Automatic Speech Recognition with Very Large Conversational Finnish and Estonian Vocabularies

Seppo Enarvi, Peter Smit, Sami Virpioja, and Mikko Kurimo, *Senior Member, IEEE*

***Abstract*—Today, the vocabulary size for language models in large vocabulary speech recognition is typically several hundreds of thousands of words. While this is already sufficient in some applications, the out-of-vocabulary words are still limiting the usability in others. In agglutinative languages the vocabulary for conversational speech should include millions of word forms to cover the spelling variations due to colloquial pronunciations, in addition to the word compounding and inflections. Very large vocabularies are also needed, for example, when the recognition of rare proper names is important.**

Previously, very large vocabularies have been efficiently modeled in conventional n-gram language models either by splitting words into subword units or by clustering words into classes. While vocabulary size is not as critical anymore in modern speech recognition systems, training time and memory consumption become an issue when state-of-the-art neural network language models are used. In this paper we investigate techniques that address the vocabulary size issue by reducing the effective vocabulary size and by processing large vocabularies more efficiently.

The experimental results in conversational Finnish and Estonian speech recognition indicate that properly defined word classes improve recognition accuracy. Subword n-gram models are not better on evaluation data than word n-gram models constructed from a vocabulary that includes all the words in the training corpus. However, when recurrent neural network (RNN) language models are used, their ability to utilize long contexts gives a larger gain to subword-based modeling. Our best results are from RNN language models that are based on statistical morphs. We show that the suitable size for a subword vocabulary depends on the language. Using time delay neural network (TDNN) acoustic models, we were able to achieve new state of the art in Finnish and Estonian conversational speech recognition, 27.1 % word error rate in the Finnish task and 21.9 % in the Estonian task.

***Index Terms*—language modeling, word classes, subword units, artificial neural networks, automatic speech recognition**

I. Introduction

Finnish and Estonian are agglutinative languages, meaning that words are formed by concatenating smaller linguistic units, and a great deal of grammatical information is conveyed by inflection. Modeling these inflected words correctly is important for automatic speech recognition, to produce understandable transcripts. Recognizing a suffix correctly can also help to predict the other words in the sentence. By collecting enough training data, we can get a good coverage of the words in one form or another—perhaps names and numbers being an exception—but we are far from having enough training data to find examples of all the inflected word forms.

Another common feature of Finnish and Estonian is that the orthography is phonemic. Consequently, the spelling of a word can be altered according to the pronunciation changes in conversational language. Especially Finnish conversations are written down preserving the variation that happens in colloquial pronunciation [1]. Modeling such languages as a sequence of complete word forms becomes difficult, as most of the forms are very rare. In our data sets, most of the word forms appear only once in the training data.

Agglutination has a far more limited impact on the vocabulary size in English. Nevertheless, the vocabularies used in English language have grown as larger corpora are used and computers are able to store larger language models in memory. Moreover, as speech technology improves, we start to demand better recognition of e.g. proper names that do not appear in the training data.

Modern automatic speech recognition (ASR) systems can handle vocabularies as large as millions of words with simple n-gram language models, but a second recognition pass with a neural network language model (NNLM) is now necessary for achieving state-of-the-art performance. Vocabulary size is much more critical in NNLMs, as neural networks take a long time to train, and training and inference times depend heavily on the vocabulary size. While computational efficiency is the most important reason for finding alternatives to word-based modeling, words may not be the best choice of language modeling unit with regard to model performance either, especially when modeling agglutinative languages.

Subword models have been successfully used in Finnish ASR for more than a decade [2]. In addition to reducing the complexity of the language model, subword models bring the benefit that even words that do not occur in the training data can be predicted. However, subwords have not been used for modeling *conversational* Finnish or Estonian before. Our earlier attempts to use subwords for conversational Finnish ASR failed to improve over word models. In this paper, we show how subword models can be used in the FST-based Kaldi speech recognition toolkit and obtain the best results to date by rescoring subword lattices using subword NNLMs, 27.1 % WER for spontaneous Finnish conversations, and 21.9 % WER for spontaneous Estonian conversations. This is the first published evaluation of subwords in conversational Finnish and Estonian speech recognition tasks.

Manuscript received January 31, 2017; revised July 7, 2017; accepted August 7, 2017. This work was financially supported by the Academy of Finland under the grant numbers 251170 and 274075, and by Kone Foundation.

The authors work in the Department of Signal Processing and Acoustics at Aalto University, Espoo, Finland. (e-mail: firstname.lastname@aalto.fi)

Digital Object Identifier 10.1109/TASLP.2017.2743344

Our conclusions are slightly different from those earlier published on standard Finnish and Estonian tasks, where n-gram models based on statistical morphs have provided a large improvement to speech recognition accuracy [3], [2], [4]. An important reason is that we are able to use very large vocabularies (around two million words) in the word-based n-gram models. Recently it has been noticed that the gap between subword and word models becomes quite small when such a large word vocabulary is used [5]. In our conversational Finnish and Estonian experiments, word and subword n-gram models performed quite similarly in terms of evaluation set word error rate. Our new observation is that neural networks are especially beneficial for modeling subwords—subword NNLMs are clearly better than word NNLMs trained using the full vocabulary.

Another approach for very large vocabulary speech recognition is using word classes in the language models. We evaluate different algorithms for clustering words into classes. Recent comparisons have shown an advantage in perplexity for the exchange algorithm over Brown clustering, while clusterings created from distributed word representations have not worked as well [6], [7], [8]. We present additionally a novel rule-based algorithm that clusters colloquial Finnish word forms, and also evaluate word error rate. Surprisingly, class-based n-gram models perform better than word models in terms of perplexity and speech recognition accuracy in conversational Finnish and Estonian.

Word classes and subword units are especially attractive in NNLMs, because the vocabulary size has a great impact on the memory consumption and computational complexity. The size of the input layer projection matrix and the output layer weight matrix, as well as the time required to normalize the output probabilities using softmax, have a linear dependency on the vocabulary size. The output normalization can also be made more efficient by using one of the several methods that try to approximate the full softmax, either by modifying the network structure or the training objective. So far the only comparison of these approximations for large-vocabulary NNLMs that we are aware of is in [9]. They found hierarchical softmax to perform best in terms of perplexity with a vocabulary of 800,000 words and a feedforward network.

We compare hierarchical softmax, sampling-based softmax, class-based models, and subword models in speech recognition on languages that are known for very large vocabularies. Both data sets contain around two million unique word forms. In our experiments where the training time was limited to 15 days, class-based NNLMs clearly exceeded the performance of word-based NNLMs in terms of perplexity and recognition accuracy. The best results were from subword models. In the Estonian task, the best subword vocabularies were quite large, and the best result was from a class-based subword model. We also test two methods for weighting separate language modeling data sets: weighted sampling, which has already been introduced in [10] and update weighting, which is a novel method.

All the neural network language modeling techniques presented in this paper have been implemented in the open-source toolkit TheanoLM [11], which we hope to lower the threshold of using neural network language models in speech recognition research.[1] We implemented hierarchical softmax [12], noise-contrastive estimation [13], and BlackOut [14] training criteria, and a lattice decoder that takes advantage of parallel computation using a GPU.

We use a fairly complex recurrent model consisting of an LSTM layer and a highway network to obtain state-of-the-art results, and run the experiments on a high-end GPU. Our experiments show that class and subword models are more attractive than word models for several reasons. They are efficient computationally and in terms of memory consumption, and they can achieve better performance than word models. Usually subword vocabularies include all the individual letters, meaning that any word that uses the same letters can be constructed. Class models are restricted to a certain vocabulary, but the efficiency is not limited by the vocabulary size, so very large vocabularies can be used.

To summarize, this is the first time Finnish and Estonian subword models have outperformed word models in conversational speech recognition, even without limiting the word vocabulary size. We compare word clustering techniques and show that class-based models outperform full-vocabulary word models in these tasks. We also present the first comparison of word, class, and subword NNLMs trained using different softmax approximations, applied to speech recognition. Finally, we test a novel method for weighting NNLM training corpora.

II. Class-Based Language Models

Finnish and Estonian are highly agglutinative languages, so the number of different word forms that appear in training corpora is huge. The pronunciation variation in colloquial Finnish is also written down, making it very difficult to reliably estimate the probability of the rare words in new contexts. If we can cluster word forms into classes based on in which contexts they appear, we can get more reliable estimates for the class n-gram probabilities. In a class-based language model, the probability of a word within its class is usually modeled simply as the unigram probability of the word in the training data [15]:

$$\begin{aligned} &P(w_t \mid w_{t-n+1} \dots w_{t-1}) = \\ &P(c(w_t) \mid c(w_{t-n+1}) \dots c(w_{t-1})) P(w_t \mid c(w_t)), \end{aligned} \tag{1}$$

where $c(w)$ is a function that maps a word to a class. This is also the model that we use in this article.

A. Statistical Methods for Clustering Words into Classes

A common cost function for learning the word classes is the perplexity of a class bigram model, which is equivalent to using the log probability objective:

$$\mathcal{L} = \sum_t [\log P(c(w_t) \mid c(w_{t-1})) + \log P(w_t \mid c(w_t))] \tag{2}$$

Finding the optimal clustering is computationally very challenging. Evaluating the cost involves summation over all

[1] https://github.com/senarvi/theanolm

adjacent classes in the training data [15]. The algorithms that have been proposed are suboptimal. Another approach that can be taken is to use knowledge about the language to group words that have a similar function.

Brown et al. [15] start by assigning each word to a distinct class, and then merge classes in a greedy fashion. A naive algorithm would evaluate the objective function for each pair of classes. One iteration of the naive algorithm would on average run in $\mathcal{O}(N_V^4)$ time, where N_V is the size of the vocabulary. This involves a lot of redundant computation that can be eliminated by storing some statistics between iterations, reducing the time required to run one iteration to $\mathcal{O}(N_V^2)$.

To further reduce the computational complexity, they propose an approximation where, at any given iteration, only a subset of the vocabulary is considered. Starting from the most frequent words, N_C words are assigned to distinct classes. On each iteration, the next word is considered for merging to one of the classes. The running time of one iteration is $\mathcal{O}(N_C^2)$. The algorithm stops after $N_V - N_C$ iterations, and results in all the words being in one of the N_C classes.

The exchange algorithm proposed by Kneser and Ney [16] starts from some initial clustering that assigns every word to one of N_C classes. The algorithm iterates through all the words in the vocabulary, and evaluates how much the objective function would change by moving the word to each class. If there are moves that would improve the objective function, the word is moved to the class that provides the largest improvement.

By storing word and class bigram statistics, the evaluation of the objective function can be done in $\mathcal{O}(N_C)$, and thus one word iterated in $\mathcal{O}(N_C^2)$ time [17]. The number of words that will be iterated is not limited by a fixed bound. Even though we did not perform the experiments in such a way that we could get a fair comparison of the training times, we noticed that our exchange implementation needed less time to converge than what the Brown clustering needed to finish.[2]

These algorithms perform a lot of computation of statistics and evaluations over pairs of adjacent classes and words. In practice the running times are better than the worst case estimates, because all classes and words do not follow each other. The algorithms can also be parallelized using multiple CPUs, on the expense of memory requirements. Parallelization using a GPU would be difficult, because that would involve sparse matrices.

The exchange algorithm is greedy so the order in which the words are iterated may affect the result. The initialization may also affect whether the optimization will get stuck in a local optimum, and how fast it will converge. We use the exchange[3] tool, which by default initializes the classes by sorting the words by frequency and assigning word w_i to class $i \mod N_C$, where i is the sorted index. We compare this to initialization from other clustering methods.

B. Clustering Based on Distributed Representation of Words

Neural networks that process words need to represent them using real-valued vectors. The networks learn the word embeddings automatically. These *distributed representations* are interesting on their own, because the network tends to learn similar representation for semantically similar words [18]. An interesting alternative to statistical clustering of words is to cluster words based on their vector representations using traditional clustering methods.

Distributed word representations can be created quickly using shallow networks, such as the Continuous Bag-of-Words (CBOW) model [19]. We use word2vec[4] to cluster words by creating word embeddings using the CBOW model and clustering them using k-means.

C. A Rule-Based Method for Clustering Finnish Words

Much of the vocabulary in conversational Finnish text is due to colloquial Finnish pronunciations being written down phonetically. There are often several ways to write the same word depending on how colloquial the writing style is. Phonological processes such as reductions ("miksi" → "miks" [*why*]) and even word-internal sandhi ("menenpä" → "menempä" [*I will go*]) are often visible in written form. Intuitively grouping these different phonetic representations of the same word together would provide a good clustering. While the extent to which a text is colloquial does provide some clues for predicting the next word, in many cases these word forms serve exactly the same function.

This is closely related to normalization of imperfect text, a task which is common in all areas of language technology. Traditionally text normalization is based on hand-crafted rules and lookup tables. In the case that annotated data is available, supervised methods can be used for example to expand abbreviations [20]. When annotations are not available, candidate expansions for a non-standard word can be found by comparing its lexical or phonemic form to standard words [21]. The correct expansion often depends on the context. A language model can be incorporated to disambiguate between the alternative candidates when normalizing text. We are aware of one earlier work where colloquial Finnish has been translated to standard Finnish using both rule-based normalization and statistical machine translation [22].

Two constraints makes our task different from text normalization: A word needs to be classified in the same way regardless of the context, and a word cannot be mapped to a word sequence. Our last clustering method, *Rules*, is based on a set of rules that describe the usual reductions and alternations in colloquial words. We iterate over a standard Finnish vocabulary and compare the standard Finnish word with every word in a colloquial Finnish vocabulary. If the colloquial word appears to be a reduced pronunciation of the standard word, these words are merged into a class. Because all the words can appear in at most one class, multiple standard words can be merged into one class, but this is rare. Thus, there will be only a handful of words in each class. Larger classes can be created

[2] We are using a multithreaded exchange implementation and stop the training when the cost stops decreasing. Our observation that an optimized exchange implementation can be faster than Brown clustering is in line with an earlier comparison [6].

[3] https://github.com/aalto-speech/exchange

[4] https://code.google.com/archive/p/word2vec/

by merging the classes produced by this algorithm using some other clustering technique.

III. Subword Language Models

Subword modeling is another effective technique to reduce vocabulary size. We use the Morfessor method [23], [24], which has been successfully applied in speech recognition of many agglutinative languages [4], [25]. Morfessor is an unsupervised method that uses a statistical model to split words into smaller fragments. As these fragments often resemble the surface forms of morphemes, the smallest information-bearing units of a language, we will use the term "morph" for them.

Morfessor has three components: a model, a cost function, and the training and decoding algorithm. The model consists of a lexicon and a grammar. The lexicon contains the properties of the morphs, such as their written forms and frequencies. The grammar contains information of how the morphs can be combined into words. The Morfessor cost function is derived from MAP estimation with the goal of finding the optimal parameters θ given the observed training data D_W:

$$\begin{aligned}\theta_{MAP} &= \arg\max_{\theta} P(\theta \mid D_W) \\ &= \arg\max_{\theta} P(\theta) P(D_W \mid \theta)\end{aligned} \tag{3}$$

The objective function to be maximized is the logarithm of the product $P(\theta)P(D_W \mid \theta)$. In a semisupervised setting, it is useful to add a hyperparameter to control the weight of the data likelihood [26]:

$$\mathcal{L}(\theta, D_W) = \log P(\theta) + \alpha \log P(D_W \mid \theta) \tag{4}$$

We use the hyperparameter α to control the degree of segmentation in a heuristic manner. This allows for optimizing the segmentation, either for optimal perplexity or speech recognition accuracy or to obtain a specific size lexicon. A greedy search algorithm is used to find the optimal segmentation of morphs, given the training data. When the best model is found, it is used to segment the language model training corpus using the Viterbi algorithm.

We apply the Morfessor 2.0 implementation[5] of the Morfessor Baseline algorithm with the hyperparameter extension [27]. In the output segmentation, we prepend and append the in-word boundaries of the morph surface forms by a "+" character. For example the compound word "luentokalvoja" is segmented into "luento kalvo ja" and then transformed to "luento+ +kalvo+ +ja" [*lecture+ +slide+ +s*] before language model training. All four different variants of a subword (e.g. "kalvo", "kalvo+", "+kalvo", and "+kalvo+") are treated as separate tokens in language model training. As high-order n-grams are required to provide enough context information for subword-based modeling, we use variable-length n-gram models trained using the VariKN toolkit[6] that implements the Kneser-Ney growing and revised Kneser pruning algorithms [28].

In the speech recognition framework based on weighted finite-state transducers (FSTs), we restrict the lexicon FST in such a way that only legal sequences (meaning that a morph can start with "+" if and only if the previous morph ends with a "+") are allowed [29]. After decoding the ASR results, the morphs are joined together to form words for scoring.

IV. Neural Network Language Models

Recurrent neural networks are known to work well for modeling language, as they can capture the long-term dependencies neglected by n-gram models [30]. Especially the subword-based approach should benefit from this capability of modeling long contexts. In this article we experiment with language models that are based on LSTMs and highway networks. These are layer types that use *sigmoid gates* to control information flow. The gates are optimized along with the rest of the neural network parameters, and learn to pass the relevant activations over long distances.

LSTM [31] is a recurrent layer type. Each gate can be seen as an RNN layer with two weight matrices, W and U, a bias vector b, and sigmoid activation. The output of a gate at time step t is

$$g(x_t, h_{t-1}) = \sigma(W x_t + U h_{t-1} + b), \tag{5}$$

where x_t is the output vector of the previous layer and h_{t-1} is the LSTM layer state vector from the previous time step. When a signal is multiplied by the output of a sigmoid gate, the system learns to discard unimportant elements of the vector depending on the gate's input.

An LSTM layer uses three gates to select what information to pass from the previous time step to the next time step unmodified, and what information to modify. The same idea can be used to select what information to pass to the next layer. Highway networks [32] use gates to facilitate information flow across many layers. At its simplest, only one gate is needed. In the feedforward case, there is only one input, x_t, and the gate needs only one weight matrix, W_σ. The gate learns to select between the layer's input and its activation:

$$\begin{aligned}g(x_t) &= \sigma(W_\sigma x_t + b_\sigma) \\ y_t &= g(x_t) \odot \tanh(W x_t + b) + (1 - g(x_t)) \odot x_t\end{aligned} \tag{6}$$

While LSTM helps propagation of activations and gradients in recurrent networks, deep networks benefit from highway connections. We did not notice much improvement by stacking multiple LSTM layers on top of each other. While we did not have the possibility to systematically explore different network architectures, one LSTM layer followed by a highway network seemed to perform well. The architecture used in this article is depicted in Figure 1. Every layer was followed by Dropout [33] at rate 0.2.

The input of the network at time step t is w_t, an index that identifies the vocabulary element. The output contains the predicted probabilities for every vocabulary element, but only the output corresponding to the target word is used. The vocabulary can consist of words, word classes, subwords, or subword classes. The choice of vocabulary does not make any

[5] https://github.com/aalto-speech/morfessor

[6] https://github.com/vsiivola/variKN

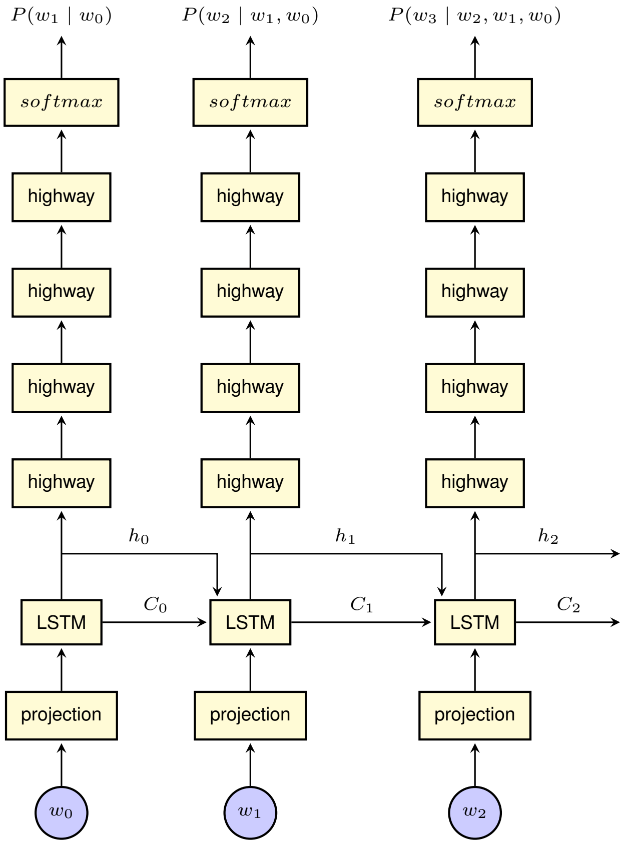

Fig. 1. The recurrent network architecture used in our experiments, unrolled three time steps. The cell state C_t of the LSTM layer conveys information over time steps until the gates choose to modify it. One gate selects part of this information as the output of the layer, h_t, which is also passed to the next time step. A highway network uses a gate to select which parts of the output are passed to the next layer unmodified and which parts are modified.

difference with regard to the neural network training, except that large vocabularies require more memory and are slower to train.

Usually word vocabularies are limited to a *shortlist* of the most frequent words. A special token such as <unk> can be used in place of any out-of-shortlist (OOS) words, which is necessary with RNN language models in particular. The NNLM can be combined with a large-vocabulary n-gram model to obtain a probability for every training word. The <unk> probability represents the total probability mass of all OOS words, which can be distributed according to n-gram language model probabilities. An n-gram language model is convenient to integrate with a feedforward NNLM, which is a particular kind of n-gram model itself, but less trivial in our RNN decoder. It also becomes computationally demanding to normalize the resulting probability distribution correctly using a large n-gram model [34].

In our baseline shortlist NNLMs we distribute the OOS probability according to a unigram model. When rescoring lattices, the output does not have to be a proper probability distribution. Assuming that the probability mass that the NNLM and n-gram models allocate to the OOS words are close to each other, a reasonable approximation is to replace the OOS probabilities with n-gram probabilities [34]. We tried this, but did not get good results because the assumption was too far from the truth. Our class NNLMs are similar to Equation 1, except that RNNs do not fix the context to n previous words—the length of the history used to predict the next word is limited only by the mini-batch size.

Memory consumption becomes a problem when using GPUs for training, since current GPU boards typically have no more than 12 GB of memory. Each layer learns a weight matrix whose dimensionality is input size by output size. For example, an NNLM with one hidden layer of size 1,000 and a vocabulary of size 100,000 requires a 1,000 by 100,000 matrix on the input and output layer. Assuming 32-bit floats are used, such a matrix uses 400 MB of memory. In addition, temporary matrices are needed when propagating the data through the network. Memory required to store the weight matrices can be reduced by using a small projection layer and a small layer before the output layer, or factorizing a large weight into the product of two smaller matrices [35]. Another possibility is to divide weight matrices to multiple GPUs. The size of the temporary data depends also on the mini-batch size.

We did the experiments with quite a complex model to see how good speech recognition accuracy we are able to achieve in these tasks. The projection layer maps words to 500-dimensional embeddings. Both the LSTM and highway network layers have 1500 outputs. With vocabularies larger than 100,000 elements we added a 500-unit feedforward layer before the output layer to reduce memory consumption. With class models mini-batches were 32 sequences and with other models 24 sequences of length 25. We also explored the possibility of training models with larger vocabularies using hierarchical softmax and sampling-based softmax. These approximations are explained in more detail in the following sections.

A. Output Normalization

The final layer normalizes the output to provide a valid probability distribution over the output classes. Normally the softmax function is used:

$$y_{t,i} = \frac{\exp(x_{t,i})}{\sum_j \exp(x_{t,j})} \tag{7}$$

At each time step t, the cross-entropy cost function requires computing the conditional probability of the target word only, $P(w_{t+1} \mid w_0 \dots w_t) = y_{t,w_{t+1}}$. Still all the activations $x_{t,j}$ are needed to explicitly normalize the probability distribution. This becomes computationally expensive, because vocabularies can be very large, and the cost of computing the normalization term scales linearly with the vocabulary size.

There has been a great deal of research on improving the speed of the softmax function by various approximations. Hierarchical NNLM is a class-based model that consists of a neural network that predicts the class and separate neural networks that predict the word inside a class [36]. This can reduce training time of feedforward networks considerably, because different n-grams are used to train each word prediction model. Hierarchical softmax is a single model that factors the output probabilities into the product of multiple softmax functions. The idea has originally been used in maximum entropy training [12], but exactly the same idea can be applied to neural networks [37]. SOUL combines a shortlist for the most frequent words with hierarchical softmax for the out-of-shortlist words [38]. Adaptive softmax [39] is a similar approach that optimizes the word cluster sizes to minimize computational cost on GPUs.

Another group of methods do not modify the model, but use sampling during training to approximate the expensive softmax normalization. These methods speed up training, but use normal softmax during evaluation. Importance sampling is a Monte Carlo method that samples words from a distribution that should be close to the network output distribution [40]. Noise-contrastive estimation (NCE) samples random words, but instead of optimizing the cross-entropy cost directly, it uses an auxiliary cost that learns to classify a word as a training word or a noise word [13]. This allows it to treat the normalization term as a parameter of the network. BlackOut continues this line of research, using a stochastic version of softmax that explicitly discriminates the target word from the noise words [14].

Variance regularization modifies the training objective to encourage the network to learn an output distribution that is close to a real probability distribution even without explicit normalization [41]. This is useful for example in one-pass speech recognition, where evaluation speed is important but the output does not have to be a valid probability distribution. The model can also be modified to predict the normalization term along with the word probabilities [42]. NCE objective also encourages the network to learn an approximately normalized distribution, and can also be used without softmax e.g. for speech recognition [43].

B. Hierarchical Softmax

Hierarchical softmax factors the output probabilities into the product of multiple softmax functions. At one extreme, the hierarchy can be a balanced binary tree that is $log_2(N)$ levels deep, where N is the vocabulary size. Each level would differentiate between two classes, and in total the hierarchical softmax would take logarithmic time. [37]

We used a two-level hierarchy, because it is simple to implement, and it does not require a hierarchical clustering of the vocabulary. The first level performs a softmax between $\sqrt{N}$ word classes and the second level performs a softmax between $\sqrt{N}$ words inside the correct class:

$$\begin{aligned}&P(w_t \mid w_0 \ldots w_{t-1}) = \\ &P(c(w_t) \mid w_0 \ldots w_{t-1}) P(w_t \mid w_0 \ldots w_{t-1}, c(w_t))\end{aligned} \tag{8}$$

This already reduces the time complexity of the output layer to the square root of the vocabulary size.

The clustering affects the performance of the resulting model, but it is not clear what kind of clustering is optimal for this kind of models. In earlier work, clusterings have been created from word frequencies [44], by clustering distributed word representations [45], and using expert knowledge [37].

Ideally all class sizes would be equal, as the matrix product that produces the preactivations can be computed efficiently on a GPU when the weight matrix is dense. We use the same word classes in the hierarchical softmax layer that we use in class-based models, but we force equal class sizes; after running the clustering algorithm, we sort the vocabulary by class and split it into partitions of size $\sqrt{N}$. This may split some classes unnecessarily into two, which is not optimal. On the other hand it is easy to implement and even as simple methods as frequency binning seem to work [44].

An advantage of hierarchical softmax compared to sampling based output layers is that hierarchical softmax speeds up evaluation as well, while sampling is used only during training and the output is properly normalized using softmax during inference.

C. Sampling-Based Approximations of Softmax

Noise-contrastive estimation [13] turns the problem from classification between N words into binary classification. For each training word, a set of noise words (one in the original paper) is sampled from some simple distribution. The network learns to discriminate between training words and noise words. The binary-valued class label C_w is used to indicate whether the word w is a training or noise word. The authors derive the probability that an arbitrary word comes from either class, $P(C_w \mid w)$, given the probability distributions of both classes. The objective function is the cross entropy of the binary classifier:

$$\begin{aligned}\mathcal{L} = \sum_w [&C_w \log P(C_w = 1 \mid w) \\ &+ (1 - C_w) \log P(C_w = 0 \mid w)]\end{aligned} \tag{9}$$

The expensive softmax normalization can be avoided by making the normalization term a network parameter that is learned along the weights during training. In a language model, the parameter would be dependent on the context words, but it turns out that it can be fixed to a context-independent constant without harming the performance of the resulting model [46]. In the beginning of the training the cost will be high and the optimization may be unstable, unless the normalization is close to correct. We use one as the normalization constant and initialize the output layer bias to the logarithmic unigram distribution, so that in the beginning the network corresponds to the maximum likelihood unigram distribution.

BlackOut [14] is also based on sampling a set of noise words, and motivated by the discriminative loss of NCE, but the objective function directly discriminates between the training word w_T and noise words w_N:

$$\mathcal{L} = \sum_{w_T} [\log P(w_T) + \sum_{w_N} \log(1 - P(w_N))] \tag{10}$$

Although not explicitly shown, the probabilities $P(w)$ are conditioned on the network state. They are computed using a weighted softmax that is normalized only on the set of training and noise words. In addition to reducing the computation, this effectively performs regularization in the output layer similarly to how the Dropout [33] technique works in the hidden layers.

Often the noise words are sampled from the uniform distribution, or from the unigram distribution of the words in the training data [46]. Our experiments confirmed that the choice of *proposal distribution* is indeed important. Using uniform distribution, the neural network optimization will not find as good parameters. With unigram distribution the problem is that some words may be sampled very rarely. Mikolov et al. [47] use the unigram distribution raised to the power of β. Ji et al. [14] make β a tunable parameter. They also exclude the correct target words from the noise distribution.

We used the power distribution with $\beta = 0.5$ for both BlackOut and NCE. We did not modify the distribution based on the target words, however, as that would introduce additional memory transfers by the Theano computation library used by TheanoLM. We observed also that random sampling from a multinomial distribution in Theano does not work as efficiently as possible with a GPU. We used 500 noise words, shared across the mini-batch. These values were selected after noting the speed of convergence with a few values. Small β values flatten the distribution too much and the optimal model is not reached. Higher values approach the unigram distribution, causing the network to not learn enough about the rare words. Using more noise words makes mini-batch updates slower, while using only 100 noise words we noticed that the training was barely converging.

These methods seem to suffer from some disadvantages. Properly optimizing the β parameter can take a considerable amount of time. A large enough set of noise words has to be drawn for the training to be stable, diminishing the speed advantage in our GPU implementation. While we did try a number of different parameter combinations, BlackOut never finished training on these data sets without numerical errors.

D. Decoding Lattices with RNN Language Models

While improving training speed is the motivation behind the various softmax approximations, inference is also slow on large networks. Methods that modify the network structure, such as hierarchical softmax, improve inference speed as well. Nevertheless, using an RNN language model in the first pass of large-vocabulary speech recognition is unrealistic. It is possible to create a list of n best hypothesis, or a word lattice, during the first pass, and rescore them using an NNLM in a second pass. We have implemented a word lattice decoder in TheanoLM that produces better results than rescoring n-best lists.

Conceptually, the decoder propagates tokens through the lattice. Each token stores a network state and the probability of the partial path. At first one token is created at the start node with the initial network state. The algorithm iterates by propagating tokens to the outgoing links of a node, creating new copies of the tokens for each link. Evaluating a single word probability at a time would be inefficient, so the decoder combines the state from all the tokens in the node into a matrix, and the input words into another matrix. Then the network is used to simultaneously compute the probability of the target word in all of these contexts.

Rescoring a word lattice using an RNN language model is equivalent to rescoring a huge n-best list, unless some approximation is used to limit the dependency of a probability on the earlier context. We apply three types of pruning, before propagation, to the tokens in the node [48]:

- **n-gram recombination**. If there are multiple tokens, whose last n context words match, keep only the best. We use $n = 22$.
- **cardinality pruning**. Keep at most c best tokens. We use $c = 62$.
- **beam pruning**. Prune tokens whose probability is low, compared to the best token. The best token is searched from all nodes that appear at the same time instance, or in the future. (Tokens in the past have a higher probability because they correspond to a shorter time period.) We prune tokens if the difference in log probability is larger than 650.

We performed a few tests with different pruning parameters and chose large enough n and c so that their effect in the results was negligible. Using a larger beam would have improved the results, but the gain would have been small compared to the increase in decoding time.

V. Experiments

A. Data Sets

We evaluate the methods on difficult spontaneous Finnish and Estonian conversations. The data sets were created in a similar manner for both languages. For training acoustic models we combined spontaneous speech corpora with other less spontaneous language that benefits acoustic modeling. For training language models we combined transcribed conversations with web data that has been filtered to match the conversational speaking style [49].

For the Finnish acoustic models we used 85 hours of training data from three sources. The first is the complete Finnish SPEECON [50] corpus. This corpus includes 550 speakers in different noise conditions that all have read 30 sentences and 30 words, numbers, or dates, and spoken 10 spontaneous sentences. Two smaller data sets of better matching spontaneous conversations were used: DSPCON [51] corpus, which consists of short conversations between Aalto University students, and FinDialogue part of the FinINTAS [52] corpus, which contains longer spontaneous conversations. For language modeling we used 61,000 words from DSPCON and 76 million words of web data. We did not differentiate between upper and lower case. This resulted in 2.4 million unique words.

For the Estonian acoustic models we used 164 hours of training data, including 142 hours of broadcast conversations, news, and lectures collected at Tallinn University of Technology [53], and 23 hours of spontaneous conversations collected at the University of Tartu[7]. These transcripts contain 1.3 million words. For language modeling we used additionally 82 million words of web data. The language model training data contained 1.8 million unique words, differentiating between upper and lower case. One reason why the Estonian vocabulary is smaller than the Finnish vocabulary, even though the Estonian data set is larger, is that colloquial Estonian is written in a more systematic way. Also standard Estonian vocabulary is smaller than standard Finnish vocabulary [25], probably because standard Finnish uses more inflected word forms.

[7] Phonetic Corpus of Estonian Spontaneous Speech. For information on distribution, see http://www.keel.ut.ee/et/foneetikakorpus.

TABLE I

Out-of-vocabulary word rates (%) of the evaluation sets, excluding start and end of sentence tokens. The last row is the full training set vocabulary, which applies also for the class models.

Vocabulary Size	Finnish	Estonian
100,000	6.67	3.89
500,000	3.36	1.59
2.4M (Fin) / 1.8M (Est)	2.31	1.01

We use only spontaneous conversations as development and evaluation data. As mentioned earlier, Finnish words can be written down in as many different ways as they can be pronounced in colloquial speech. When calculating Finnish word error rates we accept the different forms of the same word as correct, as long as they could be used in the particular context. Compound words are accepted even if they are written as separate words. However, we compute perplexities on transcripts that contain the phonetically verbatim word forms, excluding out-of-vocabulary (OOV) words. The perplexities from n-gram and neural network word and class models are all comparable to one another, because they model the same vocabulary consisting of all the training set words. Subwords can model also unseen words, so the perplexities in subword experiments are higher. OOV word rates of the evaluation sets are reported in Table I for different vocabulary sizes.

The Estonian web data is the filtered data from [49]. The same transcribed data is also used, except that we removed from the acoustic training set three speakers that appear in the evaluation set. The evaluation data is still 1236 sentences or 2.9 hours. The Finnish data is what we used in [11], augmented with 2016 data of DSPCON and read speech from SPEECON. While we now have more than doubled the amount of acoustic training data, we have only a few more hours of spontaneous conversations. The switch to neural network acoustic models had a far greater impact on the results than the additional training data. We still use the same Finnish evaluation set of 541 sentences or 44 minutes. The Finnish development and evaluation sets and reference transcripts that contain the alternative forms are included in the latest DSPCON release, without a few sentences that we could not license.

B. Models

The word based n-gram models were 4-grams, trained using the Modified Kneser-Ney implementation of SRILM toolkit [54]. Class-based models did not use Kneser-Ney smoothing, because the class n-gram statistics were not suitable for computing the Modified Kneser-Ney discount parameters. The quality of our web data is very different from the transcribed conversations, and simply pooling all the training data together would cause the larger web data to dominate the model. Instead we created separate models from different data sets, and combined them by interpolating the probabilities of the observed n-grams from the component models using weights that were optimized on the development data. In the Finnish task we created a mixture from two models, a web data model and a transcribed data model. In the Estonian task we created a mixture from three models, separating the transcribed spontaneous conversations from the broadcast conversations.

The mixture weights were optimized independently for each language model on the development data, using expectation maximization (EM). In the Finnish experiments this gave the transcribed data a weight slightly less than 0.5. In the Estonian experiments the weights of the spontaneous conversations and the web data were typically around 0.4, while the broadcasts were given a weight less than 0.2. Morph models were similarly combined from component models, but the EM optimization failed to give good weights. We used initially those optimized for the word-based models, and after the other parameters were fixed, we optimized the mixture weights for development set perplexity using a grid search with steps of 0.05.

The word clustering algorithms do not support training data weighting, so we simply concatenated the data sets. There are many parameters that can be tweaked when creating distributed word representations with word2vec. We tried clustering words using a few different parameters, and report only the best n-gram model for each class vocabulary size. Within the set of values that we tried, the best performance was obtained with continuous bag of words (CBOW), window size 8, and layer size 300 to 500.

For the subword language models, we trained Morfessor on a word list combined from all training corpora; the difference to other options such as token-based training was negligible. For each language, four segmentations were trained with α-values 0.05, 0.2, 0.5, and 1.0. This resulted in respective vocabulary sizes of 42.5k, 133k, 265k, and 468k for Finnish, and 33.2k, 103k, 212k, and 403k for Estonian. The sizes include the different morph variants with "+" prefix and affix. When training the subword n-gram models with the VariKN toolkit, the growing threshold was optimized on the development set, while keeping the pruning threshold twice as large as the growing threshold.

Word-based neural network models were trained on two shortlist sizes: 100k and 500k words. With 500k words we added a normal 500-unit layer with hyperbolic tangent activation before the output layer, which reduced memory consumption and speeded up training. The neural networks were trained using Adagrad [55] optimizer until convergence or until the maximum time limit of 15 days was reached. All neural network models were trained on a single NVIDIA Tesla K80 GPU and the training times were recorded.

We tried two different approaches for weighting the different data sets during neural network training: by randomly sampling a subset of every data set in the beginning of each epoch [10], and by weighting the parameter updates depending on from which corpus each sentence comes from. In the latter approach, the gradient is scaled by both the learning rate and a constant that is larger for higher-quality data, before updating the parameters. We are not aware that this kind of update weighting would have been used before.

Optimizing the weights for neural network training is more difficult than for the n-gram mixture models. As we do not have a computational method for optimizing the weights, we tried a few values, observing the development set perplexity during training. Sampling 20 % of the web data on each iteration, or weighting the web data by a factor of 0.4 seemed to work reasonably well. We used a slightly higher learning rate when weighting the web data to compensate for the fact that the updates are smaller on average. More systematic tests were performed using these weights with the five vocabularies in Table III.

It would be possible to train separate neural network models from each data set, but there are no methods for merging several neural networks in the similar fashion that we combine the n-gram models. Often the best possible NNLM results are obtained by interpolating probabilities from multiple models, but that kind of system is cumbersome in practice, requiring multiple models to be trained and used for inference. The layer sizes and other parameters would have to be optimized for each model separately.

We combined the NNLMs with the nonclass word or subword n-gram model by log-linear interpolation. We did not notice much difference to linear interpolation, so we chose to do the interpolation in logarithmic space, because the word probabilities may be smaller than what can be represented using 64-bit floats. We noticed that optimization of the interpolation parameters was quite difficult with our development data, so we gave equal weight to both models. In some cases it could have been beneficial to give a larger weight to the neural network model. Development data was used to select a weight for combining language model and acoustic scores from four different values.

C. Speech Recognition System

We use the Kaldi [56] speech recognition system for training our acoustic models and for first-pass decoding. The TDNN acoustic models were trained on a pure sequence criterion using Maximum Mutual Information (MMI) [57]. The data sets were cleaned and filtered using a Gaussian Mixture Model recognizer and augmented through speed and volume perturbation [58]. The number of layers and parameters of the TDNN were optimized to maximize development set accuracy on the word model. First-pass decoding was very fast with real-time factor less than 0.5. The accuracy of the first-pass recognition exceeded our earlier results on both data sets [11], [49], due to the new neural network acoustic models.

Kaldi does not, at this moment, directly support class-based decoding. Instead we created lattices using regular n-gram models, and rescored them with class n-gram and neural network models. Using the methods described in [59] it is possible to construct a word FST that represents the probabilities of a class-based language model and use this in first-pass decoding or rescoring, without explicitly using the classes. Kaldi does not have any restrictions on vocabulary size, but compiling the FST-based decoding graph from the language model, pronunciation dictionary, context dependency information, and HMM structure did consume around 60 GB of memory. The memory requirement can be lowered by reducing the number of contexts in the first-pass n-gram model.

D. Results

Table II lists perplexities and word error rates given by n-gram language models on the development data. The baseline word model performance, 30.5 % on Finnish and 23,4 % on Estonian can be compared to the various word class and subword models. We have also included results from subword classes created using the exchange algorithm for reference. The word error rates were obtained by rescoring lattices that were created using the nonclass word or subword model.

TABLE II

Results from word, class, and subword n-gram language models on development data. Includes perplexity, word error rate (%), and word error rate after interpolation with the nonclass model. Exchange algorithm is initialized using classes created with the other algorithms, in addition to the default initialization. Perplexities from word and subword models are not comparable.

Classes	Perplexity	WER	+Nonclass
Finnish, 2.4M words			
Nonclass	736	30.5	
Brown 2k	705	29.9	29.0
CBOW 2k	1017	33.2	30.2
Exchange 2k	698	29.7	29.1
Brown+Exchange 2k	701	30.0	29.0
CBOW+Exchange 2k	695	29.8	29.3
Rules+Exchange 2k	700	**29.3**	**28.9**
Brown 5k	694	29.9	29.3
CBOW 5k	861	31.4	29.8
Exchange 5k	**683**	29.9	29.3
Brown+Exchange 5k	688	29.8	29.2
CBOW+Exchange 5k	684	29.8	29.3
Rules+Exchange 5k	688	29.8	29.4
CBOW 10k	801	31.1	29.9
Exchange 10k	691	29.9	29.4
CBOW+Exchange 10k	691	29.9	29.5
Rules+Exchange 10k	690	29.9	29.4
Finnish, 42.5k subwords			
Nonclass	1127	29.9	
Exchange 5k	1433	31.2	29.8
Finnish, 133k subwords			
Nonclass	1135	30.1	
Exchange 5k	1412	31.4	30.2
Finnish, 265k subwords			
Nonclass	1128	30.2	
Exchange 5k	1334	30.6	29.2
Finnish, 468k subwords			
Nonclass	1100	30.0	
Exchange 5k	1252	30.2	29.1
Estonian, 1.8M words			
Nonclass	447	23.4	
Brown 2k	438	22.8	**22.5**
Exchange 2k	439	22.9	22.6
Brown+Exchange 2k	438	**22.6**	**22.5**
Brown 5k	432	22.7	**22.5**
Exchange 5k	432	23.0	22.6
Brown+Exchange 5k	**430**	22.8	22.7
Estonian, 33.2k subwords			
Nonclass	591	23.4	
Exchange 5k	707	24.3	23.9
Estonian, 103k subwords			
Nonclass	582	23.7	
Exchange 5k	689	24.1	23.5
Estonian, 212k subwords			
Nonclass	577	23.1	
Exchange 5k	659	23.6	23.2
Estonian, 403k subwords			
Nonclass	582	23.4	
Exchange 5k	644	23.4	23.0

In the Finnish task, 2,000, 5,000, and 10,000 class vocabularies were compared. The worst case running times of the exchange and Brown algorithms have quadratic dependency on the number of classes. With 10,000 classes, even using 20 CPUs Brown did not finish in 20 days, so the experiment was not continued. The exchange algorithm can be stopped any time, so there is no upper limit on the number of classes that can be trained, but the quality of the clustering may suffer if it is stopped early. However, with 5,000 classes and using only 5 CPUs, it seemed to converge in 5 days. Increasing the number of threads increases memory consumption. Training even 40,000 classes was possible in 15 days, but the results did not improve, so they are not reported. The most promising models were evaluated also in the Estonian task.

CBOW is clearly the fastest algorithm, which is probably why it has gained some popularity. These results show, however, that the clusters formed by k-means from distributed word representations are not good for n-gram language models. CBOW does improve, compared to the other clusterings, when the number of classes is increased. Other than that, the differences between different classification methods are mostly insignificant, but class-based models outperform word models on both languages. This result suggests that class-based softmax may be a viable alternative to other softmax approximations in neural networks. The performance of the Estonian subword models is close to that of the word model, and the Finnish subword models are better than the word model. Subword classes do not work as well, but the difference to nonclass subword models gets smaller when the size of the subword vocabulary increases.

Mostly the differences between the different initializations of the exchange algorithm seemed insignificant. However, our rule-based clustering algorithm followed by running the exchange algorithm to create 2,000 classes (Rules+Exchange 2k) gave the best word error rate on Finnish. In the NNLM experiments we did not explore with different clusterings, but used the ones that gave the smallest development set perplexity in the n-gram experiments. For Finnish the 5,000 classes created using the exchange algorithm was selected (Exchange 5k). On Estonian, initialization using Brown classes gave slightly better perplexity (Brown+Exchange 5k) and was selected for neural network models.

Table III compares training time, perplexity, and word error rate in NNLM training, when different processing is applied to the large web data set. *Uniform* means that the web data is processed just like other data sets, *sampling* means that a subset of web data is randomly sampled before each epoch, and *weighting* means that the parameter updates are given a smaller weight when the mini-batch contains web sentences. Sampling seems to improve perplexity, but not word error rate. Because sampling usually speeds up training considerably and our computational resources were limited, the rest of the experiments were done using sampling.

Table IV lists perplexities and word error rates given by neural network models on the development data. The word error rates were obtained by rescoring the same lattices as in Table II. The shortlist and word class models can predict all training set words, so the perplexities can be compared. Subword models can predict also new words, so their perplexities cannot be compared with word models. The percentage of evaluation set words that are not in the shortlist and words that are not in the training set can be found in Table I.

TABLE III
Comparison of uniform data processing, random sampling of web data by 20 %, and weighted parameter updates from web data by a factor of 0.4, in NNLM training. The models were trained using normal softmax. Includes development set perplexity, word error rate (%), and word error rate after interpolation with the n-gram model.

Subset Processing	Training Time	Perplexity	WER	+NGram
Finnish, 5k classes				
Uniform	143 h	511	26.0	25.6
Sampling	128 h	505	26.2	25.6
Weighting	101 h	521	26.4	25.5
Finnish, 42.5k subwords				
Uniform	360 h	679	25.2	**24.6**
Sampling	360 h	671	25.5	25.0
Weighting	360 h	672	**25.1**	**24.6**
Finnish, 468k subwords, 5k classes				
Uniform	141 h	790	26.0	25.0
Sampling	119 h	761	25.9	25.1
Estonian, 5k classes				
Uniform	86 h	339	**19.8**	19.9
Sampling	87 h	311	20.2	19.9
Weighting	105 h	335	20.0	**19.6**
Estonian, 212k subwords, 5k classes				
Uniform	187 h	424	20.0	19.7
Sampling	130 h	397	20.0	19.8
Weighting	187 h	409	19.9	**19.6**

The class-based models were clearly the fastest to converge, 5 to 8 days on Finnish data and 4 to 6 days on Estonian data. The experiments include shortlists of 100k and 500k words. Other shortlist models, except the Estonian 100k-word NCE, did not finish before the 360 hour limit. Consequently, improvement was not seen from using a larger 500k-word shortlist.

Our NCE implementation required more GPU memory than hierarchical softmax and we were unable to run it with the larger shortlist. With the smaller shortlist NCE was better on Finnish and hierarchical softmax was better on Estonian. We experienced issues with numerical stability using NCE with subwords, and decided to use only hierarchical softmax in the subword experiments. BlackOut training was slightly faster than NCE, but even less stable, and we were unable to finish the training without numerical errors. With hierarchical softmax we used the same classes that were used in the class-based models, but the classes were rearranged to have equal sizes as described in Section IV-B. This kind of class arrangement did not seem to improve from simple frequency binning, however.

In terms of word error rate and perplexity, class-based word models performed somewhat better than the shortlist models. The best results were from subword models. On both languages it can be seen that class-based subword models improve compared to the nonclass subword models when the vocabulary size grows. In the Finnish task, the smallest 42.5k-subword vocabulary worked well, which is small enough to use normal softmax without classes. In the Estonian task, larger subword vocabularies performed better, provided that

TABLE IV
NNLM results on development data. Word models were trained using class and shortlist vocabularies. Subword models were trained using the full subword vocabulary and on classes created from the subwords. Includes perplexity, word error rate (%), and word error rate after interpolation with the n-gram model. Perplexities from word and subword models are not comparable.

Network Output	Vocabulary	Parameters	Training Time	PPL	WER	+NGram
Finnish, 2.4M words						
HSoftmax	100k short	231M	360 h	535	26.9	25.9
NCE	100k short	230M	360 h	531	26.8	25.7
HSoftmax	500k short	532M	360 h	686	28.4	27.0
Softmax	5k classes	40M	128 h	505	26.2	25.6
Finnish, 42.5k subwords						
Softmax	42.5k full	115M	360 h	671	**25.5**	**25.0**
HSoftmax	42.5k full	116M	360 h	700	25.7	**25.0**
Softmax	5k classes	40M	146 h	857	26.2	25.5
Finnish, 133k subwords						
HSoftmax	133k full	164M	360 h	742	26.5	25.4
Softmax	5k classes	40M	190 h	811	26.1	25.4
Finnish, 265k subwords						
HSoftmax	265k full	296M	360 h	849	27.0	25.6
Softmax	5k classes	40M	133 h	813	26.2	25.3
Finnish, 468k subwords						
HSoftmax	468k full	500M	360 h	1026	28.6	26.9
Softmax	5k classes	40M	119 h	761	25.9	25.1
Estonian, 1.8M words						
HSoftmax	100k short	231M	360 h	321	20.6	19.9
NCE	100k short	230M	142 h	384	22.4	21.4
HSoftmax	500k short	532M	360 h	380	21.0	20.2
Softmax	5k classes	40M	87 h	311	20.2	19.9
Estonian, 33.2k subwords						
Softmax	33.2k full	97M	360 h	357	20.4	20.2
HSoftmax	33.2k full	97M	293 h	370	20.7	20.2
Softmax	5k classes	40M	116 h	418	20.9	20.2
Estonian, 103k subwords						
HSoftmax	103k full	134M	306 h	393	20.8	20.2
Softmax	5k classes	40M	126 h	410	20.5	19.9
Estonian, 212k subwords						
HSoftmax	212k full	243M	360 h	411	20.9	20.2
Softmax	5k classes	40M	130 h	397	**20.0**	19.8
Estonian, 403k subwords						
HSoftmax	403k full	434M	360 h	463	21.4	20.7
Softmax	5k classes	40M	124 h	395	20.3	**19.6**

the subwords were clustered into classes. The best result was obtained by clustering 403k subwords into 5,000 classes using the exchange algorithm.

Table V compares the best models on evaluation data. The best word, class, subword, and subword class n-gram and NNLM models were selected based on development set word error rate. The evaluation set results show that the advantage that the Finnish subword n-gram models had on the development set was due to the optimization of the morph segmentations on development data. Word classes are the best choice for n-gram modeling. However, neural networks seem to benefit especially subword modeling, because the overall best results are from subword NNLMs. Classes work well also in NNLMs, although the best Finnish shortlist model, 100k-word NCE, performed exceptionally well in speech recognition. Interpolation with the n-gram model gives a small but consistent improvement.

TABLE V
Performance of best n-gram and NNLM models on evaluation data. All NNLMs except the shortlist models were using softmax output. Includes perplexity, word error rate (%), and word error rate after interpolation with the nonclass or n-gram model. Perplexities from word and subword models are not comparable.

	N-Gram			NNLM		
Vocabulary	PPL	WER	+Nonclass	PPL	WER	+NGram
Finnish	Full 2.4M			Shortlist 100k (NCE)		
Word	785	31.7		618	28.1	27.9
	Rules+Exchange 2k			Exchange 5k		
Class	760	**31.4**	**31.1**	589	29.2	27.9
	Morfessor 42.5k			Morfessor 42.5k		
Subword	1313	31.7		846	**27.3**	**27.1**
	Morfessor 468k			Morfessor 468k		
Subword Class	1499	32.1	31.3	942	28.2	27.4
Estonian	Full 1.8M			Shortlist 100k (HSoftmax)		
Word	483	26.1		344	23.1	22.6
	Brown+Exchange 2k			Brown+Exchange 5k		
Class	465	**25.3**	**25.2**	324	22.2	22.2
	Morfessor 212k			Morfessor 33.2k		
Subword	628	26.0		377	22.7	22.6
	Morfessor 403k			Morfessor 403k		
Subword Class	682	25.8	25.5	403	**22.1**	**21.9**

VI. Conclusions

Our experiments show that class-based models are very attractive for conversational Finnish and Estonian speech recognition. When the vocabulary contains millions of words, class-based n-gram models perform better than normal word models. A class-based NNLM can be trained in less than a week and when the training time is limited, often performs better than word-shortlist models.

In previous work, class-based models did not outperform word-based models in recognizing standard Finnish and Estonian, such as broadcast news [5]. Improvement was made only when interpolating word and class models. One reason why word classes are especially beneficial in the conversational tasks may be that in the absence of large conversational corpora, most of our training data is from the Internet. Web data is noisy and there are many ways to write the same word.

One would expect less to be gained from using subword models, when a word model is trained from full vocabulary of millions of words. This seems to be the case, but RNNs are good at learning the structure of the language from a text that has been segmented into subwords. Subwords can also solve the vocabulary size problem with neural network models. In the Finnish task, the best results were from an NNLM trained on a relatively small 42.5k-subword vocabulary with full softmax output. In the Estonian task, the best results are from a large 403k-subword vocabulary that was clustered into 5,000 classes.

We explored the possibility of using NCE, BlackOut, or hierarchical softmax to overcome the problem of training neural networks with large output dimensionality. Generally they were slower than class-based training, and did not converge to as good a model in the 15-day time constraint, but Finnish 100k-word NCE training gave good results on the evaluation set.

The mixed results could mean that some details have been overlooked in our implementation of sampling-based softmax.

In both tasks we obtained the best word error rates from a subword NNLM interpolated with a subword n-gram model. In the Finnish task the best result was 27.1 %, which is a 14.5 % relative improvement from the 31.7 % WER given by our baseline 4-gram model. The best result in the Estonian task, 21.9 %, is a 16.1 % relative improvement from our 26.1 % baseline WER. These are the best results achieved in these tasks, and better than our previously best results by a large margin. The best previously published results are 48.4 % WER in the Finnish task [11] and 52.7 % WER in the Estonian task [49].

The corpus weighting methods that we used in NNLM training showed potential for improvement, but more thorough research should be done on how to select optimal weights.

VII. Acknowledgements

Computational resources were provided by the Aalto Science-IT project.

References

[1] S. Enarvi and M. Kurimo, "Studies on training text selection for conversational finnish language modeling," in *Proc. International Workshop on Spoken Language Translation (IWSLT)*, Dec. 2013, pp. 256–263.

[2] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to Finnish," *Computer Speech & Language*, vol. 20, no. 4, pp. 515–541, Oct. 2006.

[3] V. Siivola, T. Hirsimäki, M. Creutz, and M. Kurimo, "Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner," in *Proc. EUROSPEECH*. ISCA, Sep. 2003, pp. 2293–2296.

[4] M. Kurimo, A. Puurula, E. Arisoy, V. Siivola, T. Hirsimäki, J. Pylkkönen, T. Alumäe, and M. Saraclar, "Unlimited vocabulary speech recognition for agglutinative languages," in *Proc. HLT-NAACL*. Stroudsburg, PA, USA: Association for Computational Linguistics, Jun. 2006, pp. 487–494.

[5] M. Varjokallio, M. Kurimo, and S. Virpioja, "Class n-gram models for very large vocabulary speech recognition of Finnish and Estonian," in *Proc. International Conference on Statistical Language and Speech Processing (SLSP)*, P. Král and C. Martín-Vide, Eds. Cham, Switzerland: Springer International Publishing, Oct. 2016, pp. 133–144.

[6] R. Botros, K. Irie, M. Sundermeyer, and H. Ney, "On efficient training of word classes and their application to recurrent neural network language models," in *Proc. INTERSPEECH*. ISCA, Sep. 2015, pp. 1443–1447.

[7] J. Dehdari, L. Tan, and J. van Genabith, "Scaling up word clustering," in *Proc. NAACL HLT Demonstrations Session*, J. DeNero, M. Finlayson, and S. Reddy, Eds. Association for Computational Linguistics, Jun. 2016, pp. 42–46.

[8] M. Song, Y. Zhao, and S. Wang, "Exploiting different word clusterings for class-based rnn language modeling in speech recognition," in *Proc. ICASSP*. IEEE, Mar. 2017, pp. 5735–5739.

[9] W. Chen, D. Grangier, and M. Auli, "Strategies for training large vocabulary neural language models," in *Proc. ACL*, Aug. 2016, pp. 1975–1985.

[10] H. Schwenk and J.-L. Gauvain, "Training neural network language models on very large corpora," in *Proc. HLT/EMNLP*. Stroudsburg, PA, USA: Association for Computational Linguistics, Oct. 2005, pp. 201–208.

[11] S. Enarvi and M. Kurimo, "TheanoLM – an extensible toolkit for neural network language modeling," in *Proc. INTERSPEECH*, Sep. 2016, pp. 3052–3056.

[12] J. Goodman, "Classes for fast maximum entropy training," in *Proc. ICASSP*, vol. 1. IEEE, May 2001, pp. 561–564.

[13] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*. New Jersey, USA: Society for Artificial Intelligence and Statistics, May 2010, pp. 297–304.

[14] S. Ji, S. V. N. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey, "Blackout: Speeding up recurrent neural network language models with very large vocabularies," in *Proc. International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: https://arxiv.org/abs/1511.06909

[15] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, Dec. 1992.

[16] R. Kneser and H. Ney, "Forming word classes by statistical clustering for statistical language modelling," in *Contributions to Quantitative Linguistics*, R. Köhler and B. B. Rieger, Eds. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1993, pp. 221–226.

[17] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," in *Proc. EUROSPEECH*. ISCA, Sep. 1995, pp. 1253–1256.

[18] T. Mikolov, S. W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. NAACL HLT*. Stroudsburg, PA, USA: Association for Computational Linguistics, Jun. 2013, pp. 746–751.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. International Conference on Learning Representations (ICLR) Workshops*, 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[20] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech & Language*, vol. 15, no. 3, pp. 287–333, Jul. 2001.

[21] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a #twitter," in *Proc. ACL HLT*. Stroudsburg, PA, USA: Association for Computational Linguistics, Jun. 2011, pp. 368–378.

[22] I. Listenmaa and F. M. Tyers, "Automatic conversion of colloquial Finnish to standard Finnish," in *Proc. Nordic Conference of Computational Linguistics (NODALIDA)*, B. Megyesi, Ed. Linköping University Electronic Press / ACL, May 2015, pp. 219–223.

[23] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," in *Proc. ACL 2002 Workshop on Morphological and Phonological Learning (MPL)*, vol. 6. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 21–30.

[24] ——, "Unsupervised models for morpheme segmentation and morphology learning," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 4, no. 1, pp. 3:1–3:34, Jan. 2007.

[25] M. Creutz, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arisoy, M. Saraçlar, and A. Stolcke, "Morph-based speech recognition and modeling of out-of-vocabulary words across languages," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 5, no. 1, pp. 3:1–3:29, Dec. 2007.

[26] O. Kohonen, S. Virpioja, and K. Lagus, "Semi-supervised learning of concatenative morphology," in *Proc. Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 78–86.

[27] S. Virpioja, P. Smit, S.-A. Grönroos, and M. Kurimo, "Morfessor 2.0: Python implementation and extensions for Morfessor Baseline," Department of Signal Processing and Acoustics, Aalto University, Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, 2013.

[28] V. Siivola, T. Hirsimäki, and S. Virpioja, "On growing and pruning Kneser-Ney smoothed n-gram models," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 15, no. 5, pp. 1617–1624, 2007.

[29] P. Smit, S. Virpioja, and M. Kurimo, "Improved subword modeling for WFST-based speech recognition," in *Proc. INTERSPEECH*. ISCA, Aug. 2017, pp. 2551–2555.

[30] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. INTERSPEECH*. ISCA, Sep. 2010, pp. 1045–1048.

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

[32] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2015, pp. 2377–2385.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014.

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

[35] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, “Low-rank matrix factorization for deep neural network training with high-dimensional output targets,” in *Proc. ICASSP*. IEEE, May 2013, pp. 6655–6659.

[36] H.-K. Kuo, E. Arisoy, A. Emami, and P. Vozila, “Large scale hierarchical neural network language models,” in *Proc. INTERSPEECH*. ISCA, Sep. 2012, pp. 1672–1675.

[37] F. Morin and Y. Bengio, “Hierarchical probabilistic neural network language model,” in *Proc. International Workshop on Artificial Intelligence and Statistics (AISTATS)*, R. G. Cowell and Z. Ghahramani, Eds. New Jersey, USA: Society for Artificial Intelligence and Statistics, Jan. 2005, pp. 246–252.

[38] H. S. Le, I. Oparin, A. Messaoudi, A. Allauzen, J.-L. Gauvain, and F. Yvon, “Large vocabulary SOUL neural network language models,” in *Proc. INTERSPEECH*. ISCA, Aug. 2011, pp. 1469–1472.

[39] É. Grave, A. Joulin, M. Cissé, D. Grangier, and H. Jégou, “Efficient softmax approximation for GPUs,” in *Proc. ICML*, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, Aug. 2017, pp. 1302–1310.

[40] Y. Bengio and J.-S. Senécal, “Quick training of probabilistic neural nets by importance sampling,” in *Proc. International Workshop on Artificial Intelligence and Statistics (AISTATS)*, C. M. Bishop and B. J. Frey, Eds. New Jersey, USA: Society for Artificial Intelligence and Statistics, Jan. 2003.

[41] Y. Shi, W. Q. Zhang, M. Cai, and J. Liu, “Variance regularization of RNNLM for speech recognition,” in *Proc. ICASSP*. IEEE, May 2014, pp. 4893–4897.

[42] A. Sethy, S. F. Chen, E. Arisoy, and B. Ramabhadran, “Unnormalized exponential and neural network language models,” in *Proc. ICASSP*. IEEE, Apr. 2015, pp. 5416–5420.

[43] X. Chen, X. Liu, M. J. F. Gales, and P. C. Woodland, “Recurrent neural network language model training with noise contrastive estimation for speech recognition,” in *Proc. ICASSP*. IEEE, Apr. 2015, pp. 5411–5415.

[44] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. ICASSP*. IEEE, May 2011, pp. 5528–5531.

[45] A. Mnih and G. E. Hinton, “A scalable hierarchical distributed language model,” in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2009, pp. 1081–1088.

[46] A. Mnih and Y. W. Teh, “A fast and simple algorithm for training neural probabilistic language models,” in *Proc. ICML*, J. Langford and J. Pineau, Eds. New York, NY, USA: Omnipress, Jul. 2012, pp. 1751–1758.

[47] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2013, pp. 3111–3119.

[48] M. Sundermeyer, R. Schlüter, and H. Ney, “Lattice decoding and rescoring with long-span neural network language models,” in *Proc. INTERSPEECH*. ISCA, Sep. 2014, pp. 661–665.

[49] M. Kurimo, S. Enarvi, O. Tilk, M. Varjokallio, A. Mansikkaniemi, and T. Alumäe, “Modeling under-resourced languages for speech recognition,” *Language Resources and Evaluation (LRE)*, Feb. 2016. [Online]. Available: https://doi.org/10.1007/s10579-016-9336-9

[50] D. J. Iskra, B. Grosskopf, K. Marasek, H. van den Heuvel, F. Diehl, and A. Kießling, “SPEECON – speech databases for consumer devices: Database specification and validation,” in *Proc. International Conference on Language Resources and Evaluation (LREC)*, May 2002, pp. 329–333.

[51] Aalto University, Department of Signal Processing and Acoustics, “Aalto University DSP Course Conversation Corpus 2013–2017, Downloadable Version,” 2017. [Online]. Available: http://urn.fi/urn:nbn:fi:lb-2017092133

[52] M. Lennes, “Segmental features in spontaneous and read-aloud Finnish,” in *Phonetics of Russian and Finnish. General Introduction. Spontaneous and Read-Aloud Speech*, V. de Silva and R. Ullakonoja, Eds. Peter Lang GmbH, 2009, pp. 145–166.

[53] E. Meister, L. Meister, and R. Metsvahi, “New speech corpora at IoC,” in *Proceedings of the XXVII Fonetiikan päivät – Phonetics Symposium*, E. Meister, Ed. Tallinn, Estonia: TUT Press, Feb. 2012, pp. 30–33.

[54] A. Stolcke, “SRILM — an extensible language modeling toolkit,” in *Proc. ICSLP*, Sep. 2002, pp. 901–904.

[55] 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, Jul. 2011.

[56] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, “The Kaldi speech recognition toolkit,” in *ASRU Demonstration Session*. IEEE Signal Processing Society, Dec. 2011. [Online]. Available: http://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf

[57] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely sequence-trained neural networks for ASR based on lattice-free MMI,” in *Proc. INTERSPEECH*, Sep. 2016, pp. 2751–2755.

[58] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in *Proc. INTERSPEECH*, Sep. 2015, pp. 3586–3589.

[59] C. Allauzen, M. Mohri, and B. Roark, “Generalized algorithms for constructing statistical language models,” in *Proc. Annual Meeting of the ACL*. Sapporo, Japan: Association for Computational Linguistics, July 2003, pp. 40–47.

Seppo Enarvi received the Lic.Sc. in technology degree in computer and information science from Aalto University, Espoo, Finland, in 2012. He pursues to finish his Ph.D. on conversational Finnish speech recognition during 2017. His research interests are in machine learning, currently focusing on language modeling using neural networks.

Peter Smit received the M.Sc. in technology degree in computer science from Aalto University, Espoo, Finland, in 2011. He is currently a doctoral student in the Department of Signal Processing and Acoustics at Aalto University. His research interests are in machine learning and speech recognition and his current focus is subword-modeling techniques and automatic speech recognition for underresourced languages.

Sami Virpioja received the D.Sc. in technology degree in computer and information science from Aalto University, Espoo, Finland, in 2012. Between 2013 and 2017 was a research scientist at Lingsoft Inc., Helsinki, Finland. Currently he is a senior data scientist at Utopia Analytics Oy, Helsinki, Finland, and a postdoctoral researcher in the Department of Signal Processing and Acoustics at Aalto University. His research interests are in machine learning and natural language processing.

Mikko Kurimo (SM’07) received the D.Sc. (Ph.D.) in technology degree in computer science from the Helsinki University of Technology, Espoo, Finland, in 1997. He is currently an associate professor in the Department of Signal Processing and Acoustics at Aalto University, Finland. His research interests are in speech recognition, machine learning and natural language processing.

Erratum

A highway network layer uses a separate bias for its gate, distinguished by the index σ in Equation 6. The index was missing in the published paper.