in both encoder and decoder vocabulary with the corresponding pre-trained encoder embeddings. We train the LM on the selected sentences until perplexity on the validation set headlines stops improving and then use it to initialize the decoder parameters of the NHG model (highlighted with dotted background in Figure 1).

A similar approach, without data selection and embedding initialization, has also been used by Alifimoff (2015).

2.3 Distant Supervision Pre-Training

Approaches described in sections 2.1 and 2.2 enable full pre-training of the encoder and decoder, but this still leaves the connecting parameters (with white background in Figure 1) untrained.

As results in language modelling suggest, surrounding sentences contain useful information to predict words in the current sentence (Wang and Cho, 2016). This implies that other sentences contain informative sections that the attention mechanism can learn to attend to and general context that the initialization component can learn to extract.

To utilize this phenomenon, we propose using carefully picked sentences from the documents as pseudo-headlines and pre-train the NHG model to generate these given the rest of sentences in the document. Our pseudo-headline picking strategy consists of choosing sentences that occur within 100 first tokens of the document and were retained during cross-entropy filtering in section 2.2. Picking sentences from the beginning of the document should give us the most informative sentences, and cross-entropy filtering keeps sentences that most closely resemble headlines.

The pre-training procedure starts with initializing the encoder and decoder with LM pre-trained parameters (sections 2.1 and 2.2). After that, we continue training the attention and initialization parameters until perplexity on validation set headlines converges. We then use the trained parameters to initialize all parameters of the NHG model.

Distant supervision has been also used for multi-document summarization by Bravo-Marquez and Manriquez (2012).

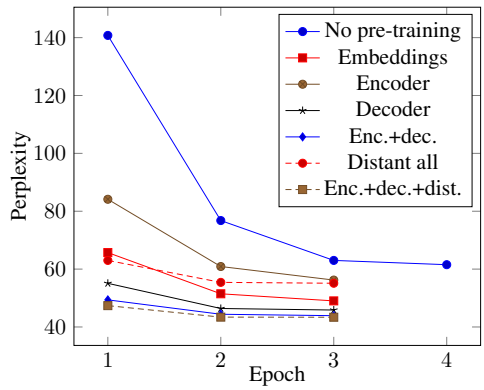


Figure 2: Validation set (EN) perplexities of the NHG model with different pre-training methods.

Model	PPL (EN)	PPL (ET)
No pre-training	65.1 ±1.0	25.9 ±0.4
Embeddings	51.8 ±0.7	20.7 ±0.3
Encoder (2.1)	59.3 ±0.9	23.5 ±0.4
Decoder (2.2)	48.3 ±0.7	18.8 ±0.3
Enc.+dec.	46.2 ±0.7	17.7 ±0.3
Distant all	58.6 ±0.9	21.3 ±0.3
Enc.+dec.+dist. (2.3)	**45.8 ±0.7**	**17.5 ±0.3**

Table 1: Perplexities on the test set with a 95% confidence interval (Klakow and Peters, 2002). All pre-trained models are significantly better than the *No pre-training* baseline.

3 Experiments

We evaluate the proposed pre-training methods in terms of ROUGE and perplexity on two relatively small datasets (English and Estonian).

3.1 Training Details

All our models use hidden layer sizes of 256 and the weights are initialized according to Glorot and Bengio (2010). The vocabularies consist of up to 50000 most frequent training set words that occur at least 3 times. The model is implemented in Theano (Bergstra et al., 2010; Bastien et al., 2012) and trained on GPUs using mini-batches of size 128. During training the weights are updated with Adam (Kingma and Ba, 2014) (parameters: α=0.001, β_1=0.9, β_2=0.999, ϵ=10^{-8} and λ=$1 - 10^{-8}$) and L_2-norm of the gradient is kept within a threshold of 5.0 (Pascanu et al., 2013). During headline generation we use beam search with beam size 5.

Model	EN				ET			
	$R1_R$	$R1_P$	RL_R	RL_P	$R1_R$	$R1_P$	RL_R	RL_P
No pre-training	20.36	33.51	17.68	29.03	26.44	34.23	25.31	32.74
Embeddings	21.09	33.36	18.23	28.72	28.42	35.94	27.02	34.16
Encoder (2.1)	21.25	34.1	18.45	29.5	**29.28**	**37.04**	**27.88**	**35.24**
Decoder (2.2)	20.11	31.1	17.43	26.87	25.12	32.6	23.89	30.99
Enc.+dec.	20.72	33.93	18.04	29.43	27.18	34.58	25.79	32.78
Distant all	20.32	31.54	17.59	27.25	26.17	34.49	24.96	32.87
Enc.+dec.+dist. (2.3)	**21.34**	**34.81**	**18.53**	**30.14**	27.74	35.46	26.35	33.67

Table 2: Recall and precision of ROUGE-1 and ROUGE-L on the test sets. Best scores in bold. Results with statistically significant differences (95% confidence) compared to *No pre-training* underlined.

3.2 Datasets

We use the CNN/Daily Mail dataset (Hermann et al., 2015)[1] for experiments on English (EN). The number of headline-document pairs is 287227, 13368 and 11490 in training, validation and test set correspondingly. The preprocessing consists of tokenization, lowercasing, replacing numeric characters with #, and removing irrelevant parts (editor notes, timestamps etc.) from the beginning of the document with heuristic rules.

For Estonian (ET) experiments we use a similarly sized (341607, 18979 and 18977 training, validation and test split) dataset that also consist of news from two sources. During preprocessing, compound words are split, words are truecased and numbers are written out as words. We used Estnltk (Orasmaa et al., 2016) stemmer for ROUGE evaluations.

3.3 Results and Analysis

Models are evaluated in terms of perplexity (PPL) and full length ROUGE (Lin, 2004). In addition to pre-training methods described in sections 2.1-2.3, we also test: initializing only the embeddings using parameters from the LM pre-trained encoder and decoder (*Embeddings*); initializing the encoder and decoder, but leaving connecting parameters randomized (*Enc.+dec.*); pre-training the whole model from random initialization with distant supervision only (*Distant all*); and a baseline that is not pre-trained at all (*No pre-training*).

All pre-training methods gave significant improvements in PPL (Table 1). The best method (*Enc.+dec.+dist.*) improved the test set PPL by 29.6-32.4% relative. Pre-trained NHG models also converged faster during training (Figure 2) and most of them beat the final PPL of the baseline already after the first epoch. General trend is that pre-training a larger amount of parameters and the parameters closer to the outputs of the NHG model improves the PPL more. *Distant all* is an exception to that observation as it used much less training data (same as baseline) than other methods.

For ROUGE evaluations, we report ROUGE-1 and ROUGE-L (Table 2). In contrast with PPL evaluations, some pre-training methods either don't improve significantly or even worsen ROUGE measures. Another difference compared to PPL evaluations is that for ROUGE, pre-training parameters that reside further from outputs (embeddings and encoder) seems more beneficial. This might imply that a better document representation is more important to stay on topic during beam search while it is less important during PPL evaluation where predicting next target headline word with high confidence is rewarded and the process is aided by previous target headline words that are fed to the decoder as inputs. It is also possible, that a well trained decoder becomes too reliant on expecting correct words as inputs making it sensitive to errors during generation which would somewhat explain why *Enc.+dec.* performs worse than *Encoder* alone. This hypothesis can be checked in further work by experimenting with methods like scheduled sampling (Bengio et al., 2015) that should increase the robustness to mistakes during generation. Pre-training all parameters on all available text (*Enc.+dec.+dist.*) still gives the best result on English and quite decent results on Estonian. Best models improve ROUGE by 0.85-2.84 points.

Some examples of the generated headlines on the CNN/Daily Mail dataset are shown in Table 3.

[1] http://cs.nyu.edu/~kcho/DMQA/

Document	a democratic congressman is at the head of a group of representatives trying to help undocumented immigrants avoid deportations with what they have called the family defender toolkit . the informational pamphlet includes a bilingual card - that some are calling a get out of deportation free card - that lists reasons a person should not be deported under expanded .
Reference headline	congressman is developing a get out of deportation toolkit to help undocumented immigrants if they are detained
No pre-training	congressman calls for undocumented immigrants
Embeddings	congressman calls for help from immigrants trying to help immigrants avoiding deportation
Encoder (2.1)	republican congressman calls for immigrants trying to avoid deportation
Decoder (2.2)	congressman who tried to stop deportations of immigrants
Enc.+Dec.	immigration congressman at the head of the head of the group who tries to avoid deportation
Distant all	congressman calls for deportation to immigrants who stay in the country
Enc.+dec.+dist. (2.3)	congressman tries to help undocumented immigrants avoid deportation
Document	a chihuahua and a bearded dragon showed off their interspecies friendship when they embarked upon a game of tag together . videoed in their front room , the dog named foxxy cleopatra and the reptile called ryuu can be seen chasing after one another around a coffee table . standing perfectly still while looking in the other direction , the bearded dragon initially appears disinterested as the chihuahua jumps around excitedly .
Reference headline	you re it!
No pre-training	is this the creepiest crawly?
Embeddings	meet the poodle!
Encoder (2.1)	it's a knockout!
Decoder (2.2)	the bearded dragon lizard: the bearded dragon lizard spotted in the middle of the street
Enc.+Dec.	oh, this is a lion!
Distant all	meet the dragon dragon: meet the dragon dragon
Enc.+dec.+dist. (2.3)	is this the world's youngest lion?

Table 3: Examples of generated headlines on CNN/Daily Mail dataset.

4 Conclusions

We proposed three new NHG model pre-training methods that in combination enable utilizing the entire dataset and initializing all parameters of the NHG model. We also evaluated and analyzed pre-training methods and their combinations in terms of perplexity (PPL) and ROUGE. The results revealed that better PPL doesn't necessarily translate to better ROUGE – PPL tends to benefit from pre-training parameters that are closer to outputs, but for ROUGE it is generally the opposite. Also, PPL benefited from pre-training more parameters while for ROUGE it was not always the case. Pre-training in general proved to be useful – our best results improved PPL by 29.6-32.4% relative and ROUGE measures by 0.85-2.84 points compared to a NHG model without pre-training.

Current work focused on maximally utilizing available headlined corpora. One interesting future direction would be to additionally utilize potentially much more abundant corpora of documents without headlines (also proposed by Shen et al. (2016)) for pre-training. Another open question is the relationship between the dataset size and the effect of pre-training.

Acknowledgments

We would like to thank NVIDIA for the donated GPU, the anonymous reviewers for their valuable comments, and Kyunghyun Cho for the help with the CNN/Daily Mail dataset.

References

Alex Alifimoff. 2015. Abstractive sentence summarization with attentive deep recurrent neural networks.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR2015, arXiv:1409.0473*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Felipe Bravo-Marquez and Manuel Manriquez. 2012. A zipf-like distant supervision approach for multi-document summarization using wikinews articles. In *International Symposium on String Processing and Information Retrieval*, pages 143–154. Springer.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and M. Alexander Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1):19–28.

Chin-Yew Lin. 2004. *Text Summarization Branches Out*, chapter ROUGE: A Package for Automatic Evaluation of Summaries.

Ingrid Mårdh. 1980. *Headlinese: On the grammar of English front page headlines*, volume 58. Liberläromedel/Gleerup.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

C. Robert Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Siim Orasmaa, Timo Petmanson, Alexander Tkachenko, Sven Laur, and Heiki-Jaan Kaalep. 2016. Estnltk - nlp toolkit for estonian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Anthony Rousseau. 2013. Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, (100):73–82.

M. Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shiqi Shen, Yu Zhao, Zhiyuan Liu, Maosong Sun, et al. 2016. Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904*.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1319–1329. Association for Computational Linguistics.

Curriculum Vitae

Personal data

Name:	Ottokar Tilk
Date of birth:	07.06.1986
Place of birth:	Tallinn, Estonia
Citizenship:	Estonia

Contact data

Email: ottokart@gmail.com

Education

2012 – ...	Tallinn University of Technology, PhD studies in computer science
2015 – 2015	Saarland University, visiting fellowship in language science (Feb – May)
2009 – 2012	Tallinn University of Technology, MSc studies in computer science
2005 – 2008	Estonian Information Technology College, professional higher education in IT systems administration

Language competence

Estonian:	Native speaker
English:	High Level

Professional employment

2013 – 2017	Institute of Cybernetics at Tallinn University of Technology; engineer
2017 – ...	Department of Software Science at Tallinn University of Technology; early stage researcher
2008 – ...	Centre of Registers and Information Systems; application administrator, .NET developer, technical architect

Summer schools

ESSLLI 2014	European Summer School of Logic, Language and Information; Tübingen, Germany
ESSCaSS	Estonian Summer School on Computer and Systems Science; 2012, 2013, 2015
EWSCS	Estonian Winter School in Computer Science; 2012 – 2014

Scientific work

[1] Tilk, O. and Alumäe, T. (2017). Low-resource neural headline generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 20–26. Association for Computational Linguistics, DOI: 10.18653/v1/w17-4503

[2] Tilk, O., Demberg, V., Sayeed, A., Klakow, D., and Thater, S. (2016). Event participant modelling with neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 171–182, Austin, Texas. Association for Computational Linguistics, DOI: 10.18653/v1/d16-1017

[3] Alumäe, T. and Tilk, O. (2016). Automatic speech recognition system for lithuanian broadcast audio. In *Human Language Technologies – The Baltic Perspective*, volume 289, pages 39–45. DOI: 10.3233/978-1-61499-701-6-39

[4] Tilk, O. and Alumäe, T. (2016). Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051. DOI: 10.21437/Interspeech.2016-1517

[5] Kurimo, M., Enarvi, S., Tilk, O., Varjokallio, M., Mansikkaniemi, A., and Alumäe, T. (2017). Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987, ISSN: 1574-0218, DOI: 10.1007/s10579-016-9336-9

[6] Tilk, O. and Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687. ISSN: 1990-9770, https://www.isca-speech.org/archive/interspeech_2015/i15_0683.html

[7] Tilk, O. and Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human Language Technologies – The Baltic Perspective*, volume 268, pages 149–152. IOS Press, DOI: 10.3233/978-1-61499-442-8-149

[8] Lohk, A., Tilk, O., and Võhandu, L. (2013). How to create order in large closed subsets of wordnet-type dictionaries. *Eesti Rakenduslingvistika Ühingu aastaraamat*, 9:149–160, DOI: 10.5128/erya9.10

Elulookirjeldus

Isikuandmed

Nimi:	Ottokar Tilk
Sünniaeg:	07.06.1986
Sünnikoht:	Tallinn, Eesti
Kodakondsus:	Eesti

Kontaktandmed

E-post: ottokart@gmail.com

Hariduskäik

2012 – ...	Tallinna Tehnikaülikool, doktoriõpingud informaatikas
2015 – 2015	Saarlandi Ülikool, külalistudeng keeleteaduses (veebruar – mai)
2009 – 2012	Tallinna Tehnikaülikool, magistriõpingud informaatikas
2005 – 2008	Eesti Infotehnoloogia Kolledž, rakenduskõrgharidusõpe IT süsteemide administreerimises

Keelteoskus

Eesti keel:	Emakeel
Inglise keel:	Kõrgtase

Töökogemus

2013 – 2017	TTÜ küberneetika instituut; insener
2017 – ...	TTÜ tarkvarateaduse instituut; nooremteadur
2008 – ...	Registrite ja Infosüsteemide Keskus; infosüsteemi haldur, .NET programmeerija, tehniline arhitekt

Suvekoolid

ESSLLI 2014	European Summer School of Logic, Language and Information; Tübingen, Saksamaa
ESSCaSS	Estonian Summer School on Computer and Systems Science; 2012, 2013, 2015
EWSCS	Estonian Winter School in Computer Science; 2012 – 2014

Teadustegevus

[1] Tilk, O. and Alumäe, T. (2017). Low-resource neural headline generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 20–26. Association for Computational Linguistics, DOI: 10.18653/v1/w17-4503

[2] Tilk, O., Demberg, V., Sayeed, A., Klakow, D., and Thater, S. (2016). Event participant modelling with neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 171–182, Austin, Texas. Association for Computational Linguistics, DOI: 10.18653/v1/d16-1017

[3] Alumäe, T. and Tilk, O. (2016). Automatic speech recognition system for lithuanian broadcast audio. In *Human Language Technologies – The Baltic Perspective*, volume 289, pages 39–45. DOI: 10.3233/978-1-61499-701-6-39

[4] Tilk, O. and Alumäe, T. (2016). Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech 2016*, pages 3047–3051. DOI: 10.21437/Interspeech.2016-1517

[5] Kurimo, M., Enarvi, S., Tilk, O., Varjokallio, M., Mansikkaniemi, A., and Alumäe, T. (2017). Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, 51(4):961–987, ISSN: 1574-0218, DOI: 10.1007/s10579-016-9336-9

[6] Tilk, O. and Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687. ISSN: 1990-9770, `https://www.isca-speech.org/archive/interspeech_2015/i15_0683.html`

[7] Tilk, O. and Alumäe, T. (2014). Multi-domain recurrent neural network language model for medical speech recognition. In *Human Language Technologies – The Baltic Perspective*, volume 268, pages 149–152. IOS Press, DOI: 10.3233/978-1-61499-442-8-149

[8] Lohk, A., Tilk, O., and Võhandu, L. (2013). How to create order in large closed subsets of wordnet-type dictionaries. *Eesti Rakenduslingvistika Ühingu aastaraamat*, 9:149–160, DOI: 10.5128/erya9.10